2017 IEEE 37th International Conference on Distributed Computing Systems

Approximation and Online Algorithms for NFV-Enabled Multicasting in SDNs

Zichuan Xu‡, Weifa Liang†, Meitian Huang†, Mike Jia†, Song Guo¶, and Alex Galis‡
† The Australian National University, Canberra, ACT 0200, Australia
‡ University College London, London WC1E 7JE, UK
¶ The Hong Kong Polytechnic University, Hong Kong
Email: z.xu@ucl.ac.uk, wliang@cs.anu.edu.au, u4700480@anu.edu.au, u5515287@anu.edu.au, song.guo@polyu.edu.hk, a.galis@ucl.ac.uk

***Abstract*—Multicasting is a fundamental functionality of networks for many applications including online conferencing, event monitoring, video streaming, and system monitoring in data centers. To ensure multicasting reliable, secure and scalable, a service chain consisting of network functions (e.g., firewalls, Intrusion Detection Systems (IDSs), and transcoders) usually is associated with each multicast request. Such a multicast request is referred to as an NFV-enabled multicast request. In this paper we study NFV-enabled multicasting in a Software-Defined Network (SDN) with the aims to minimize the implementation cost of each NFV-enabled multicast request or maximize the network throughput for a sequence of NFV-enabled requests, subject to network resource capacity constraints. We first formulate novel NFV-enabled multicasting and online NFV-enabled multicasting problems. We then devise the very first approximation algorithm with an approximation ratio of $2K$ for the NFV-enabled multicasting problem if the number of servers for implementing the network functions of each request is no more than a constant K (≥ 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 online algorithm with a competitive ratio of $O(\log n)$ when $K = 1$, 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 outperform other existing heuristics.**

I. Introduction

Network Function Virtualization (NFV) [2], [3], [7], [16], [25] is emerging as a promising paradigm that is shaping the future networking landscape, by not only bringing the promise of enabling inexpensive and flexible management solutions but also introducing new challenges to the area of network management. 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, it is very expensive to achieve the benefits of middleboxes in conventional networks, since the middleboxes are typically made by dedicated hardware devices. Underpinned by the NFV technique, Software-Defined Networking (SDN) that separates the control plane from the data plane can be utilized to enable inexpensive and flexible implementation of network functions as software components running in Virtual Machines (VMs), rather than expensive and hard-to-manage hardware middleboxes. Multicasting in SDNs that transmits data from one source to multiple destinations is a fundamental functionality of the networks, which has wide applications, such as video conferencing, multimedia distribution, software updates, and system monitoring in data centers. Such multicast requests usually require to forward their traffic to some middleboxes before reaching their destinations for security and performance concerns. To admit multicast requests with network function requirements to be implemented as VMs, in this paper 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, as the VMs in servers for network function implementations are located at different places of the SDN, while the locations of servers partially determine the cost of implementing requests. Naive placements of the VMs of each NFV-enabled multicast request at locations that are far away from its source and its destinations may incur a prohibitive communication cost [21], [22], [23], [24]. In addition, multicast requests usually arrive in the network one by one without the knowledge of future request arrivals. This leads to difficulties to estimate dynamic workloads of computing and bandwidth resources at servers and links. The challenges thus are (1) how to jointly find one or multiple servers to implement the network functions and a multicast route for each incoming multicast request while meeting its computing and bandwidth demands, (2) how to design a metric that can accurately capture the dynamic resource usage and workload in an SDN, and (3) how to devise an online algorithm with a provable competitive ratio to maximize the number of multicast requests admitted, subject to network resource capacity constraints.

Several studies on multicasting in SDNs have been conducted recently [9], [10], [27], [28]. However, most of these work did not consider network function requirements in multicast requests [9], [10], and only dealt with a single multicast request [27], [28]. In contrast, we here investigate NFV-enabled multicasting, by devising not only an approximation algorithm with a provable approximation ratio for realizing a single NFV-enabled multicast request but also an 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 the fine tradeoff between computing and bandwidth resource consumptions if no more than K servers are employed to implement the service chain of each request. We devise an approximation algorithm for the problem. We also study online

1063-6927/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDCS.2017.43

NFV-enabled multicasting and devise the very 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 that reduce the problems into other well-known optimization problems.

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 very first approximation algorithm with an approximation ratio of $2K$ for minimizing the implementation cost of the request, assuming that the number of servers used for implementing the service chain of each request is no more than K. We also investigate dynamic admissions of NFV-enabled multicast requests without the knowledge of future request arrivals with an aim to maximize the network throughput, and propose an online algorithm with a provable competitive ratio for it when $K = 1$. We finally evaluate the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms outperform other 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 capacity constraints. Section V devises an online algorithm for the online NFV-enabled multicasting problem when $K = 1$. Section VI evaluates the performance of the proposed algorithms by experimental simulation, and Section VII concludes the paper.

II. Related Work

Previous studies have extensively explored the issues on placement and resource allocation for NFVs in SDNs [2], [3], [15], [14], [18], [26]. For example, Moens *et al.* [18] investigated efficient NFV placements in SDNs, by focusing on a hybrid scenario where some network functions are implemented by dedicated physical hardware and others are implemented in VMs. Lukovszki *et al.* [15] 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). Li *et al.* [14] designed and implemented a system to provide dynamic provisions of resources 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 utilizing Integer Linear Programming (ILP) and randomized rounding techniques. Cao *et al.* [1] dealt with policy-aware traffic engineering in SDNs, by assuming that the traffic has to pass a given sequence of network functions. Cohen *et al.* [3] considered NFV placements in an SDN with and without server capacity constraints for NFV-enabled unicast requests. Kuo *et al.* [13] studied how to implement a single NFV-enabled unicast request with the end-to-end delay constraint by utilizing existing NFV in servers, and proposed a dynamic programming solution to the problem. These studies however do not consider multicasting in SDNs.

There are several studies that focused on multicasting in SDNs [9], [27], [28]. Huang *et al.* [10] devised the very first online algorithms with provable competitive ratios for online unicasting and multicasting in SDNs, under both node and link capacity constraints. However, they did not consider network function requirements of multicast requests. Huang *et al.* [9] studied the scalability problem of multicasting in SDNs, by proposing an efficient algorithm to find a branch-aware Steiner Tree (BST) for each multicast request. Their solution however may not be applicable to the NFV-enabled multicasting problem, as they did not take into account the NFV requirements of requests.

A very close work related to this paper is the one due to Zhang *et al.* [27], [28]. They investigated the NFV-enabled multicasting problem in an SDN, by assuming that there are sufficient computing and bandwidth resources in the SDN to accommodate any multicast request, for which they provided a 2-approximation algorithm if only one server ($K = 1$) is deployed for implementing the service chain of each multicast request. However, their method cannot be extended to a general case where multiple servers are employed. In reality, both computing and bandwidth resources in an SDN are limited, they need to be carefully allocated. Also, they did not consider online admissions of multicast requests. Since different requests may have different resource demands, it is crucial to determine that which requests should be admitted/rejected to maximize the network throughput.

III. Preliminaries

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

A. System model

We consider a software-defined network $G = (V, E)$ with a set V of SDN-enabled switch nodes and a set E of links between SDN-enabled switch nodes. 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 like 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 a switch node $v \in V_S$ and the bandwidth capacity of a link $e \in E$ in G, respectively. There is an SDN controller in G that controls the allocations of both the computing and bandwidth resources of G to meet the resource demands of each admitted NFV-enabled multicast request. Fig. 1 is an example of an SDN.

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*

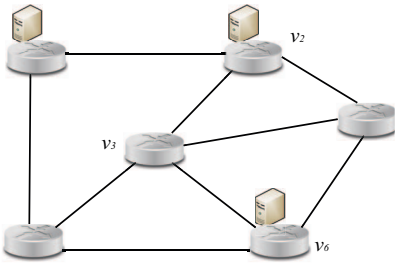


Fig. 1. An SDN G with a set $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ of SDN switches and a subset $V_S = \{v_1, v_2, v_6\}$ of switches with servers.

of r_k that consists of a sequence of network functions that must be implemented by either dedicated hardware middleboxes or virtual machines running on servers. Specifically, the service chain SC_k of request r_k enforces every message from the source of r_k to go through each of the network functions in the chain in the specified order prior to reaching its destinations, as illustrated in Fig. 2. The network functions in SC_k can be implemented by VMs in servers [7], [17], [19]. Without loss of generality, we assume that the network functions in SC_k are consolidated to a server in G. In other words, when realizing multicast request r_k, its message passes a server hosting the VM of its service chain SC_k, the traffic will be directed to the VM and all the network functions in SC_k. Denote by $C_v(SC_k)$ the amount of demanded computing resource to implement SC_k of multicast request r_k in server $v \in V_s$.

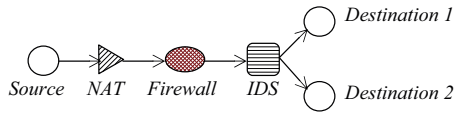
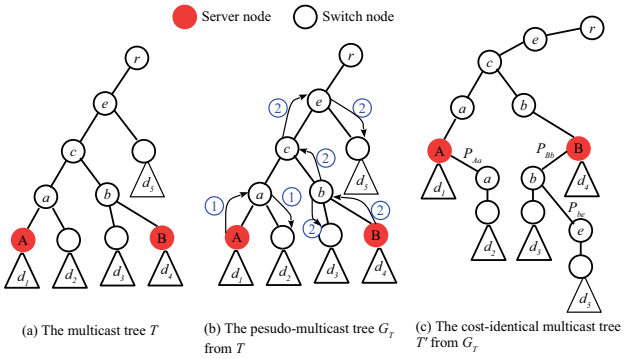


Fig. 2. A service chain $\langle$ NAT, Firewall, IDS $\rangle$.

A pseudo-multicast tree in G is a subgraph G_T derived from a multicast tree T of G for routing the data traffic of an NFV-enabled multicast request. We here use an example to illustrated the pseudo-multicast tree concept. Consider a multicast tree T as shown in Fig. 3, 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 of request r_k 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. 3(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. 3(b)). We term this routing graph derived from T as a pseudo-multicast tree G_T, as shown by Fig. 3(b). It can be seen that another tree T' (see Fig. 3(c)) derived from G_T will have the same cost as G_T, i.e., $c(T') = c(G_T)$.



(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. 3. 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 is constructed which has the identical cost as G_T.

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 the following NFV-enabled multicasting problems with and without various resource capacity constraints.

Case 1. The SDN $G = (V, E)$ has sufficient computing and bandwidth resources to meet the resource demands of any NFV-enabled multicast request. As the network operator of G charges each admitted multicast request on a pay-as-you-go basis, the major concern of the service provider is its *operational cost* that is defined as the sum of the costs of its computing and bandwidth resource consumptions for 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. Since the computing resource demands of VNFs of the service chain of each request usually is within the capacity of each server, we assume that the number of servers, each of which implements an instance of the service chain SC_k of an admitted multicast request r_k, is no more than a constant K with $K \geq 1$. The *NFV-enabled multicasting problem without SDN resource capacity constraint* 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 a constant number K of servers are used for implementing its service chain SC_k, assuming that G has sufficient computing and bandwidth resources.

Case 2. Both computing and communication resources of 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 it. Or it is too expensive to admit the request, i.e., the request should be rejected. We thus define the *NFV-enabled multicasting problem with SDN resource capacity constraint* in an SDN $G = (V, E)$ for an NFV-enabled multicast request r_k is to find a pseudo-multicast tree in G for implementing r_k such that its implementation cost is minimized, if no more than a constant number K of servers are used for implementing its service chain SC_k, subject to

computing and bandwidth capacity constraints on servers and links of G.

Both the defined problems are NP-hard, as their special case – the traditional multicast problem without the NFV constraints is NP-hard [4]. So far we have only considered a single request admission. In reality, the requests arrive into the system one by one without the knowledge of future request arrivals, we refer to this dynamic request admission as online request admissions. Considering that the limited resources of G may not be able to admit all incoming requests, some of the requests will be rejected. We thus formulate this dynamic admissions of multicast requests as the *online NFV-enabled multicasting problem* in G to admit as many NFV-enabled multicast requests as possible without the knowledge of future request arrivals, while meeting the computing and bandwidth resource demands of each admitted multicast request, subject to the computing and bandwidth capacity constraints on servers and links of G, assuming that no more than K servers are used to implement the service chain of each request.

IV. Approximation Algorithms for the NFV-enabled Multicasting Problem

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

A. Algorithm overview

The basic idea of the proposed approximation algorithms is to find a multicast tree rooted at the source and spanning all destinations, and each message 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 tradeoff 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 chosen server v is far from the destinations of r_k. Furthermore, if there are multiple servers in different branches of the multicast tree, then the message can pass through each of these servers to reach the destinations in D_k, and thus leads to less bandwidth usages from the source to the destinations, at the expense of high computing cost. We thus identify a set of servers with each implementing the service chain SC_k of r_k and find a pseudo-multicast tree for the request with the identified server(s) on the path from the source s_k to each destination $u \in D_k$. As K is a constant, we aim to find a pseudo-multicast tree in G that contains no more than K 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 $K \leq |V_s|$. As a pseudo-multicast tree for any NFV-enabled multicast request can contain at least one but no more than K servers, there are at most $\binom{|V_S|}{K}$ 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. Our strategy here is to reduce the multicast tree problem into a Steiner tree problem in an auxiliary undirected graph. An approximate solution to the latter will return 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 problem in G without network resource capacity constraints, 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 $1 \leq i \leq \binom{|V_S|}{K}$, 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 the 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 now 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. 4.

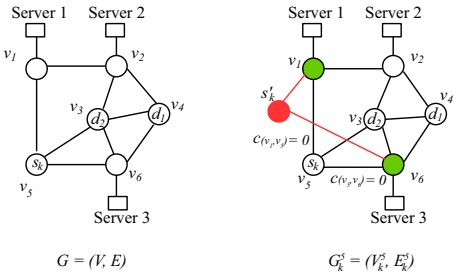


Fig. 4. An example of the auxiliary graph $G_k^5 = (V_k^5, E_k^5)$ constructed from an SDN $G = (V, E)$ with $V_S^5 = \{v_1, v_6\}$, assuming that $K = 2$ and $V_S = \{v_1, v_2, v_6\}$. There are $\binom{|V_S|}{K} = 3 \cdot 2 = 6$ auxiliary graphs derived from G, and all different combinations of servers 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, we assume that $V_S = \{v_1, v_2, \ldots, v_{|V_S|}\}$. Having constructed auxiliary graph G_k^i, we find a Steiner tree in G_k^i for request r_k. Specifically, we first find a minimum spanning tree 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 the shortest path in G_k^i between its two endpoints. Let H_k^i be a subgraph of G_k^i derived from T_{mst}^i by replacing each edge of T_{mst}^i with its corresponding shortest path in G_k^i. We then find an approximate Steiner tree T_k^i in H_k^i by applying the approximation algorithm due to Kou *et al.* [12], which will serve as the multicast tree for r_k. The detailed description of the algorithm is given in `Algorithm` 1.

Algorithm 1 `Appro_Multi`

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

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 $\binom{|V_S|}{K}$ **do**
4: Let $V_S^i = \{v_{i_1}, v_{i_2}, \ldots, v_{i_l}\} \subseteq V_S$;
5: Construct an auxiliary undirected graph $G_k^i = (V_k^i, E_k^i)$ as illustrated by Fig. 4;
6: Find an MST T_{mst}^i 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_k^i between its two endpoints;
7: Let H_k^i be a subgraph of G_k^i derived from T_{mst}^i, by replacing each edge of T_{mst}^i with the corresponding shortest path in G_k^i; find an approximate Steiner tree T_k^i in H_k^i rooted at s_k' and spanning nodes in D_k, by invoking the approximation algorithm due to Kou *et al.* [12];
8: **if** $c(T_k^i) < cost_k$ **then**
9: $cost_k \leftarrow c(T_k^i)$, $T_k \leftarrow T_k^i$; /* put this solution as a candidate solution to the problem */
10: **if** T_k contains node s_k **then**
11: Merge nodes s_k and s_k' into s_k';
12: 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;
13: **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 computing and bandwidth resource capacity constraints, by performing some 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 $b_k\}$, 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 $r_k\}$. `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 node 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`.

D. Algorithm analysis

In the following we show the correctness of `Algorithm 1`, and analyze its time complexity and approximation ratio. The analysis under resource capacity constraints can be similarly performed, and thus omitted.

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 an approximation algorithm, `Algorithm 1`, for the NFV-enabled multicasting problem with and without SDN resource capacity constraints, which delivers an approximate solution with an approximation ratio of $2K$, assuming no more than K servers will be employed for its service chain implementation. The time complexity of the algorithm is $O(|V|^3 \cdot |V_S|^K)$, where $|V_S| \ll |V|$ and $K \geq 1$ is a small integer.

Proof: Clearly, according to the construction of G_k^i, the solution delivered by `Algorithm 1` is feasible.Due to space limitation, the detailed proof for feasibility is omitted.

We now 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 transformation in Section III; 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^* for implementing SC_k with $1 \leq l \leq K$. Without loss of generality, we assume that these l nodes are $v_1, v_2, \ldots, v_l$, respectively. Clearly, it can be easily 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. We now claim that the cost of tree T_c^* is no greater than l times the cost of tree T^*, i.e., $c(T_c^*) \leq l \cdot c(T^*)$ by showing the claim as follows.

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 algorithm due to Kou *et al.* [12], then $c(T_k^{app,i}) \leq 2c(T_k^{OPT,i}) \leq 2c(T_k^i) \leq 2c(T_c^*) = 2 \cdot l \cdot c(T^*)$. Since a pseudo-multicast tree T_k with the minimum cost from the $\binom{|V_S|}{K}$ auxiliary undirected graphs G_k^i for all i with $1 \leq i \leq \binom{|V_S|}{K}$ will be found and the value of l is within $[1, K]$, the cost of the pseudo-multicast tree T_k for r_k is no greater than $2K \cdot c(T^*)$.

The analysis of the time complexity of `Algorithm 1` is omitted, due to space limitation. ■

V. Online Algorithm for the Online NFV-enabled Multicasting Problem

We now study the online NFV-enabled multicasting problem in G with SDN resource capacity constraints. We first propose a novel cost model to capture dynamic resource consumptions in G. We then devise an online algorithm with a competitive ratio for the problem when $K = 1$.

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 metric to capture dynamic consumptions of its resources in order to better utilize the resources, by encouraging the use of underloaded resources while restricting the use of overloaded resources to maximize the number of NFV-enabled multicast request admissions. A simple cost model which is referred to *the linear cost model*, is widely adopted to charge each request with a cost that is linearly proportional to the amount of its resource consumption, regardless of whether a specific resource has been overloaded or underloaded. Clearly, this model may lead to some resources being under-utilized while others being over-utilized. Consequently, significant number of requests may be rejected due to unbalanced resource utilization.

We here introduce a novel cost model that assigns an underloaded 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 computing and bandwidth resources at server node v and link e:

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

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. (1) 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.,

$$c_e(k) = B_e(\beta^{1-\frac{B_e(k)}{B_e}} - 1), \tag{2}$$

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. We will use the defined costs to guide the network resource allocations to meet the demands of NFV-enabled multicast requests.

B. Online algorithm

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 such a pseudo-multicast tree, 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 the normalized cost of the cost defined by Eq. (2), 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. (1), 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.* [12], 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 multicast tree for r_k. If server v is in any path in T from s_k to each destination, T is the multicast tree, and its cost is no more than twice the optimal one [12]; otherwise, assume that the path between s_k and a destination node d does not contain server v. Let u be the lowest common ancestor $u = LCA(v, d)$ between nodes v and d in T. Then, when the message from s_k is sent to server v for processing, the processed message continues forwarding to all destinations in the subtree rooted at v; for the destination $d \in D_k$, the processed message at node v is then sent back to node u, which then forwards toward destination d. Clearly, in the worst scenario, the processed message 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 [12] 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 the pseudo-multicast tree T_k based on T and server v is

$$\begin{aligned} &w(T_k) + w_v(k) = w(T) + w(P_{v,u}^T) + w_v(k) \le w(T) \\ &+ w(P_{v,s_k}^T) + w_v(k) \le 2w(T) + w_v(k) \le 2(w(T) + w_v(k)) \\ &\le 4w(T^*) + 2w_v(k) \le 4(w(T^*) + w_v(k)) \le 4OPT_v, \end{aligned} \tag{3}$$

where $P_{x,y}^T$ 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 $w_v(k) \ge \sigma_v$ for any $v \in V_S \cap T_k$, r_k will be rejected; and (b) if $\sum_{e \in T_k} w_e(k) \ge \sigma_e$, 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` 2.

C. Algorithm analysis

The rest is to analyze the competitive ratio of algorithm `Online_CP`. Let $\mathcal{S}(k)$ and OPT be the sets of admitted multicast requests by algorithm `Online_CP` 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 `Online_CP` while admitted by the optimal offline algorithm. Then the competitive ratio of algorithm

Algorithm 2 `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 weighted graph $G_k = (V_k, E_k; w)$, by assigning each link $e \in E$ a normalized weight $w_e(k)$ and each node $v \in V_S$ a normalized weight $w_v(k)$;
  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.* [12];
      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 $\mathcal{G}$.

`Online_CP` 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 `Online_CP` consists of three steps: (1) show the upper bound on the accumulative computing and bandwidth resources occupied by requests in $\mathcal{S}(k)$; (2) show the lower bound on the accumulative computing and bandwidth resources occupied by requests in $\mathcal{R}(k)$) and (3) derive the competitive ratio by combining the results of steps (1) and (2). We then have the following lemma.

Lemma 1: When multicast request r_k arrives, the cost sums of all servers in V_S and all links in E are $\sum_{v \in V_S} c_v(k) \leq 2\mathbb{C}(k) \cdot \log \alpha \cdot (|V| - 1)$, and $\sum_{e \in E} c_e(k) \leq 2\mathbb{B}(k) \cdot \log \beta \cdot (|V| - 1)$, 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.

Proof: The basic idea of the proof is to first derive the upper bound of the difference between the cost of each node v when $(k'+1)$th request's arrival and that of v when the k'th request's arrival. Then, the upper bound of the total cost difference of all nodes in V_S is calculated. The similar approach will be used to derive the upper bound on the edge cost. Due to space limitation, the detailed proof is omitted. ∎

We now show the lower bound on the cost of a multicast request in $\mathcal{R}(k)$ that is rejected by online algorithm `Online_CP` but admitted by an optimal offline algorithm.

Lemma 2: For $r_{k'} \in \mathcal{R}(k)$, if $\alpha = \beta = 2|V|$, we have

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

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 $1 \leq k' \leq k$.

Proof: A multicast request $r_{k'}$ will be rejected by the proposed online algorithm, `Algorithm Online_CP`, 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).

The detailed proof of the mentioned three cases are omitted, due to space limitation. ∎

We finally analyze the competitive ratio of algorithm `Online_CP`.

Theorem 2: 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 Online_CP`, 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.

Proof: The competitive ratio of algorithm `Online_CP` 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 $r_{k'}$.

$$\begin{aligned}
&\frac{|V| - 1}{4}(|\mathcal{R}(k)|) \leq \sum_{r_{k'} \in \mathcal{R}(k)} \frac{|V| - 1}{4} \\
&\leq \sum_{r_{k'} \in \mathcal{R}(k)} \sum_{e \in T^*_{k'}} (\beta^{1 - \frac{B_e(k')}{B_e}} - 1) + \alpha^{1 - \frac{C_{v^*}(k')}{C_{v^*}}} - 1, \text{by Lemma 2} \\
&= \sum_{r_{k'} \in \mathcal{R}(k)} (\sum_{e \in T^*_{k'}} c_e(k')/B_e + c_{v^*}(k')/C_{v^*}), \\
&\leq \sum_{r_{k'} \in \mathcal{R}(k)} (\sum_{e \in T^*_{k'}} c_e(k)/B_e + c_{v^*}(k)/C_{v^*}), \\
&\leq \sum_{e \in T^*_{k'}} c_e(k) \sum_{r_{k'} \in \mathcal{R}(k)} 1/B_e + c_{v^*}(k) \sum_{r_{k'} \in \mathcal{R}(k)} 1/C_{v^*}, \\
&\leq \sum_{e \in E} c_e(k) + \sum_{v \in V_S} c_v(k).
\end{aligned} \tag{5}$$

Following inequalities (5) and Lemma (1), we have

$$\begin{aligned}
&\frac{|V|-1}{4}(|\mathcal{R}(k)|) < \sum_{e\in E} c_e(k) + \sum_{v\in V_S} c_v(k)\\
&\leq 2\mathbb{C}(k)\log\alpha(|V|-1) + 2\mathbb{B}(k)\log\beta(|V|-1)\\
&\leq 2|\mathcal{S}(k)|C_{\max}\log\alpha(|V|-1) + 2|\mathcal{S}(k)|b_{\max}\log\beta(|V|-1)\\
&= 2|\mathcal{S}(k)|(|V|-1)(C_{\max}\log\alpha + b_{\max}\log\beta), \qquad (6)
\end{aligned}$$

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 (6) then can be rewritten as

$$\frac{|\mathcal{R}(k)|}{|\mathcal{S}(k)|} \leq 8(C_{\max}\log\alpha + b_{\max}\log\beta). \qquad (7)$$

The competitive ratio of algorithm `Online_CP` thus is

$$\begin{aligned}
&\frac{|\mathcal{S}(k)|}{|OPT|} \geq \frac{|\mathcal{S}(k)|}{|\mathcal{R}(k)\cup\mathcal{S}(k)|} \geq \frac{|\mathcal{S}(k)|}{|\mathcal{R}(k)|+|\mathcal{S}(k)|}\\
&= \frac{1}{|\mathcal{R}(k)|/|\mathcal{S}(k)|+1}\\
&\geq \frac{1}{1+8(C_{\max}\log\alpha + b_{\max}\log\beta)}, \quad \text{by inequality (7)}\\
&\geq \frac{1}{c'\log|V|}, \text{when } C_{\max} \text{ and } b_{\max} \text{ are given constants,}
\end{aligned}$$

where $c' > 0$ is a positive constant and $\alpha = \beta = 2|V|$. The competitive ratio of the competition ratio of algorithm `Online_CP` thus is $O(\log|V|)$ when $\alpha = \beta = 2|V|$. The analysis of the time complexity of the proposed algorithm is omitted due to space limitation. ■

VI. Performance Evaluation

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

A. Environment settings

We consider SDNs consisting of from 50 to 250 nodes, where each network is generated using GT-ITM [6]. 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 [5] and an ISP network from [20]. There are nine servers for the GÉANT topology as set in [7] and the number of servers in the ISP networks are provided by [19]. The bandwidth capacity of each link varies from 1,000 Mbps to 10,000 Mbps [11], and the computing capacity of each server varies from 4,000 to 12,000 MHz [8]. Five types of network functions, i.e., Firewall, Proxy, NAT, IDS, and Load Balancing, are considered, and their computing demands are adopted from [7], [17]. 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 $[50, 200]$ $Mbps$. We set σ_e and σ_v at $|V|-1$. The maximum number K 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.

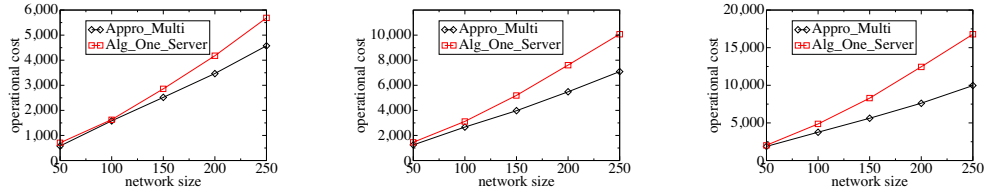
We evaluate the performance of algorithm `Appro_Multi` against the state-of-the-art, an algorithm in [27], referred to as algorithm `Alg_One_Server` that only uses a single server to implement service chain SC_k of each multicast request r_k. Namely, algorithm `Alg_One_Server` first routes the traffic 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.

We then study the performance of online algorithms. We evaluate the performance of algorithm `Online_CP` against the one of a baseline heuristic `SP`. For each 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 multicast tree with the minimum cost for r_k.

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. 5, where the operational costs and running times are the average of admitting $1,000$ NFV-enabled multicast requests. Specifically, we can see from Fig. 5 (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. 5(b) and (c). Furthermore, it can be seen from Figures 5 (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.

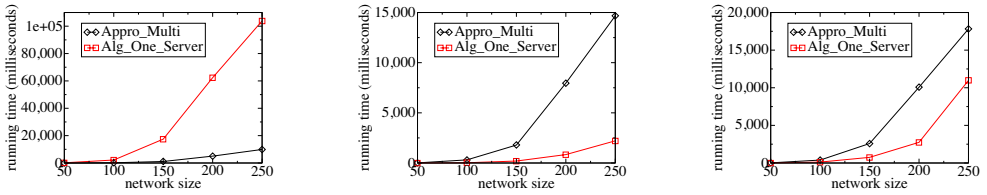
We then investigate the performance of approximation algorithms `Appro_Multi` and `Alg_One_Server` in real networks GÉANT and AS1755, by varying $\frac{D_{max}}{|V|}$ from 0.05 to 0.2. Fig. 6 shows that the operational costs and running times of both algorithms. It can be seen that the operational cost delivered by algorithm `Appro_Multi` is much lower than that



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

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

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



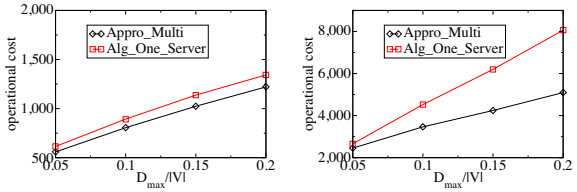
(d) Running times of different algorithms with $D_{max}/|V| = 0.05$

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

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

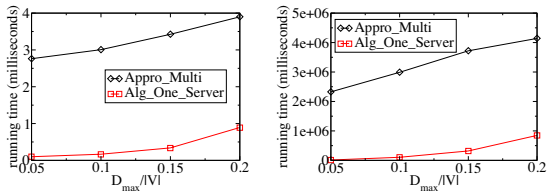
Fig. 5. The performance of algorithms `Appro_Multi` and `Alg_One_Server`

by algorithm `Alg_One_Server` while taking a slightly more running time. For example, the operational cost 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. 6 (b).



(a) Operational costs by different algorithms in GÉANT

(b) Operational costs by different algorithms in AS1755

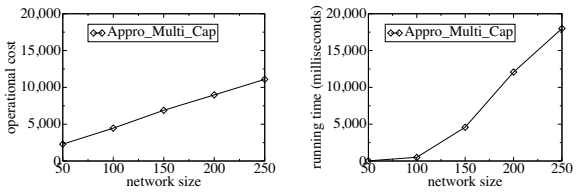


(c) Running times of different algorithms in GÉANT

(d) Running times of different algorithms in AS1755

Fig. 6. The performance of different algorithms in networks of GÉANT, AS1755, and AS4755

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 Figs. 7(a) and 5(c), we can see that the operational cost of algorithm `Appro_Multi_Cap` is larger than that of algorithm `Appro_Multi` without capacity constraints. 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.



(a) The operational cost

(b) The running time

Fig. 7. The performance of algorithms `Appro_Multi_Cap`

C. Performance evaluation of online algorithms

We then evaluate algorithm `Online_CP` against algorithm `SP`, 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 algorithm `Online_CP` are shown in Fig. 8. Specifically, from Fig. 8 (a), we can see that online algorithm `Online_CP` admits at least twice the number of requests admitted by algorithm `SP`. Although the total resource capacity of each network keeps increasing with the increase on the network size, the number of requests admitted by each mentioned algorithm does not monotonically increase. This is because there is a higher probability that the destinations of each multicast request are far away from each other, thereby increasing the rejection probability of the request, due to more bandwidth resource consumptions.

We now evaluate the performance of algorithm `Online_CP` against that of online algorithm `SP` by varying the number of requests from 50 to 300, in GÉANT

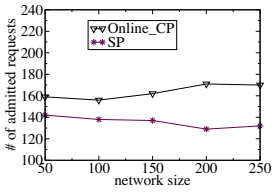
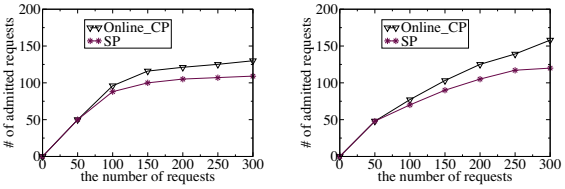


Fig. 8. The performance of online algorithms `Online_CP` and `SP`

and AS1755, respectively. It can be seen from Fig. 9 (a) that algorithms `Online_CP` and `SP` can admit almost all requests if the number of requests is no greater 100. Otherwise, algorithm `Online_CP` admits more requests than that by algorithm `SP`. Also, the performance gap between them increases with the growth of the number of requests. The reason is that online algorithm `Online_CP` considers the workload (or the utilization) of each resource by assigning each resource 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. Similar performance of algorithms `Online_CP` and `SP` in network AS1755 can also be observed in Fig. 9 (b).



(a) Network GÉANT (b) Network AS1755

Fig. 9. The performance of algorithms `Online_CP` and `SP`

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 $K \geq 1$, subject to the computing and bandwidth capacity constraints on servers and links in the network. We then studied 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 proposed an online algorithm with a competitive ratio if $K = 1$. We finally evaluated the performance of the proposed algorithms by experimental simulations. Simulation results demonstrate that the proposed algorithms outperform the other heuristics.

Acknowledgements

The work by Zichuan Xu and Alex Galis is partially supported by the H2020 projects of SONATA (http://sonata-nfv.eu) and 5GEx (https://5gex.tmit.bme.bu).

References

[1] Z. Cao, M. Kodialam, and T. V. Lakshman. Traffic steering in software defined networks: planning and online routing. *Proc. of DCC*, ACM, 2014.

[2] R. Cohen, L. Eytan, J. Naor, and D. Raz. On the effect of forwarding table size on SDN network utilization. *Proc. of INFOCOM*, IEEE, 2014.

[3] R. Cohen, L. Eytan, J. Naor, and D. Raz. Near optimal placement of virtual network functions. *Proc. of INFOCOM*, IEEE, 2015.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the Theory of NP-Completeness*. W.H. Freeman and Company, NY, 1979.

[5] GÉANT. `http://www.geant.net`.

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

[7] A. Gushchin, A. Walid, and A. Tang. Scalable routing in SDN-enabled networks with consolidated middleboxes. *Proc. of HotMiddlebox*, ACM, 2015.

[8] Hewlett-Packard Development Company. L.P. Servers for enterprise bladeSystem, rack & tower and hyperscale. ttp://www8.p.com/us/en/products/servers/, 2015.

[9] L. Huang, H. Hung, C. Lin, and D. Yang. Scalable steiner tree for multicast communications in software-defined networking. *Computing Research Repository (CoRR)*, vol. abs/1404.3454, 2014.

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

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

[12] L. Kou, G. Markowsy, and L. Berman. A faster algorithm for Steiner trees. *Acta Informatica*, Volume 15, pp. 141–145, Springer, 1981.

[13] T-W. Kuo, B-H. Liou, K. C. Lin, and M-J Tsai. Deploying chains of virtual network functions: on the relation between link and server usage. *Proc. of INFOCOM*, IEEE, 2016.

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

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

[16] L. Mamatas, S. Clayman, and A. Galis. Software-defined infrastructure. *IEEE Communications Magazine*, Vol. 53, No. 4, pp 166-174, IEEE, 2015.

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

[18] H. Moens and F. D. Turck. VNF-P: A model for efficient placement of virtualized network functions. *Proc. of CNSM*, IEEE, 2014.

[19] Z. A. Qazi, C. C. Tu, L. Chiang, R. Miao, V. Sekar, M. Yu. SIMPLE-fying middlebox policy enforcement using SDN. *Proc. of SIGCOMM*, ACM, 2013.

[20] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with rocketfuel. *Proc. of SIGCOMM*, ACM, 2002.

[21] Z. Xu and W. Liang. Minimizing the operational cost of data centers via geographical electricity price diversity, *Proc. of 6th IEEE International Conference on Cloud Computing*, IEEE, 2013.

[22] Z. Xu and W. Liang. Operational cost minimization for distributed data centers through exploring electricity price diversity. *Computer Networks*, Vol. 83, pp.59-75, Elsevier, 2015.

[23] Z. Xu, W. Liang, and Q. Xia. Electricity cost minimization in distributed clouds by exploring heterogeneities of cloud resources and user demands. *Proc. of ICPADS'15*, IEEE, 2015.

[24] Z. Xu, W. Liang, and Q. Xia. Efficient embedding of virtual networks to distributed clouds via exploring periodic resource demands. To appear in *IEEE Transactions on Cloud Computing*, Vol.XX, IEEE, 2016.

[25] Z. Xu, W. Liang, A. Galis, and Y. Ma. Throughput maximization and resource optimization in NFV-enabled networks. To appear in *Proc. of ICC'17*, IEEE, 2017.

[26] Y. Zhang *et. al*. StEERING: A software-defined networking for inline service chaining. *Proc. of ICNP*, IEEE, 2013.

[27] S. Q. Zhang, Q. Zhang, H. Bannazadeh, and A. L. Garcia. Network function virtualization enabled multicast routing on SDN. *Proc. of ICC*, IEEE, 2015.

[28] S. Q. Zhang, Q. Zhang, H. Bannazadeh, and A. L. Garcia. Routing algorithms for network function virtualization enabled multicast topology on SDN. *IEEE Transaction on Network and Service Management*, Vol.12, No.4, pp.580–594, 2015.