Efficient NFV-Enabled Multicasting in SDNs

Zichuan Xu, *Member, IEEE*, Weifa Liang, *Senior Member, IEEE*, Meitian Huang, Mike Jia, Song Guo, *Senior Member, IEEE*, and Alex Galis, *Member, IEEE*

***Abstract*— Multicasting is a fundamental functionality of many network applications, including online conferencing, event monitoring, video streaming, and so on. To ensure reliable, secure, and scalable multicasting, a service chain that consists of network functions (e.g., firewalls, intrusion detection systems, and transcoders) usually is associated with each multicast request. We refer to such a multicast request with service chain requirement as an network function virtualization (NFV)-enabled multicast request. In this paper, we study NFV-enabled multicasting in a software-defined network (SDN) with an aim to maximize network throughput while minimizing the implementation cost of admitted NFV-enabled multicast requests, subject to network resource capacity, where the implementation cost of a request consists of its computing resource consumption cost in servers and its network bandwidth consumption cost when routing and processing its data packets in the network. To this end, we first formulate two NFV-enabled multicasting problems with and without resource capacity constraints and one online NFV-enabled multicasting problem. We then devise two approximation algorithms with an approximation ratio of $2M$ for the NFV-enabled multicasting problems with and without resource capacity constraints, if the number of servers for implementing the service chain of each request is no greater than a constant M ($\geq$1). We also study dynamic admissions of NFV-enabled multicast requests without the knowledge of future request arrivals with the objective to maximize the network throughput, for which we propose an efficient heuristic, and for the special case of dynamic request admissions, we devise an online algorithm with a competitive ratio of $O(\log n)$ for it when $M = 1$, where n is the number of nodes in the network. 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.**

***Index Terms*— Network function virtualization, software-defined networks, multicasting, NFV-enabled multicasting, service chains, virtualized network functions, routing, approximation, online algorithms.**

I. Introduction

TODAY'S data centers and communication networks deploy a variety of intermediary middleboxes, e.g., firewalls, Intrusion Detection Systems (IDSs), proxies, and WAN optimizers, to guarantee the security and performance of data transfers. However, considering that the middleboxes are typically made by expensive dedicated hardware and managed manually, they not only raise the capital expenditures (CapEx) and operating expenses (OpEx) of many service providers but also increase the inflexibility of their management dramatically. Network Function Virtualization (NFV) [6], [7], [12], [27] has been emerging as a promising paradigm to reduce the CapEx and OpEx of service providers, by implementing network functions as software components running in Virtual Machines (VMs). Underpinned by the NFV technique, Software-Defined Networking (SDN) can be further utilized to enable flexible implementations of network functions, by leveraging flexible control and management empowered by the separation of the control plane and the data plane.

Multicasting is a widely-used type of communication in the mentioned NFV-enabled SDNs. Each multicast request transmits data from one source to multiple destinations. The applications of multicasting include video conferencing, multimedia distribution, and software updates in data centers. In data center networks, multicasting is widely adopted by many applications from front-end delay-sensitive cloud applications, to back-end bandwidth-hungry computations [30]. Examples include directing search queries to a set of indexing servers [3], distributing executable binaries to a group of servers participating cooperative computations such as MapReduce [28], upgrading OS and software on data center servers, replicating file chunks in distributing file systems [9]. All of such applications need network functions to guarantee the security or performance of their traffic. For example, an Application Delivery Controller (ADC) may be needed to replicate the data of an application to a group of servers, to meet the security needs of the application and provide simplified authentication, authorization and accounting [1]. In particular, with the SDN technology being envisioned as the dominant technology for the next-generation data center networks [36], [37], enabling efficient NFV-enabled multicasting in SDNs becomes an urgent task. Techniques designed for multicasting in conventional networks cannot be applied to

Manuscript received January 26, 2018; revised July 2, 2018 and September 22, 2018; accepted November 8, 2018. Date of publication November 15, 2018; date of current version March 15, 2019. The work of Zichuan Xu is supported by the National Natural Science Foundation of China (Grant No. 61802048, 61802047, 61772113, 61872053), the fundamental research funds for the central universities in China (Grant No. DUT17RC(3)061, DUT17RC(3)070), and the Xinghai Scholar Program in Dalian University of Technology, China. The work by Alex Galis is supported by EU NECOS projects (777067). The work by Song Guo is supported by National Natural Science Foundation of China (Grant No. 61872310) and Shenzhen Basic Research Funding Scheme (Grant No. JCYJ20170818103849343). The associate editor coordinating the review of this paper and approving it for publication was J. Luo. *(Corresponding author: Zichuan Xu).*

Z. Xu is with the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, School of Software, Dalian University of Technology, Dalian 116621, China (e-mail: z.xu@dlut.edu.cn).

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

S. Guo is with The Hong Kong Polytechnic University Shenzhen Research Institute and Department of Computing, The Hong Kong Polytechnic University (e-mail: song.guo@polyu.edu.hk).

A. Galis is with the Department of Electronic and Electrical Engineering, University College London, London WC1E 7JE, U.K.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCOMM.2018.2881438

NFV-enabled multicasting as they do not consider the placement of virtualized network functions (VNFs). In this paper, we thus aim to enable efficient NFV-enabled multicasting in SDNs via efficient and effective placement of VNFs for multicast requests that require forwarding their traffic to some specified middleboxes before reaching their destinations. To admit multicast requests with network function requirements that will be implemented in servers as VMs, we study the problem of *NFV-enabled multicasting in a software-defined network* that is equipped with servers to run the VMs.

Performing NFV-enabled multicasting in an SDN is significantly challenging. The VMs in servers for network function implementations are located at different locations of the SDN, and this poses a great challenge to minimize the cost of implementing multicast requests. Naive placement of the VMs of each NFV-enabled multicast request at locations that are far away from the source and/or the destinations of the request may incur a prohibitive communication cost. In addition, multicast requests usually arrive in the network one by one without the knowledge of future arrivals. This leads to difficulties to estimate dynamic workloads of both computing and bandwidth resources at servers and links. The challenges thus are (1) how to jointly find one or multiple servers to implement its network functions by finding a multicast tree for each multicast request while meeting its computing and bandwidth demands; (2) how to design a novel metric that can accurately capture the dynamic resource usages and workloads in the SDN; and (3) how to devise an online algorithm with a competitive ratio to maximize the number of multicast requests admitted, subject to resource capacity constraints.

Several studies on multicasting in SDNs have been conducted recently [15], [16], [42], [43]. However, most of them did not consider network functions in multicast requests [15], [16] or dealt with a sequence of requests without future arrival knowledge [42], [43]. In contrast, we here investigate NFV-enabled multicasting, by devising an approximation algorithm with a provable approximation ratio for realizing a single NFV-enabled multicast request. We also develop a first online algorithm with a guaranteed competitive ratio for the online NFV-enabled multicasting problem. To the best of our knowledge, we are the first to formulate a novel NFV-enabled multicast problem in SDNs with the aim to minimize its implementation cost, through striving for fine trade-offs between computing and bandwidth resource consumptions if no more than M servers are employed to implement the service chain of each request. We devise the first approximation algorithm for the problem. We also study online NFV-enabled multicasting and devise the first online algorithm with a provable competitive ratio if only one server is deployed for its service chain implementation. The key ingredients in the design of both approximation and online algorithms are a series of non-trivial reductions. The developed analytical techniques for approximation and competitive ratios of the proposed algorithms may have independent of interest on the analysis of online algorithms for other optimization problems that involve different types of resource optimization.

The main contributions of this paper are as follows. We first study the problem of NFV-enabled multicasting in an SDN to minimize the implementation cost of each NFV-enabled multicast request, in terms of both computing and bandwidth resource consumptions. We then devise the first approximation algorithm with an approximation ratio of $2M$ to minimize the implementation cost of each request, assuming that the number of servers used for implementing its service chain is no more than M. We also investigate dynamic admissions of NFV-enabled multicast requests without the knowledge of their future arrivals with an aim to maximize the network throughput by proposing an online algorithm. Furthermore, we devise a novel online algorithm with a provable competitive ratio for a special case of dynamic request admissions when $M = 1$. We finally evaluate the performance of the proposed algorithms by experimental simulations. Experimental results show that the proposed algorithms outperform existing heuristics.

The rest of the paper is organized as follows. Section II reviews the related work. Section III introduces the system model, notations, and problem definitions. Section IV devises approximation algorithms for the NFV-enabled multicasting problem with and without network resource capacity constraints. Section V devises online algorithms for the online NFV-enabled multicasting problem. Section VI evaluates the performance of the proposed algorithms by experimental simulation, and Section VII concludes the paper.

II. Related Work

SDN and NFV as emerging technologies have been shaping the future networking landscape, by bringing the promise of enabling inexpensive and flexible management solutions [14], [18], [34]. Traditional routing algorithms that are designed for conventional networks are inapplicable to NFV-enabled SDNs. Novel routing algorithms that jointly performs traffic routing and NFV placements are needed. There are studies that explored the issues on placement and resource allocation for NFVs in SDNs [6], [7], [22], [26], [31], [41]. Most of these studies however do not consider multicasting in SDNs. For example, Moens and Turck [31] investigated efficient NFV placement in SDNs. They focused on a hybrid scenario where some network functions are implemented by dedicated physical hardware while others are implemented in VMs. Lukovszki and Schmid [26] studied the problem of online admission and embedding of service chains (i.e., a sequence of virtualized network functions) into a substrate network (i.e., an SDN with both bandwidth and computing resource capacities on its links and nodes, assuming that servers are installed at each node in the network. Li *et al.* [22] designed and implemented a system that enables dynamic resource provision in an NFV-enabled SDN. They also studied the problem of maximizing the total number of unicast requests that can be assigned to each service chain, by formulating an Integer Linear Programming (ILP) solution and developing a randomized rounding method. Cao *et al.* [5] considered policy-aware traffic engineering in SDNs, by assuming that the traffic has to pass a given sequence of network functions. Cohen *et al.* [7] considered NFV placement in an SDN with and without server capacity constraints for NFV-enabled unicast requests. They reduced their problems into incapacitated and capacitated facility location problems respectively, and provided a bi-criteria approximation solution

to the problems. Kuo *et al.* [21] studied how to implement a single NFV-enabled unicast request with the end-to-end delay constraint by proposing a dynamic programming solution to the problem. Guo *et al.* [11] investigated the throughput maximization problem in NFV-enabled SDNs with respect to service chaining specifications. They also studied the online version that assumes the future arrivals of requests are not known in advance. Xu *et al.* [39] studied the throughput maximization and resource optimization problem in NFV-enabled networks, by assuming that the instances of different types of service chains have been instantiated in data centers. An optimal solution is proposed for a special case of the problem, and an approximation algorithm is devised for the original problem.

There are several studies focusing on multicasting in SDNs [15], [42], [43]. Huang *et al.* [16] recently devised the first online algorithms with provable competitive ratios for online unicasting and multicasting in SDNs, under both node capacities (forwarding table sizes) and link capacities (bandwidth capacities) constraints. However, they did not consider network function requirements of unicast and multicast requests. Huang *et al.* [15] studied the scalability of multicasting in SDNs, by proposing an efficient algorithm to find a branch-aware Steiner Tree (BST) for each multicast request. Their solution is not applicable to the NFV-enabled multicasting problem, due to the lack of efficient methods to deal with NFV placements. A very closely related work to this study is the one by Zhang *et al.* [42], [43]. They investigated the NFV-enabled multicasting problem in SDNs, by assuming that there are sufficient computing and bandwidth resources to accommodate any multicast request, for which they provided a 2-approximation algorithm for the problem if only one server ($M = 1$) is deployed for implementing the service chain of each multicast request. However, their method cannot be extended to a general case of the problem where multiple servers are employed. It must be mentioned that for the sake of reliability, it is usually better to have the traffic of each multicast being processed by multiple servers ($M > 1$). In case of the failure of a server, the other servers can continue the data traffic of the request without being interrupted. Furthermore, in reality, it is not uncommon that both computing and bandwidth resources in an SDN are limited, and they need to be carefully allocated. Furthermore, Zhang *et al.* [42], [43] did not consider dynamic admissions of a sequence of multicast requests, which is much complicated compared with admitting one or a set of given requests. Since the resources allocated to current requests will heavily impact the admissions of future requests and different requests may have different resource demands, it becomes very crucial that which requests should be admitted/rejected to maximize the network throughput.

Unlike most existing studies on online multicast routing problems in traditional networks that considered either the node capacity [17], [24], [25] or the link bandwidth constraint [2], [32], we take into account both the computing resource capacity of servers and the bandwidth constraint on links in an SDN. This is a much more challenging optimization problem, due to (1) the need of innovative cost models that can accurately capture the usage costs of two different types of resources and new techniques to analyze the performance of proposed online algorithms; and (2) the need of jointly considering placing virtualized network functions to the servers and finding multicast routes for each multicast request. In addition, our problem is much more general than the Steiner tree problem, because we need to not only the server locations of VNFs but also the routing graph consisting of the server locations. Therefore, traditional Steiner tree algorithms, such as Kou *et al.*'s algorithm [20], cannot be directly applicable for the NFV-enabled multicasting problem.

III. Preliminaries

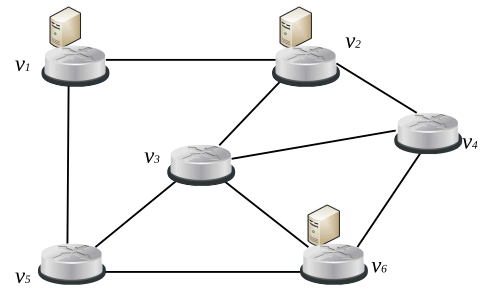
In this section we first introduce the system model, notations and notions, and then define the problems.

A. System Model

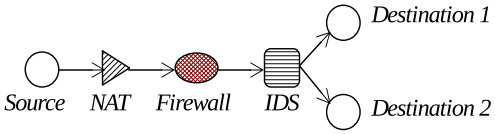
We consider a software-defined network $G = (V, E)$ with a set V of SDN-enabled switch nodes and a set E of links between SDN-enabled switch nodes. Some of the switch nodes in V are attached with computing servers that can implement various network functions as virtual machines (VMs). The communication delay between a switch node and the server attached to it usually is negligible in comparison with the communication delay with 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 servers. Notice that each node $v \in V_S$ is treated as a switch node without an attached server if its server is not used for implementing VMs. Otherwise, the VM implementation cost of v must be taken into account. Denote by C_v and B_e the computing capacity of the server attached to switch node $v \in V_S$ and the bandwidth capacity of link $e \in E$ in G, respectively. There is an SDN controller in G that controls the allocations of both computing and bandwidth resources of G to meet the resource demands of each admitted NFV-enabled multicast request. Specifically, when an SDN switch does not know how to handle the first packet of a newly arrived multicast request, it will send a message to the controller. Based on the resource availabilities of links and nodes that are collected by the controller, a route for the request will be calculated and installed in the switches in the route. Fig. 1 is an example of an SDN, where switch nodes v_1, v_2, and v_6 are attached with servers, while the rest of its nodes are not.

B. NFV-Enabled Multicast Requests and Pseudo-Multicast Trees

An NFV-enabled multicast request r_k is represented by a quadruple $r_k = (s_k, D_k; b_k, SC_k)$, where $s_k \in V$ is the source, D_k is the set of destinations (or terminals) with $D_k \subseteq V$, b_k is the demanded bandwidth by r_k, and SC_k is the *service chain* of r_k that consists of a sequence of network functions that must be implemented by either dedicated hardware middleboxes or virtual machines. Specifically, the service chain SC_k of request r_k enforces that every packet from the source of r_k to go through each of the network functions of



(a) An SDN G with a set $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ of SDN-enabled switches and a subset $V_S = \{v_1, v_2, v_6\}$ of switches that are attached with servers.



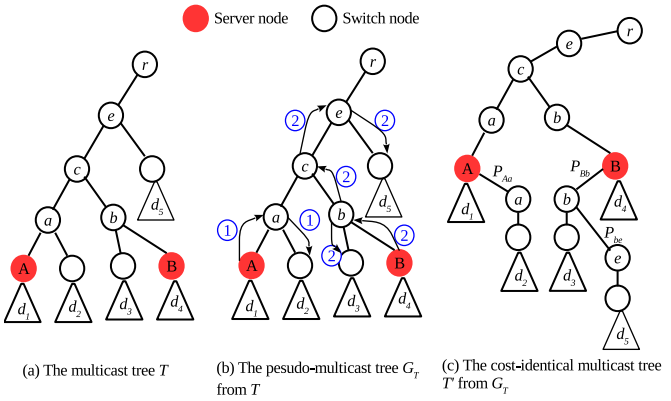
(b) A service chain $\langle$ NAT, Firewall, IDS $\rangle$.

Fig. 1. System model and a service chain example.

the service chain in the specified order prior to reaching its destinations, as illustrated in Fig. 1 (b). The network functions in SC_k can be implemented by VMs in servers [12], [29], [33], [40]. Without loss of generality, we assume that the network functions in SC_k are *consolidated to a server* in G. Specifically, when realizing multicast request r_k, its packet passes a server hosting the VMs of its service chain SC_k, the traffic will be directed to the VMs. Denote by $C_v(SC_k)$ the amount of demanded computing resource to implement SC_k of multicast request r_k in server $v \in V_S$.

A pseudo-multicast tree in G is a graph G_T derived from a multicast tree T for the data traffic routing of an NFV-enabled multicast request. We here use an example to illustrate the pseudo-multicast tree concept.

Consider a multicast tree T as shown in Fig. 2, where nodes A and B are attached with servers for processing the NFVs in SC_k of a multicast request r_k, the set of destinations is $\{d_1, d_2, d_3, d_4, d_5\}$. Recall that a packet from source s_k must pass through a server for processing the NFVs in SC_k prior to reaching all destinations. However, in this case, only the destinations d_1 and d_4 in T can correctly receive the processed packet, because there are servers at A and B respectively, while the other three destinations d_2, d_3 and d_5 cannot. To enable the packet to pass through a server before reaching d_2, d_3 and d_4, the packet routing proceeds as follows. When the packet is processed in node A, the processed packet is sent back to node a along the tree path $P_{A,a}$, node a then forwards the processed packet to d_2 (see Fig. 2(b)). Similarly, the processed packet at B will be sent back to node b along the tree path $P_{B,b}$. Assume that the distance between nodes A and e is greater than the distance between nodes B and e, the processed packet at node b will be further forwarded to node e. The processed packet will finally reach node e and be forwarded to node d_5 (see Fig. 2 (b)). We term this routing graph derived from T as a pseudo-multicast tree G_T,



(a) The multicast tree T (b) The pesudo-multicast tree G_T from T (c) The cost-identical multicast tree T' from G_T

Fig. 2. A pseudo-multicast tree G_T derived from a multicast tree T for an NFV-Enabled multicast request r_k, and another tree T' derived from G_T which has the identical cost as G_T.

as shown by Fig. 2 (b). It can be seen that another tree T' (see Fig. 2 (c)) derived from G_T will have the same cost as G_T, i.e., $c(T') = c(G_T)$.

Notice that given an NFV-enabled multicast request, its pseudo-multicast tree may not be unique, because its packet can be directed to different destinations via different paths. However, the determination of a pseudo-multicast tree can be tailored to fit different optimization objectives. For example, if the objective is to minimize the cost of implementing the request, the found pseudo-multicast tree should be able to achieve the lowest cost.

C. Problem Definitions

Given an SDN $G = (V, E)$ and a multicast request r_k $(= (s_k, D_k; b_k, SC_k))$, we consider three NFV-enabled multicasting problems with and without resource capacity constraints as follows.

Problem 1: Assuming that $G = (V, E)$ has sufficient computing and bandwidth resources to meet the resource demands of any NFV-enabled multicast request, the network operator of G charges each admitted multicast request on a pay-as-you-go basis, its major concern is its *operational cost* that is defined as the sum of costs of consumed computing and bandwidth resources by all admitted requests. Let c_e and c_v be the usage costs of one unit of bandwidth and computing resources at link $e \in E$ and server $v \in V_S$, respectively. Denote by T_k the pseudo-multicast tree for a multicast request r_k, the implementation cost of r_k is defined by

$$c(T_k) = b_k \cdot \sum_{e \in T_k} c_e + C_v(SC_k) \sum_{v \in V_S^k} c_v, \tag{1}$$

where V_S^k is the set of servers that implement the service chain of request r_k. Notice that the edge cost $b_k \cdot \sum_{e \in T_k} c_e$ is incurred due to the consumption of bandwidth resource.

Notice that one edge in G may be traversed more than once in the pseudo-multicast tree, and each time its usage cost is counted. Since the computing resource demand of the service chain of each request usually is no greater than the computing capacity of each server, we assume that the number of servers,

each of which implements the VNFs of the service chain SC_k of a single multicast request r_k, is no more than a constant M with $M \geq 1$.

The *NFV-enabled multicasting problem without SDN resource capacity constraints* in G for an NFV-enabled multicast request r_k is to find a pseudo-multicast tree such that its implementation cost is minimized, if no more than M servers are used for implementing its service chain SC_k, assuming that G has sufficient computing and bandwidth resources for the request.

Problem 2: Both computing and communication resources in G are capacitated. Then, for an incoming NFV-enabled multicast request, the network may or may not have enough resources at that moment to admit the request. Or it is too expensive to admit the request, i.e., the request should be rejected.

The *NFV-enabled multicasting problem with SDN resource capacity constraint* in G for a single NFV-enabled multicast request r_k is to find a pseudo-multicast tree for the request such that its implementation cost is minimized, if no more than M servers are used for implementing its service chain SC_k, subject to the computing and bandwidth capacities in G.

Notice that the both defined problems are NP-hard, as their special case - the traditional multicast problem without service chain requirements is NP-hard [23].

Problem 3: In reality, requests arrive into the network one by one without the knowledge of future request arrivals. We refer to this dynamic request admission as online request admissions. As G is a service SDN and open for public access, all of its computing and bandwidth resources are dynamically allocated to users based on the 'pay-as-you-go' principle, its resources may not always meet the resource demands of all requests at any time, while each arrived request must be responded by either admitting or rejecting it immediately. We thus formulate dynamic admissions of NFV-enabled multicast requests as the *online NFV-enabled multicasting problem* in G with the aim to admit as many as NFV-enabled multicast requests without the knowledge of future request arrivals, while meeting both computing and bandwidth resource demands of each admitted request, subject to both computing and bandwidth capacities on servers and links in G, assuming that no more than M servers are used to implement the service chain of each request.

D. Approximation and Competitive Ratios

Due to NP-hardness of the defined problems, we will propose approximation algorithms with approximation ratios and online algorithms with competitive ratios for them, where the approximation and competitive ratios are defined as follows, respectively.

The Approximation Ratio: Given a value $\gamma \geq 1$, a *γ-approximation algorithm* for a minimization problem P_1 is a polynomial time algorithm $\mathcal{A}$ that outputs a solution whose value is no more than γ times the value of an optimal solution for any instance I of P_1, where γ is termed as the approximation ratio of algorithm $\mathcal{A}$.

Let OPT and S be an optimal solution of the offline problem and the solution delivered by an online algorithm $\mathcal{A}'$ for a maximization problem P_2 respectively, where a sequence of requests arrives one by one without the knowledge of future request arrivals. *The competitive ratio* of the online algorithm $\mathcal{A}'$ is ξ if $\frac{S}{OPT} \geq \frac{1}{\xi}$ for any instance I of the maximization problem P_2.

For the sake of convenience, symbols used in this paper are summarized in Table I.

IV. Approximation Algorithms for the NFV-Enabled Multicasting Problem

In this section we deal with the NFV-enabled multicasting problem with and without SDN resource capacity constraints.

A. Algorithm Overview

The basic idea of the proposed approximation algorithms is to find a pseudo-multicast tree rooted at the source and spanning all destinations, and each packet from the source to destinations passes through a server in the tree, such that the cost of the tree is minimized. To this end, the finest trade-off between the computing and communication costs needs to be explored. Specifically, if a server v with a lower computing cost is included in the pseudo-multicast tree for multicast request r_k, the computing cost of implementing r_k may be reduced. This however will increase the communication cost if the location of server v is far from the destinations of r_k. On the other hand, if there are multiple servers located at different branches of the multicast tree, then the packet can pass through each of the servers to reach its destinations in D_k. This will lead to less bandwidth usages from the source to the destinations, which is achieved at the expense of high computing cost by employing multiple servers. We thus need to identify a set of servers with each implementing the service chain SC_k of r_k and find a pseudo-multicast tree including the identified server(s) on the path from the source s_k to each destination $u \in D_k$. As M is a constant, we aim to find a pseudo-multicast tree in G that contains no more than M servers and the path in the tree from s_k to each destination $u \in D_k$ must pass through one of the identified servers such that the cost of the tree is minimized.

Recall that there are $|V_S|$ switches in G with servers, clearly $M \leq |V_S|$. As a pseudo-multicast tree for any NFV-enabled multicast request can contain at least one but no more than M servers, there are at most $\binom{|V_S|}{M}$ combinations of servers that can meet the computing resource demand of service chain SC_k of request r_k. For each combination of servers, a pseudo-multicast tree in G can be identified, and the tree with the minimum cost is then used to implement r_k. We thus reduce the NFV-enabled multicast problem to a Steiner tree problem in an auxiliary undirected graph. An approximate solution to the latter returns an approximate solution to the former.

B. Approximation Algorithm Without Resource Capacity Constraints

Given an NFV-enabled multicast request r_k, we now devise an approximation algorithm for the NFV-enabled multicasting

TABLE I
SYMBOLS

Symbols	Meaning
$G=(V,E)$	a software-defined network with a set V of switches and a set E of links
v and e	a switch node $v \in V$ and a link $e \in E$
$V_S (\subset V)$	the subset of switches nodes attached with servers
C_v and B_e	computer capacity of the server attached to node $v \in V_S$ and the bandwidth capacity of link $e \in E$
$r_k=(s_k, D_k; b_k, SC_k)$	a request with source node s_k, a set D_k of destination (or terminals), an amount b_k of bandwidth resource demand, and service chain requirement SC_k
$C_v(SC_k)$	the amount of demanded computing resource to implement SC_k of multicast request r_k in the server that is attached to $v \in V_S$.
T and G_T	a multicast tree and the pseudo-multicast tree derived from it
$c(T)$	the total edge of a multicast tree T
c_e and c_v	the usage costs of one unit of bandwidth and computing resources at link $e \in E$ and server $v \in V_S$
V_S^k	the set of servers that implement the service chain of request r_k
T_k	the pseudo-multicast tree for r_k
M	the maximum number of servers that can be used to implement the service chain of each request
γ	the approximation ratio of an approximation algorithm
OPT and S	an optimal solution to the offline problem and the solution delivered by an online algorithm for a maximization problem P_2
S/OPT	the competitive ratio of an online algorithm $\mathcal{A}'$
$G_k^i=(V_k^i, E_k^i; c)$	the ith auxiliary graph constructed from graph G for request r_k, where $1 \le i \le \binom{\lvert V_S \rvert}{M}$, V_k^i is node set, E_k^i is edge set, and c is an edge weight function
s'_k	a virtual source node of r_k
T_{mst}^i	the minimum spanning tree in a complete graph consisting of nodes in $\{s'_k\} \cup D_k$
H_k^i	the subgraph of G_k^i derived from T_{mst}^i
T_k^i	Steiner tree in H_k^i
G'	a subgraph of G where edges and node with insufficient available resources are removed, which is used in algorithm `Appro_Multi_Cap`
$C_v(k)$ and $B_e(k)$	the amount of available computing and bandwidth resources at node v and link e, when multicast request r_k arrives
$c_v(k)$ and $c_e(k)$	the unit resource usage costs at server node v and link e of the kth request r_k
α and β	constant bases in exponential functions
$V_S(k)$	$=\{v \mid v \in V_S, C_v(k) \ge C_v(SC_k)\}$
$w(e)$	weight of an edge in the auxiliary graph G_k^i
v' and v''	virtual nodes for server v
σ_v and σ_e	two thresholds for node and edge costs of implementing a multicast request
$P_{x,y}^T$	a path in T between node x and node y
$\mathcal{S}(k)$ and $\mathcal{R}(k)$	the set of admitted and rejected requests by algorithm 3

problem in G without SDN resource capacity constraint, by reducing it to the Steiner tree problem in an auxiliary undirected graph $G_k^i = (V_k^i, E_k^i; c)$ with an edge weight function c for all i with $1 \le i \le \binom{|V_S|}{M}$, where $V_k^i = V \cup \{s'_k\}$, $E_k^i = E \cup \{(s'_k, v) \mid v \in V_S^i\}$, V_S^i $(\subseteq V_S)$ is the ith combination of servers in V_S, and s'_k is a *virtual source* of request r_k. For each $v \in V_S^i$, if edge $(s_k, v) \in E$ in G, the cost of edge $(s_k, v) \in E_k^i$ is assigned zero. s'_k is the new source in G_k^i, replacing the original source s_k. Notice that the original source s_k is still contained in G_k^i serving as a 'regular' switch node without an attached server. To guarantee that the traffic of r_k passes through its service chain SC_k that is implemented in one or multiple servers in V_S^i $(\subseteq V_S)$, we connect s'_k with all server nodes in V_S^i, where the edge between s'_k and each server node $v \in V_S^i$ in G_k^i represents a shortest path $p_{s_k,v}$ in G between nodes s_k and v. The weight of edge (s'_k, v) is the cost sum of the edges in path $p_{s_k,v}$ plus the cost of implementing SC_k in server v, i.e., $c_{(s'_k,v)} = \sum_{e \in p_{s'_k,v}} c_e \cdot b_k + c_v(SC_k)$, where $c_v(SC_k)$ is the cost of the amount $C_v(SC_k)$ of computing resource consumption for implementing SC_k. In addition, the weight c_e of each edge $e \in E_k^i \cap E$ is the cost $c_e \cdot b_k$ of allocating the amount b_k of bandwidth resource to request r_k on edge $e \in E$. An example of the constructed auxiliary graph G_k^i that is derived from the SDN in Fig. 1 is shown in Fig. 3.

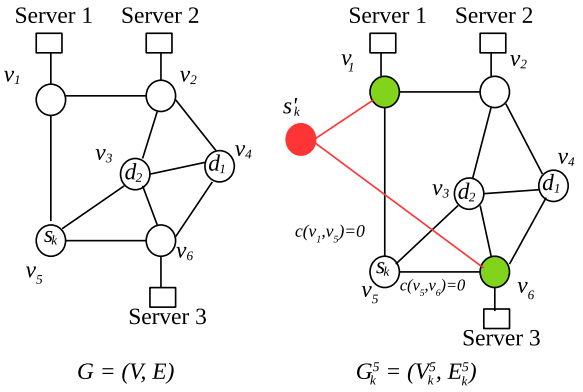
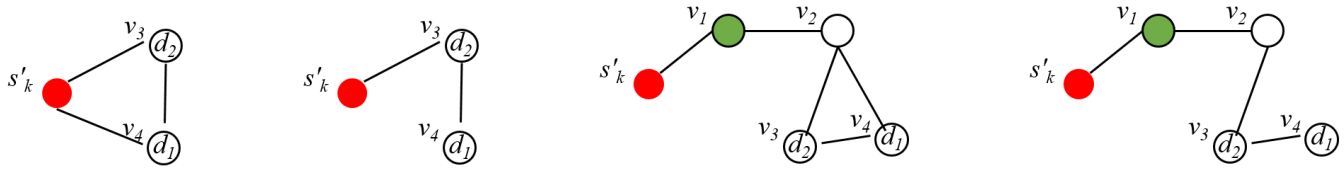


Fig. 3. An example of the auxiliary graph $G_k^5 = (V_k^5, E_k^5)$ constructed from an SDN $G = (V, E)$ with $V_S^5 = \{v_1, v_6\}$, assuming that $M = 2$ and $V_S = \{v_1, v_2, v_6\}$. There are $\binom{|V_S|}{M} = 3 \cdot 2 = 6$ auxiliary graphs derived from G, and all different combinations of servers in V_S are $V_S^1 = \{v_1\}$, $V_S^2 = \{v_2\}$, $V_S^3 = \{v_6\}$, $V_S^4 = \{v_1, v_2\}$, $V_S^5 = \{v_1, v_6\}$, and $V_S^6 = \{v_2, v_6\}$.

For the sake of convenience, in the rest of this paper we assume that $V_S = \{v_1, v_2, \ldots, v_{|V_S|}\}$. Having constructed the auxiliary graph G_k^i, we now find a Steiner tree in G_k^i for request r_k. We first find a minimum spanning tree (MST) T_{mst}^i in a complete graph consisting of nodes in $\{s'_k\} \cup D_k$, in which each edge is assigned a weight that is equal to the length of



(a) A complete graph induced by the nodes in $\{s'_k, d_2, d_1\}$.

(b) A MST T^i_{mst} based on the complete graph built in (a)

(c) A subgraph H^i_k of G^5_k constructed based on T^i_{mst} by replacing edges (s'_k, d_2) and (d_1, d_2) in T^i_{mst} with paths $\{(s'_k, v_1), (v_1, v_2), (v_2, v_3)\}$ and $\{(v_4, v_2), (v_2, v_3)\}$ in G^5_k, respectively.

(d) The final pseudo-multicast tree by finding a Steiner tree in H^i_k.

Fig. 4. An example of steps of algorithm Appro_Multi based on auxiliary graph G^5_k in Fig. 3. Notice that edges (s'_k, d_2), (s'_k, d_1), and (d_1, d_2) in the complete graph of (a) corresponds to paths $p_{s'_k,d_2} = \{(s'_k, v_1), (v_1, v_2), (v_2, v_3)\}$, $p_{s'_k,d_1} = \{(s'_k, v_6), (v_6, v_4)\}$, and $p_{d_1,d_2} = \{(v_4, v_2), (v_2, v_3)\}$ in G^5_k, respectively.

the shortest path in G^i_k between its two endpoints. Let H^i_k be a subgraph of G^i_k derived from T^i_{mst} by replacing each edge of T^i_{mst} with its corresponding shortest path in G^i_k. We then find an approximate Steiner tree T^i_k in H^i_k, by applying the approximation algorithm due to Kou *et al.* [20], which will serve as the pseudo-multicast tree for r_k. Although there are more efficient approximation algorithms for the Steiner problem such as [4] and [35], either they are randomized algorithms and their solutions are based on randomized rounding, which may not be feasible, or they have theoretical interest by giving better approximation ratios but take a much longer running time, which may not be practical under our application scenario where each request must be responded immediately, by either admitting or rejecting the request; otherwise, users may not be happy to the service and will escape from the services. While Kou *et al.*'s algorithm can produce a quality solution in a reasonable amount of time, and the solution is no greater than twice the optimal one. Furthermore, Kou *et al.*'s algorithm can be easily implemented distributively.

The detailed algorithm for the NFV-enabled multicast problem without SDN resource capacity constraints is given in Algorithm 1.

To illustrate the steps of Algorithm 1, we use G^5_k in Fig. 3 as an example to show how the algorithm works in finding MST T^i_{mst} and H^i_k in Fig. 4.

Algorithm 1 Appro_Multi

Input: $G = (V, E)$, V_S, a multicast request $r_k = (s_k, D_k; b_k, SC_k)$, and $M \geq 1$.

Output: A pseudo-multicast tree T_k for implementing the multicast request r_k with the minimum cost.

1: $cost_k \leftarrow \infty$; $T_k \leftarrow \emptyset$; /* the cost of the pseudo-multicast tree */
2: /* each combination of choosing i servers from $|V_S|$ servers */;
3: **for** $i \leftarrow 1$ to $\binom{|V_S|}{M}$ **do**
4: Construct an auxiliary undirected graph $G^i_k = (V^i_k, E^i_k)$, as illustrated in Fig. (3);
5: Find an MST T^i_{mst} in a complete graph induced by the nodes in $\{s'_k\} \cup D_k$ with the weight of each edge being the length of the shortest path in G^i_k between its two endpoints;
6: Let H^i_k be a subgraph of G^i_k derived from T^i_{mst}, by replacing each edge of T^i_{mst} with the corresponding shortest path in G^i_k; find an approximate Steiner tree T^i_k in H^i_k rooted at s'_k and spanning nodes in D_k, by invoking the approximation algorithm due to Kou *et al.* [20];
7: **if** $c(T^i_k) < cost_k$ **then**
8: $cost_k \leftarrow c(T^i_k)$, $T_k \leftarrow T^i_k$; /* a candidate solution to the problem */
9: **if** T_k contains node s_k **then**
10: Merge nodes s_k and s'_k into s'_k;
11: Rename s'_k in T_k as s_k, and let T_k be the resulting graph (the pseudo-multicast tree) for data traffic routing of request r_k;
12: **return** T_k and its cost $c(T_k)$.

C. Approximation Algorithm With Resource Capacity Constraints

We now deal with the NFV-enabled multicasting problem under both computing and bandwidth resource capacity constraints, by performing minor modifications to Algorithm 1. Specifically, a subgraph $G' = (V', E')$ of G is constructed, where $V' = V$, $E' = \{(u, v) \mid (u, v) \in E$, and the residual bandwidth at link (u, v) is no less than $2 \cdot b_k\}$, a subset set V'_S of V_S will be used, and $V'_S = \{v_i \mid v_i \in V_S$ if the available computing resource at v_i can meet the computing resource demands of $r_k\}$. Notice that link e is in E' only if its residual bandwidth is no less than 2 b_k. Specifically, an amount b_k of bandwidth resource is reserved for the traffic of r_k before being processed by its service chain. Another b_k bandwidth resource is reserved to its traffic after being processed by its service chain in case the processed traffic traverse the same link of its original traffic. This is shown in Lemma 2.

Algorithm 1 then is applied to graph G', using the server set V'_S. Clearly, all the resource demands by r_k will be met. In case G' is disconnected, and none of its connected components contains the source and all destinations of r_k and at least one server node, then the request should be rejected, because there are no sufficient resources in G for

its implementation. For simplicity, this algorithm is referred to as algorithm `Appro_Multi_Cap`.

Notice that in this paper we adopt a conservative way to perform resource allocation when admitting requests, by assuming that the system must have the sufficient resources to meet the resource demand of any admitted request r_k. However, reserving at least 2 b_k amounts of bandwidth in each link may exclude some links with less than $2b_k$ but greater than b_k. Consequently, a few request may be rejected despite that there are sufficient resources for its admission. To this end, a greedy strategy can be adopted. That is, it first applies algorithm 1 in a subgraph with link residual capacity no less than b_k. If a pseudo-multicast tree for the request can be derived from the solution, then, the request is admitted. Otherwise, we apply algorithm 1 with each link of at least 2 b_k residual capacity for the request. If a pseudo-multicast tree can be obtained, the request is admitted; it is rejected otherwise. In practice, a rejected request may not be rejected, and instead, the request can be placed to a waiting queue until it is admitted when the system has sufficient resources to meet its resource demands.

D. Algorithm Analysis

The rest is to show the correctness of `Algorithm Appro_Multi` and `Appro_Multi_Cap`, and analyze its time complexity and the approximation ratio.

Lemma 1: Algorithm `Appro_Multi` delivers a feasible solution for the NFV-enabled multicasting problem with and without SDN resource capacity constraints.

See Appendix for the detailed proof.

To show the solution of algorithm `Appro_Multi_Cap` is feasible, we need to ensure that the solution delivered by it meets the network bandwidth capacity constraint on each link $e \in E$ as shown in Lemma 2.

Lemma 2: Given an SDN $G = (V, E)$ and a sequence of NFV-enabled multicast requests, let request $r_k = (s_k, D_k; b_k, SC_k)$ be the kth multicast request, the solution delivered by algorithm `Appro_Multi_Cap` must preserve the network bandwidth capacity of each link, if the residual bandwidth of each edge in $G_k^i(V_k^i, E_k^i; w)$ is no less than $2b_k$ for all i with $1 \leq i \leq \binom{|V_S|}{M}$.

See the Appendix for the detailed proof.

Theorem 1: Given an SDN $G = (V, E)$, a set V_S of switch nodes with each having an attached server, and an NFV-enabled multicast request $r_k = (s_k, D_k; b_k, SC_k)$, there is a $2M$ approximation algorithm, `Algorithm 1`, for the NFV-enabled multicasting problem with and without SDN resource capacity constraints, assuming no more than M servers will be employed for its service chain implementation, where the approximation ratio $2M$ is the best. The time complexity of the algorithm is $O(|V|^3 \cdot |V_S|^M)$, where $|V_S| \ll |V|$ and $M \geq 1$ is a small integer.

See Appendix for the detailed proof.

V. Online Algorithms for the Online NFV-Enabled Multicasting Problem

In this section we study the online NFV-enabled multicasting problem in G with network resource capacity constraints. We first propose a novel cost model to capture dynamic resource consumptions in G. We then develop an efficient online heuristic for the problem based on the proposed cost model. We finally devise an online algorithm with a competitive ratio for a special case of the problem when $M = 1$.

A. Cost Model

Given an SDN $G = (V, E)$ with limited computing and bandwidth capacities at its servers and links, there is a need of a cost model to capture dynamic consumption of these resources in order to better guide the admissions of future requests and utilize the resources. A simple cost model, referred to *the linear cost model*, is widely adopted to assign each request with a cost that is proportional to the amount of its resource consumption regardless of whether a specific resource is overloaded or under-loaded. Clearly, this model may lead to some resources being under-utilized while others being over-utilized. Consequently, the significant number of requests may be rejected due to unbalanced resource utilization. Intuitively, overloaded resources usually have higher probabilities of violating the resource demands of admitted requests, due to the high dynamics of resource consumptions. This eventually will affect the admissions of future requests. Therefore, we encourage the use of under-loaded resources while restricting the use of overloaded resources to maximize the number of NFV-enabled multicast request admissions.

Motivated by the above concern, we here introduce a novel cost model that assigns an under-loaded resource with a lower cost and an overloaded resource with a higher cost. Thus, the resources in the network can be maximally allocated among user requests, thereby maximizing the network throughput. Specifically, let $C_v(k)$ be the amount of available computing resource at the server attached to a switch node $v \in V_S$ and $B_e(k)$ the amount of available bandwidth at link $e \in E$, respectively, when multicast request r_k arrives.

To capture the resource use of r_k, we use exponential functions to represent the costs $c_v(k)$ and $c_e(k)$ of its usages of both computing and bandwidth resources at server node v and link e:

$$c_v(k) = C_v(\alpha^{1-\frac{C_v(k)}{C_v}} - 1), \tag{2}$$

where α is a constant with $\alpha > 1$, $C_v(k) = C_v(k-1) - C_v(SC_k)$ if r_k is admitted and $C_v(0) = C_v$ initially. $(1 - \frac{C_v(k)}{C_v})$ in Eq. (2) is the utilization ratio of computing resource at server $v \in V_S$. The rationale behind is that the use of less residual computing resource will be charged with a higher cost, while the use of plenty of residual computing resource will be charged with a much less cost.

The cost $c_e(k)$ of using the bandwidth resource at link e prior to the admission of r_k can be similarly defined, i.e.,

$$c_e(k) = B_e(\beta^{1-\frac{B_e(k)}{B_e}} - 1), \tag{3}$$

where β is a constant with $\beta > 1$, $B_e(k) = B_e(k-1) - b_k$ if r_k is admitted, and $B_e(0) = B_e$ initially. Notice that the values of α and β reflect the sensitivity of resource usages. A larger value implies that its utilization of a specific resource is more sensitive.

B. Online Algorithm

For each incoming multicast request r_k $(= (s_k, D_k; b_k, SC_k))$, denote by $V_S(k)$ the subset of servers in V_S that have sufficient residual computing resources to implement its service chain SC_k, i.e., $V_S(k) = \{v \mid v \in V_S, C_v(k) \geq C_v(SC_k)\}$. It thus follows that there are $\binom{|V_S(k)|}{M}$ combinations of servers that can meet the computing resource demand of the service chain SC_k. For each combination of servers in V_S^i, a pseudo-multicast tree T_k^i in an undirected auxiliary graph $G_k^i = (V_k^i, E_k^i; w)$ can be identified, and a pseudo-multicast tree with the minimum weight among all T_k^i with $1 \leq i \leq \binom{|V_S(k)|}{M}$ is then used for the request implementation. The construction of $G_k^i = (V_k^i, E_k^i; w)$ is given as follows.

Given an SDN $G = (V, E)$ and an NFV-enabled multicast request $r_k = (s_k, D_k; b_k, SC_k)$, we construct an auxiliary undirected graph $G_k^i = (V_k^i, E_k^i; w)$ for all i with $1 \leq i \leq \binom{|V_S(k)|}{M}$, where $V_k^i = V \cup \{s_k'\} \cup \{v', v'' \mid v \in V_S^i\}$ and $V_S(k) \subseteq V) s = \{v \mid v \in V_S$ and $C_v(k) \geq c_v(SC_k)\}$, $E_k^i = \{e \mid e \in E$ and $B_e(k) \geq 2\, b_k\} \cup \{(s_k', v'), (v', v'') \mid v \in V_S^i\} \cup \{(v'', u) \mid (v, u) \in E, v \in V_S^i, u \in V,$ and $B_{e=(u,u)} \geq 2b_k\}$, V_S^i $(\subseteq V_S(k))$ is the i-th combination of the servers in $V_S(k)$, and s_k' is a virtual source of request r_k. Fig. 5 illustrates the construction of G_k^i when $i = 5$ and $M = 2$.

Intuitively, each edge in $\{(s_k', v') \mid v \in V_S^i\}$ represents a shortest path $p_{s_k,v}$ in $G_k = (V_k, E_k; w)$ between nodes s_k and $v \in V_S$ while each edge in $\{(v', v'') \mid v \in V_S^i\}$ represents the server usage cost at v, where $V_k = V$, $E_k = E$, and $w(e)$ is the normalized cost of each edge $e \in E_k$ defined in Eq. (3). We then assign each edge $e \in E_k^i$ a weight in G_k^i as follows.

$$w(e) = \begin{cases} \sum_{e \in p_{s_k,v}} \dfrac{c_e(k)}{B_e} & \text{if } e = (s_k', v') \text{ with } v \in V_S^i, \\ \dfrac{c_v(k)}{C_v} & \text{if } e = (v', v'') \text{ with } v \in V_S^i, \\ \dfrac{c_{(v,u)}(k)}{B_{(u,v)}} & \text{if } e = (v'', u) \text{ with } v \in V_S^i,\ u \in V, \\ & \text{and } (v, u) \in E, \\ \dfrac{c_e(k)}{B_e} & \text{otherwise,} \end{cases} \tag{4}$$

where $c_v(k)$ and $c_e(k)$ are the usage costs of computing resource at server node v and the bandwidth resource at link $e \in E$ when request r_k arrives, as defined in Eqs. (2) and (3), respectively. In other words, the differences between G_k^i and G are that we remove the edges without sufficient residual bandwidth to implement multicast request r_k, and add edges between virtual source s_k' and every v' (where $v \in V_S^i$) to represent a shortest path in G between s_k and v and edges (v', v'') to represent the resource usage cost of the server at v, and connect v'' to each neighbor node of v in G. It is worth noting that since an amount b_k of bandwidth resource along the edges in the shortest path $p_{s_k,v}$ for each $v \in V_S$ is reserved for the data traffic of r_k from s_k to server v for processing, each edge $(v_i, v_j) \in p_{s_k,v}$ is in E_k^i only if it has at least b_k amounts of available bandwidth. For simplicity, we refer to the edges that are derived from switches in V_S as *node-derived edges*. Denote by $E_{k,node}^i$ the set of the node-derived edges. Clearly, $E_{k,node}^i = \{(v', v'') \mid \forall v \in V_S^i\}$.

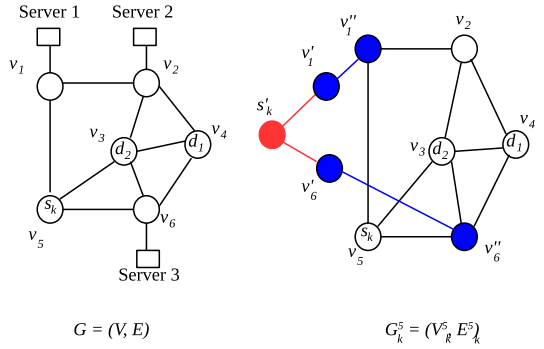


Fig. 5. An example of the auxiliary graph $G_k^5 = (V_k^5, E_k^5; w)$ constructed from an SDN $G = (V, E)$ with $V_S^5(k) = \{v_1, v_6\}$ for an incoming NFV-enabled multicast request r_k that contains a source $s_k = v_5$ and a destination set $D_k = \{v_3, v_4\}$, assuming that $M = 2$ and $V_S(k) = \{v_1, v_2, v_6\}$. There are $\binom{|V_S|}{M} = 3 \cdot 2 = 6$ auxiliary graphs derived from G with each corresponding a different combination of servers in $V_S(k)$, i.e., $V_S^1 = \{v_1\}$, $V_S^2 = \{v_2\}$, $V_S^3 = \{v_6\}$, $V_S^4 = \{v_1, v_2\}$, $V_S^5 = \{v_1, v_6\}$, and $V_S^6 = \{v_2, v_6\}$.

Having constructed the auxiliary graph $G_k^i = (V_k^i, E_k^i; w)$, we find an approximate Steiner tree T_k^i in G_k^i by applying the approximation algorithm due to Kou *et al.* [20]. We use the same process to find an approximate Steiner tree T_k^i for each value of i with $1 \leq i \leq \binom{|V_S(k)|}{M}$, i.e., different combinations of servers. Denote by $T_k^{\min}$ a pseudo-multicast tree with the minimum weight, i.e., $\sum_{e \in T_k^{\min}} w(e) \leq \sum_{e \in T_k^i} w(e)$ for all i with $1 \leq i \leq \binom{|V_S(k)|}{M}$.

To decide whether request r_k should be admitted, we here adopt a novel *admission control policy* that jointly considers both edge and node resource consumption costs, as the admission of r_k will significantly impact the admissions of future requests whose arrivals are not known in advance. The rationale behind adopting an admission control policy is to control the impact of admissions of requests with large resource demands on the admissions of future requests. Specifically, if a request with a very large resource demand is admitted, the future requests may be rejected when it occupies the resource for a long time. This unfortunately will heavily reduce the network throughput. The admission control policy thus described as follows.

A multicast request r_k will be admitted if it meets conditions: (i) $\sum_{e \in T_k^{\min} \cap E_{k,node}^{\min}} w(e) < \sigma_v$; and (ii) $\sum_{e \in T_k^{\min} \cap (E_k^{\min} \setminus E_{k,node}^{\min})} w(e) < \sigma_e$, where σ_e and σ_v are the pre-defined admission control thresholds for both bandwidth and computing usage costs of $T_k^{\min}$, respectively.

If $T_k^{\min}$ does exist, a feasible solution to the original problem will be derived from $T_k^{\min}$, by replacing its each edge (s_k', v') with the corresponding shortest path $p_{s_k,v}$ in G and implementing the service chain SC_k on the servers in $T_k^{\min}$. Note that no actual resource allocation is performed when finding T_k^i, and the resources are allocated to implement request r_k only if T_k^i meets the admission control conditions; otherwise, request r_k will be rejected.

The detailed description of the algorithm is given in `Algorithm 2`.

Algorithm 2 `Online_Heu`

Input: An SDN $G = (V, E)$, a set V_S of switches with each having server attached to it, the bandwidth resource capacity B_e of each link $e \in E$ and computing resource capacity C_v of each $v \in V_S$, a sequence of NFV-enabled multicast requests r_k that arrive at the network one by one, and values for the pre-defined admission control thresholds σ_e and σ_v.

Output: Admit or reject each incoming multicast request r_k. If admitted, a pseudo-multicast tree for the request will be returned.

$\mathcal{G} \leftarrow \emptyset$; /* a set of pseudo-multicast trees */
for each multicast request r_k **do**
  $cost_k \leftarrow \infty$; $T_k^{\min} \leftarrow \emptyset$; /* the cost of the pseudo-multicast tree */
  Find the shortest paths in $G_k = (V_k, E_k; w)$ from s_k to all nodes in V_S, and let $p_{s_k,v}$ be the found path from s_k to a node $v \in V_S$;
  for $i \leftarrow 1$ to $\binom{|V_S|}{M}$ **do**
    Construct an auxiliary undirected graph $G_k^i = (V_k^i, E_k^i; w)$ as shown in Fig. 5;
    Find an approximate Steiner tree T_k^i in G_k^i rooted at s_k' and spanning the nodes in D_k, by invoking the approximation algorithm due to Kou *et al.* [20];
    if $c(T_k^i) < cost_k$ **then**
      $cost_k \leftarrow c(T_k^i)$;
      $T_k^{\min} \leftarrow T_k^i$; /* a candidate solution */
  if $T_k^{\min}$ does not exist **then**
    Reject multicast request r_k;
  else
    if $(\sum_{e \in T_k^{\min} \cap E_{k,node}^{\min}} w(e) < \sigma_v)$ and $(\sum_{e \in T_k^{\min} \cap (E_k^{\min} \setminus E_{k,node}^{\min})} w(e) < \sigma_e)$ **then**
      An undirected graph G_k'' is derived from $T_k^{\min}$ by replacing each edge (s_k', v') in $T_k^{\min}$ with its corresponding shortest path $p_{s_k,v}$ in G;
      $\mathcal{G} \leftarrow \mathcal{G} \cup \{G_k''\}$;
    else
      Reject multicast request r_k;
return $\mathcal{G}$.

Notice that to ensure that the solution delivered by `Algorithm 2` meets the network bandwidth capacity constraint on each link $e \in E$, link e is contained in E_k^i only if its residual bandwidth is no less than 2 b_k, as shown in Lemma 2.

We finally analyze the performance of `Algorithm 2` by stating it in Theorem 2.

Theorem 2: Given an SDN $G = (V, E)$ with the computing capacity C_v for the server attached to each switch $v \in V_S$ and the bandwidth capacity B_e for each link $e \in E$, and a sequence of multicast requests with the kth multicast request being represented by $r_k = (s_k, D_k; b_k, SC_k)$, `Algorithm 2` deliver a feasible solution for the online NFV-enabled multicasting problem if the request sequence consists of k NFV-enabled multicast requests, and takes $O(k|V|^3|V_S|^M)$ time, where M is an integer constant which is the number of servers for the request service chain processing.

See the Appendix for the detailed proof.

C. Online Algorithm With $M = 1$

We now propose an online algorithm with a provable competitive ratio for a special online NFV-enabled multicasting problem where only a single server will be used to implement the service chain of each multicast request, i.e., $M = 1$. The basic idea behind the algorithm is to determine whether every incoming NFV-enabled multicast request r_k will be admitted or rejected, depending on *a given admission control policy*. If r_k is admissible, the algorithm requires to jointly find a server with sufficient computing resource to implement the service chain SC_k and a pseudo-multicast tree for r_k, such that the cost of implementing the request is minimized, subject to the resource capacity constraints on the network.

To find a pseudo-multicast tree for request r_k, we have an important observation: one of the $|V_S|$ servers must be contained in any pseudo-multicast tree for each request r_k, *the pseudo-multicast tree must include the server as one of the destination nodes of* r_k. Thus, we can find a Steiner tree in $G_k = (V_k, E_k; w)$ for request r_k with source s_k and the destination set $D_k \cup \{v\}$, where $v \in V_S$ has sufficient computing resource. Notice that $G_k(V_k, E_k; w)$ is an undirected graph that is identical to $G(V, E)$, i.e., $V_k = V$ and $E_k = E$. The weight $w_e(k)$ of each edge $e \in E_k$ is assigned a normalized cost of the cost defined by Eq. (3), that is, $w_e(k) = c_e(k)/B_e$, while the weight $w_v(k)$ of each node $v \in V_S$ for r_k is the normalized cost of the cost defined by Eq. (2), i.e, $w_e(k) = c_v(k)/C_v$.

Let T be an approximate Steiner tree in G_k rooted at s_k and spanning the terminals in D_k by the approximation algorithm due to Kou *et al.* [20], where G_k is the graph $G(V, E)$ that considered the first $k-1$ requests already, partial resources at its servers and links are occupied by some of the first $(k-1)$ requests at this moment. We build a pseudo-multicast tree for r_k. If server v is in any path in T from s_k to each destination, T is the pseudo-multicast tree, and its cost is no more than twice the optimal one [20]; otherwise, assume that the path between s_k and a destination node d does not contain any server v. Let u be the Lowest Common Ancestor (LCA) in T between nodes v_1 and v_2, i.e., $u = LCA(v_1, v_2)$. Then, when the packet from s_k is sent to server v for processing, the processed packet continues forwarding to all destinations in the subtree rooted at v; for the destination $d \in D_k$, the processed packet at node v is then sent back to node u, which then forwards toward destination d. Clearly, in the worst scenario, the processed packet will be sent back to the source s_k for multicasting.

Let T and T^* be the found approximate Steiner tree by the approximation algorithm in [20] and the optimal one. Denote by T_k and T_k^* the pseudo-multicast tree based on T and the optimal multicast tree, respectively. The sum of the weights of edges in the pseudo-multicast tree T_k based on T and server

v is

$$\begin{aligned} & w(T_k) + w_v(k) \\ & = w(T) + w(P^T_{v,u}) + w_v(k) \leq w(T) \\ & \quad + w(P^T_{v,s_k}) + w_v(k) \leq 2w(T) + w_v(k) \\ & \leq 2(w(T) + w_v(k)) \\ & \leq 4w(T^*) + 2w_v(k) \leq 4(w(T^*) + w_v(k)) \\ & \leq 4OPT_v, \end{aligned} \tag{5}$$

where $P^T_{x,y}$ is a path in T between node x and node y, and OPT_v is the optimal cost of the pseudo-multicast tree using server v as its service chain processing server. Thus, the optimal solution OPT for request r_k in G_k thus is $OPT = \min_{v \in V_S}\{OPT_v\}$.

We then adopt the following admission control policy to guide the admission of each multicast request r_k: (a) If $w_v(k) \geq \sigma_v$ for any $v \in V_S \cap T_k$, r_k will be rejected; and (b) if $\sum_{e \in T_k} w_e(k) \geq \sigma_e$, r_k will be rejected, where T_k is a pseudo-multicast tree delivered by an algorithm in G_k for r_k, $\sigma_v > 0$ and $\sigma_e > 0$ are admission control thresholds of computing and bandwidth respectively, and $\sigma_v = \sigma_e = |V| - 1$. The detailed online algorithm, referred to as `Online_CP`, is given in `Algorithm` 3.

We finally analyze the competitive ratio of `Algorithm` 3. Let $\mathcal{S}(k)$ and OPT be the sets of admitted multicast requests by `Algorithm` 3 and an optimal offline algorithm when multicast request r_k arrives. Let $\mathcal{R}(k)$ be the set of multicast requests that are rejected by `Algorithm` 3 while admitted by the optimal offline algorithm. Then the competitive ratio of `Algorithm` 3 is $\frac{|\mathcal{S}(k)|}{|\mathcal{S}(k) \cap OPT| + |\mathcal{R}(k)|} \geq \frac{|\mathcal{S}(k)|}{|\mathcal{R}(k)| + |\mathcal{S}(k)|}$, since $OPT = \mathcal{R}(k) \cup (OPT \cap \mathcal{S}(k)) \subseteq \mathcal{R}(k) \cup S(k)$. Specifically, the analysis of the competitive ratio of `Algorithm` 3 proceeds as follows. We first show the upper bound on the accumulative computing and bandwidth resources occupied by admitted requests in $\mathcal{S}(k)$ in Lemma 3. We then prove the lower bound on the accumulative computing and bandwidth resources occupied by rejected requests in $\mathcal{R}(k)$ in Lemma 4. We finally derive the competitive ratio by combining the results of the upper and lower bounds on admitted and rejected requests by the proposed on line algorithm in Theorem 3.

Lemma 3: When a multicast request r_k arrives, the cost sum of all servers in V_S is $\sum_{v \in V_S} c_v(k) \leq 2\mathbb{C}(k) \cdot \log \alpha \cdot (|V| - 1)$ and the cost sum of all links in E is $\sum_{e \in E} c_e(k) \leq 2\mathbb{B}(k) \cdot \log \beta \cdot (|V| - 1)$, respectively, provided that $b_{k'} \leq \frac{\min_{e \in E} B_e}{\log \beta}$ and $C_v(SC_{k'}) \leq \frac{\min_{v \in V_S} C_v}{\log \alpha}$ with $1 \leq k' \leq k$, where $\mathbb{B}(k)$ and $\mathbb{C}(k)$ are the accumulative amounts of bandwidth and computing resources being occupied by the admitted requests in $\mathcal{S}(k)$, respectively.

See the Appendix for the proof.

In the following, we show the lower bound on the cost of a multicast request in $\mathcal{R}(k)$ that is rejected by `Algorithm` 3 but admitted by an optimal offline algorithm.

Lemma 4: For each NFV-enabled multicast request $r_{k'} \in \mathcal{R}(k)$, if $\alpha = \beta = 2|V|$, we have

$$w(T'_{k'}) + w_{v'}(k') \geq \frac{|V| - 1}{4}, \tag{6}$$

Algorithm 3 `Online_CP`

Input: $G = (V, E)$, V_S, B_e for each $e \in E$, C_v for each $v \in V_S$, a sequence of multicast requests that arrive at the network one by one with each request $r_k = (s_k, D_k; b_k, SC_k)$, σ_e, and σ_v.

Output: The admission or rejection of each incoming NFV-enabled multicast request, if admitted, a pseudo-multicast tree for the request will be delivered.

$\mathcal{G} \leftarrow \emptyset$; /* the pseudo-multicast trees for the admitted requests */
for each incoming request r_k **do**
  $T_k \leftarrow \emptyset$; /* the pseudo-multicast tree for r_k if it is existent */
  $cost \leftarrow \infty$; /* the cost of the pseudo-multicast tree T_k for r_k */
  Construct an undirected graph $G_k = (V_k, E_k; w)$, by assigning a normalized weight $w_v(k)$ for each each node $v \in V_S$ and a normalized weight $w_e(k)$ for each link $e \in E$;
  for each $v \in V_S$ **do**
    if $w_v(k) < \sigma_v$ **then**
      Find an approximate Steiner tree T in G_k with the terminal set $\{s_k, v\} \cup D_k$ by the algorithm due to Kou *et al.* [20];
      if $\sum_{e \in T} w_e(k) < \sigma_e$ **then**
        Compute the lowest common ancestor $u = LCA(v, d_1, d_2, \ldots, d_{|D_k|})$ in T, where $LCA(x_1, x_2, \ldots, x_n) = LCA(LCA(x_1, x_2, \ldots, x_{n-1}), x_n)$;
        Calculate the cost $cost(k)$ of pseudo-multicast tree derived from T for r_k;
        $cost(k) \leftarrow c(T) + c_v(SC_k) + c(p_{v,u})$;
        if $cost(k) < cost$ **then**
          $T_k \leftarrow T$, $cost \leftarrow cost(k)$;
  if $T_k \neq \emptyset$, **then** Admit request r_k, $\mathcal{G} \leftarrow \mathcal{G} \cup \{< r_k, T_k, cost >\}$;
  else Reject request r_k;
return $\mathcal{G}$.

where $T'_{k'}$ is the Steiner tree found by the optimal offline algorithm to route the traffic of $r_{k'}$, and v' is the switch in $T'_{k'}$ whose server is selected to implement server chain $SC_{k'}$, assuming that both the following inequalities are met: $b_{k'} \leq \frac{\min_{e \in E} B_e}{\log \beta}$, and $C_v(SC_{k'}) \leq \frac{\min_{v \in V_S} C_v}{\log \alpha}$ *with* $1 \leq k' \leq k$.

See the proof in Appendix.

We finally analyze the competitive ratio of `Algorithm` 3.

Theorem 3: Given an SDN $G = (V, E)$ with computing capacity C_v of each server node $v \in V_S$ and bandwidth capacity B_e for each link $e \in E$, a sequence of NFV-enabled multicast requests with the kth multicast request r_k being represented by a quadruple $(s_k, D_k; b_k, SC_k)$, there is an online algorithm, `Algorithm` 3, with a competitive ratio of $O(\log |V|)$ for the online NFV-enabled multicasting problem if only one server is contained in the pseudo-multicast tree for the service chain implementation of the request, provided that $b_{k'} \leq \frac{\min_{e \in E} B_e}{\log \beta}$ and $C_v(SC_{k'}) \leq \frac{\min_{v \in V_S} C_v}{\log \alpha}$ with

$1 \leq k' \leq k$. The algorithm takes $O(k|V|^3)$ time if the request sequence contains k NFV-enabled multicast requests.

See the proof in Appendix.

VI. Performance Evaluation

In this section we evaluate the performance of the proposed algorithms through experimental simulation, using both synthetic networks generated by the GT-ITM tool and real networks, in a platform programmed using Java. We also investigate the impact of important parameters on the performance of the proposed algorithms.

A. Environment Settings

We consider SDNs consisting of from 50 to 250 nodes, where each network is generated using GT-ITM [10]. The number of servers in each network is set to 10% of the network size, and they are randomly co-located with switches in the network. We also use real network topologies, i.e., GÉANT [8] and an ISP network from [38]. There are nine servers for the GÉANT topology as set in [12] and the number of servers in the ISP networks is provided by [33]. The bandwidth capacity of each link varies from 1,000 Mbps to 10,000 Mbps [19], and the computing capacity of each server varies from 4,000 to 12,000 MHz [13]. Five types of network functions, i.e., Firewall, Proxy, NAT, IDS, and Load Balancing, are considered, and their computing demands are adopted from [12] and [29]. The source and destination nodes of each multicast request is randomly generated, *the ratio* of the maximum number D_{max} of destinations of a multicast request to the network size $|V|$ is randomly drawn in the range of [0.05, 0.2], and its bandwidth resource demand is randomly drawn in the range of [50, 200] *Mbps*. Using the hourly rate (price) of a general purpose m3.xlarge Amazon EC2 instance as a reference, the computing resource usage cost is set at \$0.25 per MHz. We assume that the cost of consuming a unit bandwidth resource is proportional to the length of a link, thus the cost of bandwidth resource usage of a network link varies between \$0.002 and \$0.005 per Mbps. We set both σ_e and σ_v at $|V| - 1$. The maximum number M of servers that can be used to implement the service chain of each multicast request is 3. The running time of each algorithm is obtained based on a machine with a 3.40GHz Intel i7 Quad-core CPU and 16 GiB RAM. Unless otherwise specified, these parameters will be adopted in the default setting.

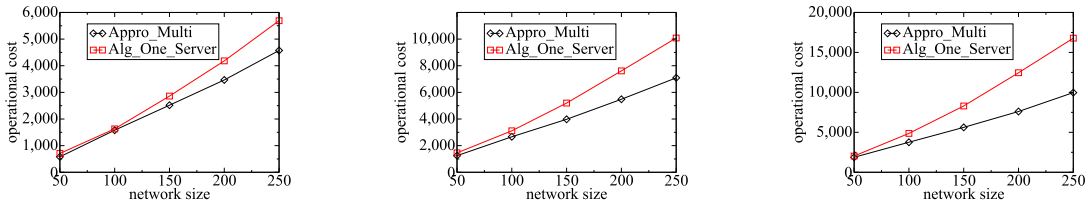
We evaluate the performance of algorithm `Appro_Multi` against the state-of-the-art – an algorithm in [42], referred to as algorithm `Alg_One_Server`, which only uses a single server to implement service chain SC_k of each multicast request r_k. Namely, it first routes the traffic of r_k to a server, and then finds a Minimum Spanning Tree (MST) of a complete graph G_c containing the destinations of r_k, where the edge between two destinations in G_c represents the shortest path between the two nodes in the original network. It then expands the MST into its corresponding subgraph in the original network. It finally selects the combination of server and subgraph with the minimum cost.

For the online algorithms, we study the performance of algorithms `Online_Heu` and `Online_CP` against those of a baseline heuristic `SP`. For multicast request r_k, algorithm `SP` first removes links and nodes that do not have enough available resources to admit r_k, and then assigns each link and each switch node in V_S with the same weight. For each candidate server in V_S, it then finds a shortest path from s_k to v and a single-source shortest path tree rooted at v and spanning all destinations of r_k. It finally uses a pseudo-multicast tree with the minimum cost for r_k. We also compare the performance of algorithms `Online_Heu` and `Online_CP` against their counterparts without the admission control policy, which are denoted by algorithms `Online_Heu_NoAd` and `Online_CP_NoAd`, respectively. Specifically, in algorithms `Online_Heu_NoAd` and `Online_CP_NoAd` the only condition that can reject a multicast request is the lack of available resources to fulfil the resource demands of the request. In the following experiments, we use the *operational cost* to represent the total cost of all admitted requests.

B. Performance Evaluation of Approximation Algorithms

We first evaluate the performance of algorithm `Appro_Multi` against that of algorithm `Alg_One_Server` by varying the network size from 50 to 250 and the ratio of the maximum number D_{max} of destinations of each request to the network size $|V|$ from 0.05 to 0.2. The operational cost and running time curves delivered by algorithms `Appro_Multi` and `Alg_One_Server` are drawn in Fig. 6, where the operational costs and running times are the average of admitting 1,000 NFV-enabled multicast requests. Specifically, we can see from Fig. 6 (a) that the operational cost by algorithm `Appro_Multi` is around 80% of that of algorithm `Alg_One_Server`. The reason is that algorithm `Appro_Multi` may use multiple servers that are close to the destinations of the request to implement the service chain of the request, which can significantly reduce the cost of bandwidth resource usage. Furthermore, it can be seen from the figure that the performance gap between the two algorithms becomes larger and larger, with the increase on the network size. The rationale behind is that algorithm `Appro_Multi` has more chances to select a set of servers that are closer to the destinations of each request, considering that more servers in larger networks are to be chosen. The similar performance behavior can be observed from Figs. 6(b) and (c). Furthermore, it can be seen from Figures 6 (d) - (f) that approximation algorithm `Appro_Multi` takes a slightly more time than that of algorithm `Alg_One_Server`, as different combinations of servers in V_S are to be considered.

We then investigate the performance of approximation algorithms `Appro_Multi` and `Alg_One_Server` in real networks GÉANT and AS1755, by varying $\frac{D_{max}}{|V|}$ from 0.05 to 0.2. Fig. 7 shows that the operational costs and running times of both algorithms. It can be seen that the operational cost delivered by algorithm `Appro_Multi` is much lower than that by algorithm `Alg_One_Server` while taking a slightly more running time. For example, the operational cost



(a) Operational costs with $D_{max}/|V| = 0.05$

(b) Operational costs with $D_{max}/|V| = 0.1$

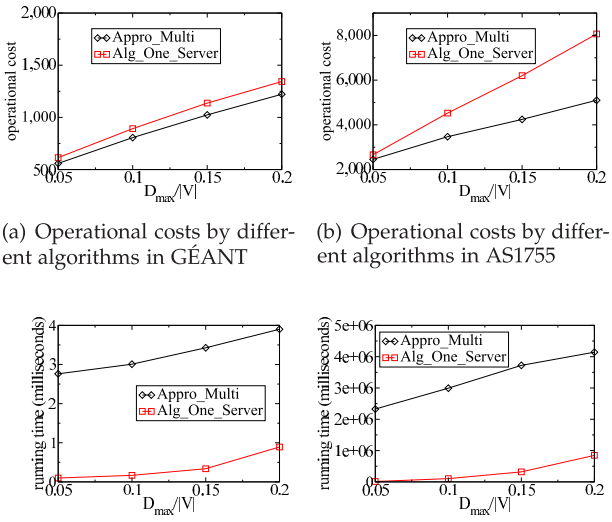
(c) Operational costs with $D_{max}/|V| = 0.2$

(d) Running times with $D_{max}/|V| = 0.05$

(e) Running times with $D_{max}/|V| = 0.1$

(f) Running times with $D_{max}/|V| = 0.2$

Fig. 6. The performance of algorithms `Appro_Multi` and `Alg_One_Server` with different ratios of $D_{max}/|V|$.



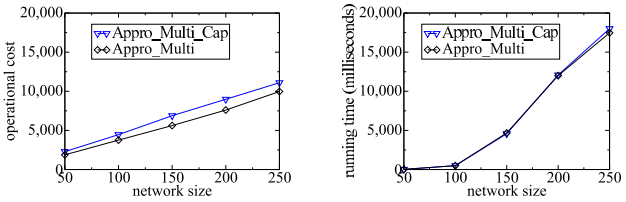
(a) Operational costs by different algorithms in GÉANT

(b) Operational costs by different algorithms in AS1755

(c) Running times of different algorithms in GÉANT

(d) Running times of different algorithms in AS1755

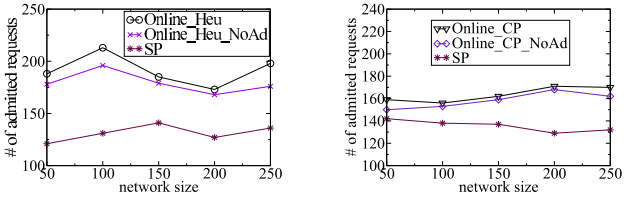
Fig. 7. The performance of different algorithms in networks of GÉANT and AS1755 with $M = 3$.



(a) The operational cost

(b) The running time

Fig. 8. The performance of algorithms `Appro_Multi_Cap`.



(a) The number of admitted requests

(b) The number of admitted requests

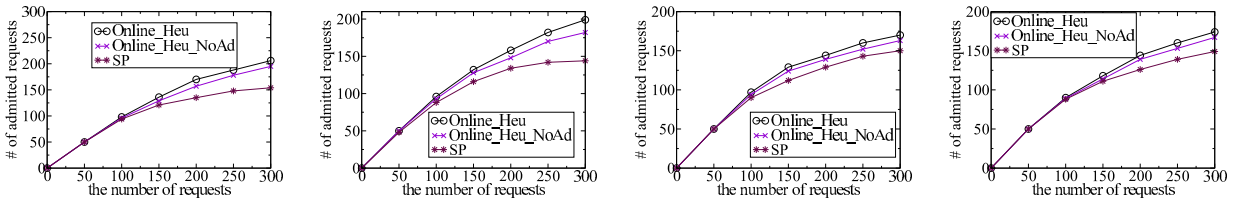
Fig. 9. The performance of online algorithms `Online_Heu`, `Online_CP`, and `SP` by varying the network sizes from 50 to 250.

by algorithm `Appro_Multi` in network AS1755 is around 30% lower than that of algorithm `Alg_One_Server` when $\frac{D_{max}}{|V|} = 0.15$ as shown in Fig. 7 (b).

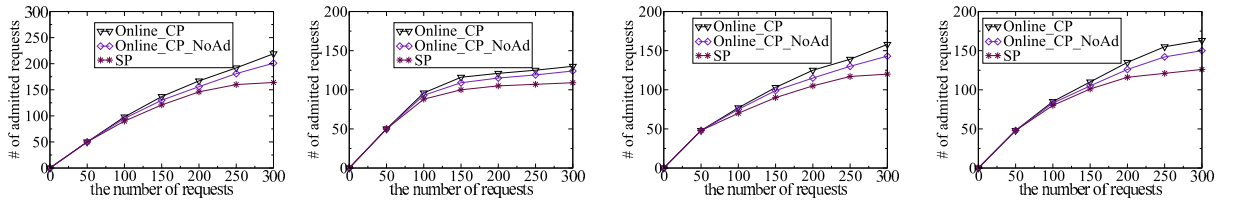
The rest is to evaluate the performance of approximation algorithm `Appro_Multi_Cap`, by setting $\frac{D_{max}}{|V|}$ at 0.2. Notice that algorithm `Alg_One_Server` is not considered as a benchmark as it does not consider SDN resource capacity constraints. From Figure 8(a), we can see that the operational cost of algorithm `Appro_Multi_Cap` is larger than that of algorithm `Appro_Multi`. The reason is that the algorithm `Appro_Multi_Cap` excludes the servers and links without enough available resources from the consideration for an incoming multicast request, this may reduce the number of combinations of servers that can be explored to implement request.

C. Performance Evaluation of Online Algorithms

We then evaluate the proposed online algorithms `Online_Heu` and `Online_CP` against algorithms `SP`, `Online_Heu_NoAd`, and `Online_CP_NoAd`, by varying the network size from 50 to 250 for a monitoring period that consists of 300 multicast requests. The numbers of admitted requests by algorithms `Online_Heu` and `Online_CP` are shown in Fig. 9. Specifically, we can see from Fig. 9 (a) that online algorithm `Online_Heu` admits at least twice numbers of requests admitted by algorithm `SP`. Although

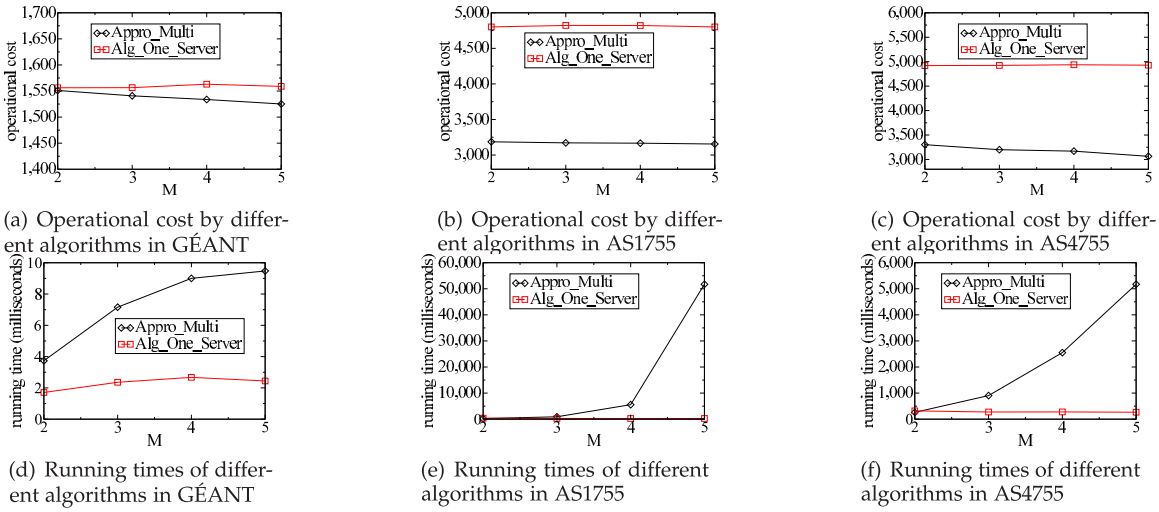


(a) The number of admitted requests by different algorithms in GT-ITM generated networks (b) The number of admitted requests by different algorithms in GÉANT (c) The number of admitted requests by different algorithms in AS1755 (d) The number of admitted requests by different algorithms in AS4755



(e) The number of admitted requests by different algorithms in GT-ITM generated networks (f) The number of admitted requests by different algorithms in GÉANT (g) The number of admitted requests by different algorithms in AS1755 (h) The number of admitted requests by different algorithms in AS4755

Fig. 10. The performance of algorithms `Online_Heu`, `Online_CP`, and `SP` by varying the number of requests in a monitoring period from 50 to 300 while fixing the network size at 100.



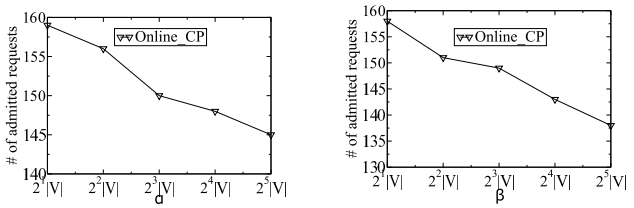
(a) Operational cost by different algorithms in GÉANT (b) Operational cost by different algorithms in AS1755 (c) Operational cost by different algorithms in AS4755

(d) Running times of different algorithms in GÉANT (e) Running times of different algorithms in AS1755 (f) Running times of different algorithms in AS4755

Fig. 11. The performance of algorithms `Appro_Multi` and `Alg_One_Server` in networks of GÉANT, AS1755, and AS4755.

the total resource capacity of each network keeps increasing with the increase on its size, the number of requests admitted by each mentioned algorithm on it does not monotonically increase. This is because it is very likely that the locations of the destinations of each multicast request in the network are far away from each other, thereby increasing the rejection likelihood of the request, due to more bandwidth resource consumptions. In addition, we can see that algorithm `Online_Heu` admits more requests than that by algorithm `Online_Heu_NoAd` without the admission control policy. The rationale behind is that the admission control policy rejects requests with high resource demands to preserve the resourced for future requests with low resource demands. Similar performance results delivered by online algorithm `Online_CP` can be observed in Fig. 9 (b).

We now evaluate the performance of online algorithms `Online_Heu` and `Online_CP` against online algorithms `SP`, `Online_Heu_NoAd`, and `Online_CP_NoAd`, by varying the number of requests from 50 to 300, in a GT-ITM generated network with size 100, and real networks GÉANT, AS1755, and AS4755, respectively. It can be seen from Fig. 10 (a) that algorithms `Online_Heu`, `SP`, `Online_Heu_NoAd` can admit almost all requests if the number of requests is no greater 100. Otherwise, algorithm `Online_Heu` admits more requests than those by algorithms `SP` and `Online_Heu_NoAd`. Also, the performance gap between algorithms `Online_Heu` and `SP` increases with the growth of the number of requests. The reason is that online algorithm `Online_Heu` considers the workload (or the utilization) of each resource by assigning each resource



(a) The number of admitted requests with $\beta = 2^1|V|$

(b) The number of admitted requests with $\alpha = 2^1|V|$

Fig. 12. The impact of parameters α and β on the performance of algorithms `Online_CP` by varying α or β while fixing the other at $2^1|V|$.

an exponential cost, while algorithm `SP` does not take the resource workload and assigns the same weight to the same amount of resource at different nodes and links, which may lead to the excessive usage of a heavily-loaded resource. From Fig. 10 (e), we can also see the similar performance of algorithm `Online_CP` against algorithms `Online_CP_NoAd` and `SP` in GT-ITM generated topologies. Regarding their performance in networks GÉANT, AS1755, and AS4755, Figures 10 (b) - (d), and Figures 10 (f) - (h) illustrates similar trends of algorithms `Online_Heu` and `Online_CP`, respectively.

D. Impact of Parameters on the Performance of Different Algorithms

In the following we study the impact of the number of servers M for implementing the service chain of each multicast on the performance of approximation algorithm `Appro_Multi` against that of approximation algorithm `Alg_One_Server` in real networks GÉANT, AS1755, and AS4755 [8], [38], by varying the value of M from 2 to 5. The results are depicted in Fig. 11. From Figures 11 (a), (b), and (c), it can be seen that the operational cost delivered by algorithm `Appro_Multi` is consistently lower than that by algorithm `Alg_One_Server` for different networks with different Ms. For example, as shown in Fig. 11 (c), the operational cost of approximation algorithm `Appro_Multi` in network AS4755 is at least 30% less than that of algorithm `Alg_One_Server`. Also, the operational cost of algorithm `Appro_Multi` decreases while its running time increases, with the increase of M, due to more server combinations to be explored.

We finally investigate the impact of α and β on the performance of online algorithm `Online_CP`, by varying α and β from $2^1|V|$ to $2^5|V|$ while setting $\sigma_v = \sigma_e = |V| - 1$. Fig. 12 plots the performance curves of algorithm `Online_CP`. It can be seen from Fig. 12 (a) that when β is fixed, the larger the value of α, the less number of requests can be admitted by algorithm `Online_CP`. Also, when α is fixed, similar performance can be found in Fig. 12 (b).

VII. Conclusion

In this paper we studied NFV-enabled multicasting in an SDN. We first devised an approximation algorithm with a constant approximation ratio, assuming that the number of servers for implementing each service chain is no more than a constant $M \geq 1$, subject to the computing and bandwidth capacity constraints. We then studied dynamic admissions of NFV-enabled multicast requests without the knowledge of future arrivals, with the objective to maximize the network throughput, for which we proposed an efficient heuristic and an online algorithm with a provable competitive ratio if $M = 1$. We finally evaluated the performance of the proposed algorithms by experimental simulations. Simulation results demonstrate that the proposed algorithms outperform the other heuristics.

Appendix

Proof of Lemma 1

Proof: We here show that the solution delivered by `Algorithm 1` is feasible. Each path p in T_k from s'_k to one destination $d \in D_k$ corresponds to a path in G from s_k to d. This is evidenced by the fact that any path in subgraph H_k^i of G_k^i, starting from s'_k must use one of its incident edges in E_k^i, and the another endpoint v of the edge must be a server in $V_S^i \subseteq V_S$, following the construction of G_k^i. This implies that each path in T_k from s_k to any destination $d \in D_k$ must use one of the servers in V_S^i. And T_k includes all nodes in D_k. Thus, the tree obtained is a feasible pseudo-multicast tree for multicast request r_k.

Proof of Lemma 2

Proof: Following the construction of G_k^i, each approximate Steiner tree T_k^i in G_k^i will include at least one edge between the virtual source node s'_k and a switch node attached to server $v_j \in V_S^i$, which is replaced by the corresponding shortest path p_{s_k,v_j} in G between s_k and v_j for the admission of request r_k. However, there may exist an edge $e \in E$ which is in both $T_k^i \setminus \{(s'_k, v_j) \mid v_j \in V_S^i\}$ and p_{s_k,v_j}. Whenever this happens, edge e will be used twice in the data traffic routing of request r_k: one is when e is in path $p_{s_k,u}$ between the source s_k and a server $u \in V_S^i$, where the edge $(s'_k, u) \in \{(s'_k, v_j) \mid v_j \in V_S^i\} \subset E_k^i$ derived from server u is included in $T_k^{\min}$; and another is that edge e is in a path between server u and at least one destination in D_k. Thus, each link $e \in E_k^i$ must have at least the amount $2b_k$ of residual bandwidth for the admission of request r_k, to ensure that the data traffic of request r_k can be routed from its source s_k to each of its destinations.

Proof of Theorem 1

Proof: We first analyze the approximation ratio of `Algorithm 1`. Let G_T^* be the optimal pseudo-multicast tree for the NFV-enabled multicast request r_k in G. If G_T^* is not a multicast tree, there is a corresponding tree T' with the identical cost as G_T^*, following the reduction in subsection III-B; otherwise, G_T itself is a multicast tree. From now on, we denote by T^* either the optimal multicast tree G_T^* or its corresponding cost-identical tree T'. We assume that there are l servers in T^* with each implementing SC_k with $1 \leq l \leq M$. Without loss of generality, we assume that these l nodes are $v_1, v_2, \ldots, v_l$, respectively. Clearly, it can be shown that none of pairs of these nodes in T^* has the ancestor and descendant relationship in terms of a node being used as

a server, otherwise the node in V_S will be treated as a regular switch node without the use of its server. Each subtree $T^*_{v_i}$ of T^* rooted at v_i contains some destinations, and all of the l subtrees will contain all the destinations in D_k, following its definition. We construct another tree $T_c^* = (V', E')$ which is derived from T^* by compressing the path in T^* from s_k to each node v_i as follows. We replace the source node s_k by a node s'_k and the path in T^* from s_k to v_i by an edge (s'_k, v_i), and assign the edge a weight that is the sum of all edge costs in the path plus the cost of using server v_i. In the worst case, each of such compressions can increase the cost of the optimal tree, and there are in total l compressions. Furthermore, for each compression, if the cost sum of all edges from the source s_k to each v_i dominates the cost of the tree, each of such compressions can increase the total cost by a value that equals the cost of T^*. We thus claim that the cost of tree T_c^* is no greater than l times the cost of tree T^*, i.e., $c(T_c^*) \leq l \cdot c(T^*)$.

It can be seen that there is a multicast tree T_k^i in G_k^i rooted at source s'_k and spanning all destinations in D_k, which has the same topological structure as T_c^*, however, it has a lower cost compared with that of T_c^*, i.e., $c(T_k^i) \leq c(T_c^*)$. This is because the weight of each edge in G_k^i between s'_k and v_i is the length of the shortest path in G between the two nodes plus the cost of using server v_i, while the corresponding edge weight in T_c^* is the sum of all edge weights in the path in T^* between s_k and v_i plus the cost of using server v_i.

Let $T_k^{OPT,i}$ be an optimal multicast tree in G_k^i rooted at s'_k and spanning all destinations in D_k and each path in the tree from s'_k to a destination goes through one of the servers in V'_S. Then, $c(T_k^{OPT,i}) \leq c(T_k^i)$ as T_k^i is one of the multicast trees for multicast request r_k. Let $T_k^{app,i}$ be an approximate multicast tree in G_k^i for multicast request r_k, by the approximation algorithm with an approximation ratio of 2 due to Kou *et al.* [20]. We then have $c(T_k^{app,i}) \leq 2\ c(T_k^{OPT,i})$. Since $c(T_k^i) \leq c(T_c^*)$ and $c(T_c^*) \leq l \cdot c(T^*)$, we have $c(T_k^{app,i}) \leq 2\ c(T_k^{OPT,i}) \leq 2\ c(T_k^i) \leq 2\ c(T_c^*) \leq 2 \cdot l \cdot c(T^*)$. Since a pseudo-multicast tree T_k with the minimum cost from the $\binom{|V_S|}{M}$ auxiliary undirected graphs G_k^i for all i with $1 \leq i \leq \binom{|V_S|}{M}$ will be found and the value of l is within $[1, M]$, the cost of the pseudo-multicast tree T_k for r_k is no greater than $2M \cdot c(T^*)$.

We finally analyze the time complexity of `Algorithm 1` as follows. The algorithm proceeds iteratively. Within each iteration, it first constructs an auxiliary graph G_k^i, and then finds an approximate Steiner tree $T_k^{app,i}$ in G_k^i for each multicast request r_k. It takes $O(|E|+|V|\log|V|)$ time to find a single-source shortest path tree in G_k by Dijkstra's algorithm, while it takes $O(|V_k^i|^3) = O(|V|^3)$ time to find an approximate Steiner tree $T_k^{app,i}$ [20]. There are $O(\binom{|V_S|}{M})$ $(= O(|V_S|^M))$ iterations. The algorithm thus takes $O(|V|^3 \cdot |V_S|^M)$ time. For example, if $|V_S| = O(\log|V|)$ and $M = 3$, then the time complexity of `Algorithm 1` is $O(|V|^3 \cdot \log^3|V|)$.

Proof of Theorem 2

Proof: We first show that the solution delivered by `Algorithm 2` is feasible, meeting the computing and bandwidth resource demands of multicast request r_k. This can be evidenced by the construction of G_k^i for all i with $1 \leq i \leq \binom{|V_S(k)|}{M}$. Specifically, for meeting the computing resource demands of request r_k, only the switch nodes whose attached servers with sufficient available computing resource will be included in V_S^i. For the bandwidth resource demand b_k of multicast request r_k, an amount b_k of bandwidth resource is reserved on each link in the shortest path $p_{s_k,v}$ in G from s_k to each $v \in V_S$ for the transfer of the original traffic of r_k, and an equal amount is allocated to transfer the processed traffic by each selected server to destinations in D_k. By Lemma 2, it can be seen that the bandwidth capacity constraint on each link is met. Notice that if a switch $v' \in V_S$ is not selected in the solution, the preserved resource b_k in each edge of path $p_{s_k,v}$ will be released to the resource pool. Also, following the construction of G_k^i, it can be seen that each routing path from s'_k to a destination node u must pass through a node u in the path and u is the neighbor of s'_k, $u \in V_S^i$ is a server with sufficient computing resource for SC_k, i.e., the available computing resource at u is at least $C_u(SC_k)$. Thus, the solution is a feasible solution.

We then analyze the running time of the proposed online algorithm. Clearly, the construction of auxiliary graph G_k^i takes $O(|V|\log|V|+|E|)$ time, as its edge assignment needs finding a single-source shortest path tree in G which takes $O(|V|\log|V|+|E|)$ time. Finding an approximate Steiner tree in G_k^i takes $O(|V_k^i|^3) = O(|V|^3)$ time, by the approximation algorithm in [20]. There are $O(\binom{|V_S(k)|}{M})$ $(= O(|V_S(k)|^M) = O(|V_S|^M))$ different values of i, and transforming a minimum-cost approximate Steiner tree $T_k^{\min}$ to a feasible solution of the problem takes $O(|V_S| + |V|)$ time. Therefore it takes $O((|V|\log|V|+|E|+|V|^3) \cdot |V_S|^M) = O(|V|^3 \cdot |V_S|^M)$ time for each incoming multicast request. The algorithm thus takes $O(k|V|^3|V_S|^M)$ time if there are k NFV-enabled multicast requests in the request sequence. The theorem follows.

Proof of Lemma 3

Proof: Consider any $k' \in \mathcal{S}(k)$ admitted by the online algorithm, its data traffic is first routed to a server $v \in V_S$ that hosts its service chain SC_k via path $p_{s_k,v}$ in G, and then to each of its destinations in D_k through a subtree T_v with root v. The costs of computing resource usage of different servers in V_S are different, we thus have

$$\begin{aligned} c_v(k'+1) - c_v(k') &\leq C_v(\alpha^{1-\frac{C_v(k'+1)}{C_v}} - \alpha^{1-\frac{C_v(k')}{C_v}}) \\ &= C_v\alpha^{1-\frac{C_v(k')}{C_v}}(\alpha^{\frac{C_v(SC_{k'})}{C_v}} - 1) \\ &= C_v\alpha^{1-\frac{C_v(k')}{C_v}}(2^{\frac{C_v(SC_{k'})\log\alpha}{C_v}} - 1) \quad (7) \\ &\leq C_v\alpha^{1-\frac{C_v(k')}{C_v}}(\frac{C_v(SC_{k'})\log\alpha}{C_v}) \quad (8) \\ &= \alpha^{1-\frac{C_v(k')}{C_v}} C_v(SC_{k'}) \cdot \log\alpha. \quad (9) \end{aligned}$$

Notice that the derivation from Eq. (7) to inequality (8) is due to that $2^x - 1 \leq x$ for $0 \leq x \leq 1$. Similarly, if edge $e \in E$ is in a pseudo-multicast tree for multicast request r_k, we have

$$c_e(k'+1) - c_e(k') \leq \beta^{1-\frac{B_e(k')}{B_e}} b_{k'} \cdot \log\beta. \quad (10)$$

Notice that $c_v(0) = 0$ for all $v \in V_S$. If an NFV-enabled multicast request $r_{k'}$ is admitted, this means that

$$w_v(k') = \alpha^{1-\frac{C_v(k')}{C_v}} - 1 \leq \sigma_v = |V| - 1, \quad (11)$$

We now calculate the cost sum of all edges and the server attached to $v \in V_S$ when admitting the multicast request r_k. Notice that if an edge in E is not included in T_k for r_k, its cost does not change after the admission of r_k. So does the cost of a server that is not used for implementing SC_k of r_k. Assuming that v is chosen for $r_{k'}$, we have

$$\begin{aligned}
&\sum_{v\in V_S} c_v(k'+1) - c_v(k') \\
&\quad = c_v(k'+1) - c_v(k') \\
&\quad \le (\alpha^{1-\frac{C_v(k')}{C_v}})C_v(SC_{k'})\log\alpha, \quad \text{by inequality (9)} \\
&\quad = (\alpha^{1-\frac{C_v(k')}{C_v}}-1)C_v(SC_{k'})\log\alpha + C_v(SC_{k'})\log\alpha \\
&\quad \le (\alpha^{1-\frac{C_v(k')}{C_v}}-1)C_v(SC_{k'})\log\alpha \\
&\qquad + (|V|-1)C_v(SC_{k'})\log\alpha, \quad \text{since} |V|>1 \\
&\quad \le 2C_v(SC_{k'})\log\alpha(|V|-1) \quad \text{by inequality (11).}
\end{aligned} \tag{12}$$

Similarly we have

$$\sum_{e\in E} c_e(k'+1) - c_e(k') \le 2b_{k'}\log\beta(|V|-1). \tag{13}$$

We finally have

$$\begin{aligned}
\sum_{v\in V_S} c_v(k) &= \sum_{k'=1}^{k-1}\sum_{v\in V_S}(c_v(k'+1)-c_v(k')) \\
&\le \sum_{k'\in\mathcal{S}(k)} 2\ C_v(SC_{k'})\cdot\log\alpha\cdot(|V|-1) \\
&= 2\mathbb{C}(k)\cdot\log\alpha\cdot(|V|-1),
\end{aligned}$$

and similarly $\sum_{e\in E} c_e(k) \le 2\mathbb{B}(k)\cdot\log\beta\cdot(|V|-1)$.

Proof of Lemma 4

Proof: A multicast request $r_{k'}$ will be rejected by the proposed online algorithm, `Algorithm 3`, because of the following cases. Case 1, there is no sufficient computing resource for implementing the service chain of $r_{k'}$; Case 2. There is no sufficient bandwidth resource for routing the traffic of $r_{k'}$ to its destinations; Case 3. The weighted sum of edges in the pseudo-multicast tree for $r_{k'}$ is too high (Step 9), and/or the weight of the selected server attached to switch v to implement the service chain of $r_{k'}$ is too high (Step 7).

Case 1: Let v' be the switch whose attached server is selected to implement the service chain of request k'. As the request is rejected by `Algorithm 3` due to insufficient available computing resource, we have $C_{v'}(k') < C_v(k')$. Therefore, $1-\frac{C_{v'}(k')}{C_{v'}} \ge 1-\frac{C_v(k')}{C_v(SC_{k'})} \ge 1-\frac{1}{\log\alpha}$, since $C_v(SC_k) \le \frac{\min_{v\in V_S} C_v}{\log\alpha}$. We then have

$$\begin{aligned}
w(T'_{k'}) + w_{v'}(k') &\ge \alpha^{1-\frac{C_{v'}(k')}{C_{v'}}} - 1 > \alpha^{1-\frac{1}{\log\alpha}} - 1 \\
&= \frac{\alpha}{2} - 1 = |V| - 1.
\end{aligned} \tag{14}$$

Case 2: If request $r_{k'}$ is rejected in this case, there exists an edge $e' \in T'_{k'}$ that does not have enough bandwidth resource. This implies that for edge e', we have $B_{e'}(k') < b_{k'}$. Thus, we have $1-\frac{B_{e'}(k')}{B_{e'}} \le 1-\frac{b_{k'}}{B_{e'}} \le 1-\frac{1}{\log\beta}$, since $b_{k'} \le \frac{\min_{e\in E} B_e}{\log\beta}$. Similarly, we have

$$\begin{aligned}
&w(T'_{k'}) + w_{v'}(k') \\
&\quad \ge \sum_{e\in T'_{k'}} \beta^{1-\frac{B_e(k')}{B_e}} - 1 \\
&\quad \ge \beta^{1-\frac{B_{e'}(k')}{B_{e'}}} - 1 \ge \beta^{1-\frac{1}{\log\beta}} - 1 \ge \frac{\beta}{2} - 1 = |V| - 1.
\end{aligned} \tag{15}$$

Case 3: Let $T_{k'}$ be the pseudo-multicast tree by `Algorithm 3` for multicast request $r_{k'}$. According to inequality (5), we have $w(T_{k'}) + w_{v'}(k') \le 4(w(T^*_{k'}) + w_{v'}(k')) \le 4(w(T'_{k'}) + w_{v'}(k'))$, where $T^*_{k'}$ and v' are the optimal multicast tree and the server for $r_{k'}$ by the optimal solution. Since the proposed online algorithm rejected the request, we have $w(T_{k'}) > \sigma_e$ and/or $w_{v'}(k') > \sigma_v$. If $w(T_{k'}) > \sigma_e$, we have $w(T'_{k'}) + w_{v'}(k') \ge \frac{w(T_{k'})+w_{v'}(k')}{4} \ge \frac{w(T_{k'})}{4} \ge \frac{\sigma_e}{4} = \frac{|V|-1}{4}$, similarly, if $w_{v'}(k') > \sigma_v$, we have $w(T'_{k'}) + w_{v'}(k') \ge \frac{\sigma_v}{4} = \frac{|V|-1}{4}$. Summarizing the three cases, we have $w(T'_{k'}) + w_{v'}(k') \ge \frac{|V|-1}{4}$.

Proof of Theorem 3

Proof: The competitive ratio of `Algorithm 3` is analyzed as follows. Let $T^*_{k'}$ be the optimal multicast tree by the optimal offline algorithm for request $r_{k'} \in \mathcal{R}(k)$ and $v^* \in T^*_{k'}$ be the server for the service chain $SC_{k'}$ of $r_{k'}$.

$$\begin{aligned}
&\frac{|V|-1}{4}(|\mathcal{R}(k)|) \\
&\le \sum_{r_{k'}\in\mathcal{R}(k)} \frac{|V|-1}{4} \\
&\le \sum_{r_{k'}\in\mathcal{R}(k)}\sum_{e\in T^*_{k'}} (\beta^{1-\frac{B_e(k')}{B_e}}! - 1) + \alpha^{1-\frac{C_{v^*}(k')}{C_{v^*}}} - 1, \text{by Lemma 4} \\
&= \sum_{r_{k'}\in\mathcal{R}(k)} (\sum_{e\in T^*_{k'}} c_e(k')/B_e + c_{v^*}(k')/C_{v^*}), \\
&\le \sum_{r_{k'}\in\mathcal{R}(k)} (\sum_{e\in T^*_{k'}} c_e(k)/B_e + c_{v^*}(k)/C_{v^*}) \\
&\le \sum_{e\in T^*_{k'}} c_e(k) \sum_{r_{k'}\in\mathcal{R}(k)} 1/B_e + c_{v^*}(k) \sum_{r_{k'}\in\mathcal{R}(k)} 1/C_{v^*}, \\
&\le \sum_{e\in E} c_e(k) + \sum_{v\in V_S} c_v(k).
\end{aligned} \tag{16}$$

Following inequalities (16) and Lemma 3, we have

$$\begin{aligned}
&\frac{|V|-1}{4}(|\mathcal{R}(k)|) \\
&< \sum_{e\in E} c_e(k) + \sum_{v\in V_S} c_v(k) \\
&\le 2\mathbb{C}(k)\log\alpha(|V|-1) + 2\mathbb{B}(k)\log\beta(|V|-1) \\
&\le 2|\mathcal{S}(k)|C_{\max}\log\alpha(|V|-1) + 2|\mathcal{S}(k)|b_{\max}\log\beta(|V|-1) \\
&= 2|\mathcal{S}(k)|(|V|-1)(C_{\max}\log\alpha + b_{\max}\log\beta),
\end{aligned} \tag{17}$$

where $C_{\max} = \arg\max_k C_v(SC_k)$ and $b_{\max} = \arg\max_k b_k$, i.e., the maximum computing and bandwidth resource demands of all requests. Inequality (17) then can be rewritten as

$$|\mathcal{R}(k)|/|\mathcal{S}(k)| \le 8(C_{\max}\log\alpha + b_{\max}\log\beta). \tag{18}$$

The competitive ratio of `Algorithm` 3 thus is

$$\begin{aligned}
&\frac{|\mathcal{S}(k)|}{|OPT|} \\
&\geq \frac{|\mathcal{S}(k)|}{|\mathcal{R}(k) \cup \mathcal{S}(k)|} \\
&\geq \frac{|\mathcal{S}(k)|}{|\mathcal{R}(k)| + |\mathcal{S}(k)|} = \frac{1}{|\mathcal{R}(k)|/|\mathcal{S}(k)| + 1} \\
&\geq \frac{1}{1 + 8(C_{\max} \log \alpha + b_{\max} \log \beta)}, \quad \text{by inequality (18)} \\
&\geq \frac{1}{c' \log |V|}, \text{when } C_{\max} \text{ and } b_{\max} \text{ are given constants,}
\end{aligned}$$

where $c' > 0$ is a positive constant and $\alpha = \beta = 2|V|$. The competitive ratio of the competition ratio of `Algorithm` 3 thus is $O(\log |V|)$ when $\alpha = \beta = 2|V|$.

The rest is to analyze the time complexity of the proposed algorithm. The construction of G_k takes $O(|V_k| + |E_k|) = O(|V| + |E|)$ time, while finding an approximate multicast tree from G_k takes $O(|V_k|^3) = O(|V|^3)$ time, by the 2-approximation algorithm of Kou *et al.* [20]. `Algorithm` 3 thus takes $O(k|V|^3)$ time if there are k requests.

REFERENCES

[1] (2017). *Application Delivery Controller*. [Online]. Available: http://searchnetworking.techtarget.com/definition/Application-delivery-controller

[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] L. A. Barroso, J. Dean, and U. Holzle, "Web search for a planet: The Google cluster architecture," *IEEE Micro*, vol. 23, no. 2, pp. 22–28, Mar./Apr. 2003.

[4] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità, "An improved LP-based approximation for Steiner tree," in *Proc. ACM STOC*, 2010, pp. 1–10.

[5] Z. Cao, M. Kodialam, and T. V. Lakshman, "Traffic steering in software defined networks: Planning and online routing," in *Proc. ACM DCC*, 2014, pp. 65–70.

[6] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "On the effect of forwarding table size on SDN network utilization," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 1734–1742.

[7] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE INFOCOM*, 2015, pp. 1346–1354.

[8] *GÉANT*. Accessed: Dec. 2017. [Online]. Available: http://www.geant.net

[9] S. Ghemawat, H. Gobioff, and S. T. Leung, "The Google file system," in *Proc. SOSP*, 2003, pp. 29–43.

[10] Accessed: Oct. 2017. [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/

[11] L. Guo, J. Pang, and A. Walid, "Dynamic service function chaining in SDN-Enabled networks with middleboxes," in *Proc. IEEE ICNP*, Nov. 2016, pp. 1–10.

[12] A. Gushchin, A. Walid, and A. Tang, "Scalable routing in SDN-enabled networks with consolidated middleboxes," in *Proc. ACM HotMiddlebox*, 2015, pp. 55–60.

[13] Hewlett-Packard Development Company. (2015). *L.P. Servers for Enterprise BladeSystem, Rack & Tower and Hyperscale*. [Online]. Available: http://www8.hp.com/us/en/products/servers/

[14] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[15] L.-H. Huang, H.-J. Hung, C.-C. Lin, and D.-N. Yang. (May 2014). "Scalable steiner tree for multicast communications in software-defined networking." [Online]. Available: https://arxiv.org/abs/1404.3454

[16] M. Huang, W. Liang, Z. Xu, W. Xu, S. Guo, and Y. Xu, "Dynamic routing for network throughput maximization in software-defined networks," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.

[17] K. Kar, M. Kodialam, T. V. Lakshman, and L. Tassiulas, "Routing for network capacity maximization in energy-constrained ad-hoc networks," in *Proc. IEEE INFOCOM*, Mar./Apr. 2003, pp. 673–681.

[18] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[19] 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.

[20] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta Inf.*, vol. 15, no. 2, pp. 141–145, 1981.

[21] 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. IEEE INFOCOM*, Apr. 2016, pp. 1–9.

[22] Y. Li, L. T. X. Phan, and B. T. Loo, "Network functions virtualization with soft real-time guarantees," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.

[23] W. Liang, "Constructing minimum-energy broadcast trees in wireless ad hoc networks," *Proc. ACM Mobihoc*, 2002, pp. 112–122.

[24] W. Liang and X. Guo, "Online multicasting for network capacity maximization in energy-constrained ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 5, no. 9, pp. 1215–1227, Sep. 2006.

[25] W. Liang and Y. Liu, "Online data gathering for maximizing network lifetime in sensor networks," *IEEE Trans. Mobile Comput.*, vol. 6, no. 1, pp. 2–11, Jan. 2007.

[26] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," in *Proc. SIROCCO*, 2015, pp. 104–118.

[27] L. Mamatas, S. Clayman, and A. Galis, "A service-aware virtualized software-defined infrastructure," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 166–174, 2015.

[28] (2017). *MapReduce*. [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

[29] J. Martins *et al.*, "ClickOS and the art of network function virtualization," in *Proc. USENIX NSDI*, 2014, pp. 459–473.

[30] M. McBride, *Multicast in the Data Center Overview*, document draft-IETF-mboned-dc-deploy-00, 2013. [Online]. Available: https://tools.ietf.org/id/draft-ietf-mboned-dc-deploy-00.html#rfc.status

[31] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *Proc. IEEE CNSM*, Nov. 2014, pp. 418–423.

[32] S. Plotkin, "Competitive routing of virtual circuits in ATM networks," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 6, pp. 1128–1136, Aug. 1995.

[33] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," in *Proc. ACM SIGCOMM*, 2013, pp. 27–38.

[34] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Trans. Commun.*, vol. 64, no. 9, pp. 3746–3758, Sep. 2016.

[35] G. Robins and A. Zelikovsky, "Tighter bounds for graph Steiner tree approximation," *SIAM J. Discrete Math.*, vol. 19, no. 1, pp. 122–134, 2005.

[36] (2017). *Software-Defined Data Center*. [Online]. Available: https://en.wikipedia.org/wiki/ Software-defined_data_center

[37] (2017). *Software-Defined Data Center—In Depth*. [Online]. Available: https://www.vmware.com/solutions/software-defined-datacenter/in-depth.html

[38] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," in *Proc. ACM SIGCOMM*, 2002, pp. 133–145.

[39] Z. Xu, W. Liang, A. Galis, and Y. Ma, "Throughput maximization and resource optimization in NFV-enabled networks," in *Proc. IEEE ICC*, May 2017, pp. 1–7.

[40] Z. Xu, W. Liang, M. Huang, M. Jia, S. Guo, and A. Galis, "Approximation and online algorithms for NFV-enabled multicasting in SDNs," in *Proc. IEEE 37th Intl Conf Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 625–634.

[41] Y. Zhang, *et. al*, "StEERING: A software-defined networking for inline service chaining," in *Proc. IEEE ICNP*, Oct. 2013, pp. 1–10.

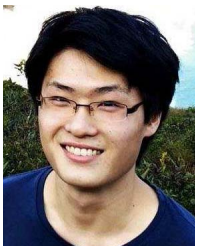
[42] S. Q. Zhang, Q. Zhang, H. Bannazadeh, and A. Leon-Garcia, "Network function virtualization enabled multicast routing on SDN," in *Proc. IEEE ICC*, Jun. 2015, pp. 5595–5601.

[43] S. Q. Zhang, Q. Zhang, H. Bannazadeh, and A. Leon-Garcia, "Routing algorithms for network function virtualization enabled multicast topology on SDN," *IEEE Trans. Netw. Service Manage.*, vol. 12, no. 4, pp. 580–594, Dec. 2015.

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 from 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, network function virtualization, software-defined networking, wireless sensor networks, routing protocol design for wireless networks, algorithmic game theory, and optimization problems.

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

Meitian Huang received the B.Sc. degree (Hons.) in computer science from 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.



Mike Jia received the B.Sc. degree in mathematics and computer science from Imperial College London, U.K., in 2013, and the B.Sc. degree (Hons.) in computer science from Australian National University in 2014, where he is currently pursuing the Ph.D. degree in computer science. His research interests include mobile cloud computing and software-defined networks.

Song Guo (M'02–SM'11) is currently a Full Professor with the Department of Computing, The Hong Kong Polytechnic University. His research interests are mainly in the areas of big data, cloud computing, mobile computing, and distributed systems, with over 450 papers published in major conferences and journals. He currently serves as the Director and a member of the Board of Governors of the IEEE ComSoc. His work was recognized by the 2016 Annual Best of Computing: Notable Books and Articles in Computing in ACM Computing Reviews. He is a recipient of the 2018 IEEE TCGCC Best Magazine Paper Award, the 2017 IEEE Systems Journal Annual Best Paper Award, and other six Best Paper Awards from the IEEE/ACM conferences. He also served as the General and Program Chair for numerous IEEE conferences. He was an Associate Editor of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. He is currently an Associate Editor of the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING, the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, and the IEEE NETWORK. He is an IEEE ComSoc Distinguished Lecturer.

Alex Galis is currently a Professor in networked and service systems with University College London. He has co-authored 10 research books and more that 250 publications in the future internet areas, such as system management, networks and services, networking clouds, and 5G virtualization and programmability.

He was a member of the Steering Group, Future Internet Assembly, where he also led the Management and Service-Aware Networking Architecture Working Group. He was the TPC chair of 14 IEEE conferences.

He is involved in IETF and ITU-T SG13 network slicing activities. He was the a Vice Chair of the ITU-T SG13 Group on future networking. He is also a Co-Editor of the *IEEE Communications Magazine* feature topic on advances in networking software. He is also involved in the IEEE SDN initiative.