NFV-Enabled Multicasting in Mobile Edge Clouds with Resource Sharing

Zichuan Xu
Dalian University of Technology
Dalian, Liaoning, China
z.xu@dlut.edu.cn

Yutong Zhang
Dalian University of Technology
Dalian, Liaoning, China
zhangyutong@mail.dlut.edu.cn

Weifa Liang
The Australian National University
Canberra, Australia
wliang@cs.anu.edu.au

Qiufen Xia
Dalian University of Technology
Dalian, Liaoning, China
qiufenxia@dlut.edu.cn

Omer Rana
Cardiff University
Cardiff, United Kingdom
RanaOF@cardiff.ac.uk

Alex Galis
University College London
London, United Kingdom
a.galis@ucl.ac.uk

Guowei Wu
Dalian University of Technology
Dalian, Liaoning, China
wgwdut@dlut.edu.cn

Pan Zhou
Huazhong University of Science and Technology
Wuhan, Hubei, China
zhoupannewton@gmail.com

ABSTRACT

Driven by stringent delay requirements of mobile applications, the mobile edge cloud has emerged as a major platform to offer low latency network services from the edge of networks. Most conventional network services are implemented via hardware-based network functions, such as firewalls and load balancers, to guarantee service security and performance. However, implementing such hardware-based network functions incurs high purchase and maintenance costs. Network function virtualization (NFV) as a promising technology exhibits great potential to reduce the purchase and maintenance costs by implementing network functions as software in virtual machines (VMs). In this paper, we consider a fundamental problem of NFV-enabled multicasting in a mobile edge cloud, where each multicast request requires to process its traffic in a specified sequence of network functions (referred to as a service chain) before the traffic from a source to a set of destinations. We devise a provable approximation algorithm with an approximation ratio for the problem if requests do not have delay requirements; otherwise, we propose an efficient heuristic for it. We also evaluate the performance of the proposed algorithms against the state-of-the-art NFV-enabled multicasting algorithms, and results show that our algorithms outperform their counterparts.

ACM Reference Format:

Zichuan Xu, Yutong Zhang, Weifa Liang, Qiufen Xia, Omer Rana, Alex Galis, Guowei Wu, and Pan Zhou. 2019. NFV-Enabled Multicasting in Mobile Edge Clouds with Resource Sharing. In *Proceedings of ICPP '19: 48th International Conference on Parallel Processing (ICPP '19).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

ICPP '19, August 5-8, 2019, Kyoto Research Park, Kyoto, Japan

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Data traffic in the mobile edge has been increasing dramatically due to abundant multimedia data and data generated from various applications, such as social networks and Internet-of-Things (IoT). Such data traffic usually needs to be transferred to multiple subscribers, which is referred to as *multicasting*. It however places a great strain on the network, by not only requiring various network functions such as firewalls, Intrusion Detection Systems (IDSs), proxies, and WAN optimizers, to guarantee the data transfer security, but also having stringent Quality-of-Service (QoS) requirements to make sure that the traffic is transferred on time. The emerging technique of Mobile Edge Cloud (MEC) [4, 10, 11, 13, 17, 19, 23, 30, 31, 33] enables the provisioning of low-latency and inexpensive resources for multicasting within the proximity of users in edge of networks. Also, Network Function Virtualization (NFV) utilizes virtualization technology to reduce dependency on underlying hardware by moving network functions from dedicated hardware to virtual machines (VMs) that can run on commodity hardware, thereby reducing the maintenance cost. In this paper, we consider NFV-enabled multicasting in a mobile edge cloud, where each user request requires to forward its traffic to pass a sequence of network functions, referred to as *service chains*, before reaching its destinations.

Implementing NFV-enabled multicast requests in mobile edge clouds poses many challenges. The first challenge is that a mobile edge cloud usually has limited computing resource in implementing VNFs of service chains. Allowing multicast requests to share existing VNF instances can significantly improve the resource utilization of the mobile edge cloud. It however requires strategic selections of existing VNF instances or creating new VNF instances. Careless selection or instantiation of VNF instances can lead to low resource utilizations in the mobile edge cloud. That is, how to find appropriate cloudlets for the VNFs of a service chain of a request such that its implementation cost is minimized while meeting its end-to-end delay requirement? Which existing VNF instances can be used for which request? The second challenge is that each NFV-enabled multicast request usually has a QoS requirement to guarantee its traffic reaching its destinations within the specified end-to-end delay requirement. How to meet the end-to-end delay requirement of each admitted NFV-enabled multicast request is challenging. The third challenge is that each service chain consists of a sequence of VNFs, and where such VNFs can be placed and how should the placed VNFs be chained together, given that the computing resource of each cloudlet is limited. How to jointly route the traffic of each request and place VNF instances into the mobile edge cloud?

Existing studies on multicasting usually focus on conventional networks or software-defined networks, without considering the service chain requirements of requests [12, 13, 34]. The solutions of these studies thus cannot be applied to NFV-enabled multicasting directly, due to lack of efficient methods of jointly finding locations for VNFs and routing paths for the request. There are a few recent studies on NFV-enabled multicasting. They however did not consider the delay requirements of multicast requests [26], assumed that only one instance is used for each VNF of the service chain [34], or the VNFs in each service chain are consolidated into a single location [31]. This unfortunately may increase the cost/delay of implementing NFV-enabled multicast requests. The reason is that placing VNFs into multiple cloudlets allows the selection of cloudlets that can achieve lower costs/delays.

The main contributions of this paper are as follows. We first study the problem of NFV-enabled multicasting problem in a mobile edge cloud with cloudlet being deployed, our objective is to minimize the implementation cost of each NFV-enabled multicast request. Specifically, for a NFV-enabled multicast request, we assume that each cloudlet has sufficient computing resource to accommodate the request. We devise the very first approximation algorithm with an approximation ratio for it by a novel reduction of the problem without end-to-end delay requirements to a Steiner tree problem. We also propose an efficient heuristic for NFV-enabled multicasting problem with the end-to-end delay requirements of requests. We finally evaluate the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms outperform existing algorithms.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 introduces the system model, notations, and problem definition. Section 4 devises an approximation algorithm for the NFV-enabled multicasting problem without end-to-end delay requirements in a mobile edge cloud, and proposes an efficient heuristic for the problem with delay requirements through using the proposed approximation algorithm as a subroutine. Section 6 evaluates the performance of the proposed algorithms by experimental simulation, and Section **??** concludes the paper.

2 RELATED WORK

Recently traffic steering in NFV-enabled networks has attracted much attention from the literature [3, 4, 10, 11, 13, 17, 19, 23, 30, 31, 33]. Most of these studies investigated unicasting between pairs of nodes. For example, Moens *et. al.* [23] focused on hybrid networks with both hardware and software network functions. Yu *et. al* [21] investigated the profit maximization of placing VNFs into a set of locations, and they considered the delay requirement of each unicast request. Also, Xu *et. al.* [29] studied the offloading of delay-sensitive tasks with network function requirements in a mobile edge cloud network by proposing efficient heuristics and an online algorithm with a competitive ratio. Although there exist studies that consider the delay requirements of user requests [16, 21, 29], they only consider unicast requests and their solutions cannot be applied to the NFV-enabled multicasting problem. Recently, Chen and Wu [3] devised a series of innovative algorithms for the VNF placement. Their algorithms show great potential in balancing the set-up and bandwidth consumption costs, thereby minimizing the cost of implementing NFV-enabled unicast requests.

There are studies on multicasting in conventional wired or wireless networks [1, 24]. Recently, with the emerging of new networking technologies such as mobile edge computing, software-defined networking (SDN) and NFV, multicasting has re-gained the attention by the research community [12, 13]. For example, Huang *et al.* [13] studied online multicasting in software-defined networks with both node and link capacity constraints. Huang *et al.* [12] studied the scalability problem of multicasting in SDNs, by proposing an efficient algorithm to find a branch-aware Steiner Tree for each multicast request. These solutions however cannot be directly applied to the problem of NFV-enabled multicasting in mobile edge clouds, because they did not consider the service chain requirements of multicast requests.

Studies that investigated NFV-enabled multicasting include [26, 27, 31, 34]. For instance, Zhang *et al.* [34] investigated the NFV-enabled multicasting problem in an SDN without resource capacities, assuming that data traffic of each multicast request can only be processed by one server. Xu *et al.* [31, 32] considered the NFV multicasting problem by assuming the traffic of each request can be processed by multiple servers, with the objective to minimize the implementation cost. Approximation and online algorithms are proposed. They however assumed that the VNFs in each service chain is consolidated into a single data center. Ren *et al.* [26] investigated the problem of embedding a service graph that consisting of VNF instances into a substrate network. This study assumed that the traffic of each multicast request can be processed by multiple instances of the VNFs in its service chain. An approximation algorithm with an approximation ratio of $1 + \rho$ is proposed, where ρ is the best approximation ratio of Steiner tree problem. Soni *et al.* [27] proposed a scalable multicast group management scheme and a load balancing method for the routing of best-effort traffic and bandwidth-guaranteed traffic. These studies did not consider the end-to-end delay requirement of each multicast request.

3 PRELIMINARIES

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

3.1 System model

We consider a mobile edge cloud $G = (V, E)$ with a set V of switches, a set C of cloudlets that can implement various network functions as software in VMs, and a set E of links between switches and the cloudlets. Each cloudlet is attached to a switch in V via optical fibers, and the communication delay between a switch and its attached cloudlet is negligible. Let V_{CL} be the set of switches with attached cloudlets, clearly, $V_{CL} \subseteq V$. Cloudlets are usually deployed in shopping malls, airports, or base stations. Due to the space limitations of installing cooling equipments in those places, each cloudlet is usually equipped with limited number of servers and thus has limited computing resource to implement VNF instances. Denote by C_v the computing capacity of the cloudlet attached to a switch node $v \in V_{CL}$. In addition, transferring data through links in E incurs communication latencies. Let d_e be the delay of transmitting a unit data traffic via link $e \in E$. We assume that there is an SDN controller that both makes traffic steering decisions and manages network function instances that run on a server in the mobile edge cloud G. Fig. 1 is an illustrative example of a mobile edge cloud.

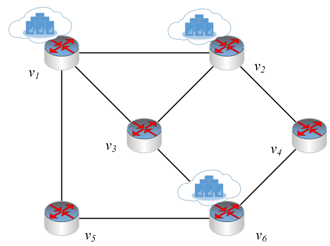


Figure 1: A cloud network G with a set $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ of SDN switches and a subset $V_{CL} = \{v_1, v_2, v_6\}$ of switches with attached cloudlets.

3.2 NFV-enabled multicast requests and service chains

A NFV-enabled multicast request is a request that transfers an amount of data traffic from a source to a given set of destinations and requires the traffic being processed by a sequence of VNFs before reaching its destinations. Let r_k be a NFV-enabled multicast request, which is denoted 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 with $D_k \subseteq V$, b_k is the size of its data traffic, and SC_k is the *service chain* of r_k that consists of a sequence of VNFs. Without loss of generality, we assume that the data traffic b_k of each request r_k is given, as it usually can be obtained from historical information. Let $\mathcal{F}$ be the set of network functions that are provided by the network service provider in the mobile edge cloud G. A network function $f_l \in \mathcal{F}$ can be needed by request r_k to form its service chain SC_k. Assume that there are L_k network functions in SC_k, where $1 \leq l \leq L_k$ for each SC_k and $SC_k \subset \mathcal{F}$. We further assume that there is a number of already instantiated VNF instances for each type of network function f_l in cloudlets of the mobile edge cloud G. Due to the resource capacity constraints of the cloudlets, we allow the instances of VNF f_l can be shared among different requests.

To implement request r_k, we need to enforce every data packet in its traffic from the source s_k of r_k to go through an instance of each network function $f_l \in SC_k$ in its service chain prior to reaching its destinations in D_k, as illustrated in Fig. 2. To this end, an instance must be selected for each VNF $f_l \in SC_k$, or a new instance of f_l must be instantiated in a cloudlet of G. Without loss of generality, we assume that existing or newly created instances of VNFs of SC_k can be placed in multiple cloudlets, because a single cloudlet may not have all the instances of the VNFs in SC_k or there is inadequate computing resource in a cloudlet to create new instances for all VNFs in SC_k.

Each multicast request needs an amount of computing resource to process its traffic. Let $C_{unit}(f_l)$ be the amount of computing resource needed to process a unit amount of its data r_k. If f_l is implemented in a newly created instance of f_l, the total amount of computing resource that should be assigned to the new instance is $C_{unit}(f_l) \cdot b_k$. Otherwise, an existing instance of f_l should have at least an amount $C_{unit}(f_l) \cdot b_k$ of available computing resource to process the traffic of r_k. Notice that we assume that the accumulative available resources in the cloudlets of G are higher then the total resource demand of a single request r_k; however, for a specific cloudlet in V_{CL}, it may not have enough amount of resource that is demanded by r_k.

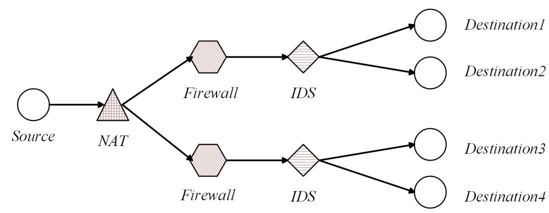


Figure 2: A service chain $\langle$ NAT, Firewall, IDS $\rangle$ with one instance of NAT and two instances of Firewall and IDS.

3.3 Delay requirements of multicast requests

The experienced delay of each NFV-enabled multicast request consists of the total processing delay in the selected cloudlets and the total transfer delay from the source to cloudlets and from the cloudlets to the destinations, which are defined in the following.

Processing delay: The processing delay experienced by a multicast request r_k depends on both the amount of data traffic that needs to be processed and the computing resource assigned to process the traffic. Without loss of generality, we assume that the processing delay $d^p_{k,l}$ of each multicast request r_k by VNF f_l is proportion to the amount of traffic it needs to process, i.e.,

$$d^p_{k,l} = \alpha_l \cdot b_k, \tag{1}$$

where α_l is a given proportional factor of VNF f_l and b_k is size of data traffic of request r_k. The accumulative processing delay incurred due to the traffic processing by network functions in SC_k of r_k thus is

$$d^p_k = \sum\nolimits_{f_l \in SC_k} d^p_{k,l}. \tag{2}$$

Transmission delay: Let $\mathcal{P}_k$ be the set of routing paths from source s_k to destinations in D_k, with each path $p_m \in \mathcal{P}_k$ denotes a routing path from s_k to destination $t_m \in D_k$. The transmission delay of each r_k is the maximum end-to-end delay incurred in the paths in $\mathcal{P}_k$. Denote by d^t_k the transmission delay of request r_k, which can be defined as,

$$d^t_k = \arg\max\nolimits_{p_m \in \mathcal{P}_k} \sum_{e \in p_m} d_e \cdot b_k. \tag{3}$$

The delay experienced by multicast request r_k thus is

$$d_k = d_k^p + d_k^t, \tag{4}$$

which needs to be no greater than its specified delay requirement D_k, i.e.,

$$d_k \le D_k. \tag{5}$$

3.4 Cost models

As the network service provider that operates its mobile edge cloud G charges each admitted multicast request on a pay-as-you-go basis, the major concern of the service provider is its *operational cost*, which usually consists of computing resource usage costs in cloudlets, bandwidth resource usage costs in links, and VNF instance instantiation costs. Let $c(e)$ and $c(v)$ be the usage costs of one unit of bandwidth and computing resources at link $e \in E$ and cloudlet $v \in V_{CL}$, respectively. Denote by $c_l(v)$ the cost of instantiating an instance of network function f_l in cloudlet $v \in V_{CL}$, and let $n'_{l,v}$ be the number of newly created instances for network function f_l in cloudlet v.

The operational cost of the admission of multicast request r_k thus is

$$c_k = \sum_{f_l \in SC_k} \sum_{v \in V_{CL}} \Big(y_{k,l,v} \cdot \big(c(v_{f_l,r_k}) \cdot b_k + c_l(v_{f_l,r_k}) \big) + n'_{l,v} \cdot c_l(v) \Big) + \sum_{e \in T_k} c(e) \cdot b_k. \tag{6}$$

3.5 Problem definition

Given a mobile edge cloud (MEC) $G = (V, E)$ with a set V_{CL} of cloudlets with $V_{CL} \subset V$, and a set of multicast requests R in which each multicast request r_k is represented by $(s_k, D_k; b_k, SC_k)$, assuming that each multicast request can be implemented using the computing resources assigned to existing VNF instances, the *NFV-enabled multicasting problem* in mobile edge cloud G is to route the traffic of request r_k to each destination in D_k by chaining either existing or newly created instances of VNF in different cloudlets, such that the operational cost (i.e., Eq.(6)) of the implementation of r_k is minimized, while meeting the end-to-end delay requirement D_k of r_k and the capacity constraint on each cloudlet $v \in V_{CL}$.

The NFV-enabled multicasting problem is NP-hard, as its special case – the traditional multicast problem without NFV service chain constraints is NP-hard [5].

4 ALGORITHMS FOR THE NFV-ENABLED MULTICASTING PROBLEM IN AN MEC

In this section we deal with the NFV-enabled multicasting problem. We first devise an approximation algorithm for the problem without delay requirements. We then propose an efficient heuristic for the problem by incorporating the end-to-end delay requirement.

4.1 An approximation algorithm for the problem without delay requirements

The basic idea of the proposed approximation algorithm is to reduce the problem in a sub-network of G to the Steiner tree problem in an auxiliary graph G', via a non-trivial reduction. Since each cloudlet $v \in V_{CL}$ has computing capacity to implement the VNFs of each request, the VNFs in each service chain SC_k can be implemented in multiple cloudlets or consolidated into a single cloudlet to save the communication cost due to the transmissions between cloudlets. To guarantee that each cloudlet has sufficient computing resource to implement the VNFs in SC_k of each multicast request r_k, we adopt a conservative method of reserving $\sum_{f_l \in SC_k} b_k \cdot C_{unit}(f_l)$ resource for r_k in each cloudlet. The cloudlet with an amount of available computing resource that is less than $\sum_{f_l \in SC_k} b_k \cdot C_{unit}(f_l)$ will be removed from the network G, where the available resource in idle VNF instances are also accounted.

The construction of auxiliary graph $G' = (V', E')$: We now construct the auxiliary graph G' based on the sub-network of G. To this end, we start by constructing the node set V' of G'. Specifically, we first add source node s_k into the auxiliary graph. We also add each node in V into V', i.e., $V' \leftarrow V' \cup V$. Notice that, since $V_{CL} \subset V$, all switch nodes in V_{CL} are added into V as well. However, only their functionalities of forwarding traffic will be used.

A multicast request can share the computing resource that is assigned to an existing idle VNF instance, as long as its assigned computing resource is larger than the amount demanded by r_k, i.e., $C_v(f_l) \ge C_{unit}(f_l) \cdot b_k$. Or, VNFs in SC_k of multicast request r_k can be assigned to newly instantiated VNF instances. To determine whether we make use of existing VNF instances or creating new ones, we create a *widget* for each cloudlet $v \in V_{CL}$ and network function $f_l \in SC_k$ to represent the resource availability of the cloudlet v for f_l by two cases: **case 1**: the amount of available computing resource to instantiate new instances of VNFs; **case 2**: existing VNF instances of f_l in $v \in V_{CL}$ that are available to process the traffic of r_k.

For **case 1**, we add a pair of *virtual VNF nodes* into the widget, to represent each of existing VNF instances of f_l with sufficient computing resource processing the data traffic of r_k in cloudlet $v \in V_{CL}$. Denote by $f'_{i,l,v}$ and $f''_{i,l,v}$ the pair of virtual VNF nodes for the ith VNF instance of f_l in cloudlet $v \in V_{CL}$. We then add an edge from $f'_{i,l,v}$ to $f''_{i,l,v}$ into the widget. The weight of edge $\langle f'_{i,l,v}, f''_{i,l,v} \rangle$ is the cost of processing a unit traffic by an existing VNF instance of f_l in cloudlet v, i.e., $w(f'_{i,l,v}, f''_{i,l,v}) = c(v_{f_l,r_k})$.

For **case 2**, we add a pair of *virtual cloudlets* for each cloudlet $v \in V_{CL}$ into each widget to denote the amount of available computing resource to instantiate a new instance of f_l in cloudlet v, as shown in Fig. 3. Let $v'_{k,l}$ and $v''_{k,l}$ be such a pair of virtual cloudlets for the lth VNF and cloudlet v. To make sure the processing and transmission costs are considered jointly, we connect each pair of virtual cloudlets, $v'_{k,l}$ and $v''_{k,l}$, i.e., $E' \leftarrow E' \cup \{\langle v'_{k,l}, v''_{k,l} \rangle\}$. The weight of edge $\langle v'_{k,l}, v''_{k,l} \rangle$ is the sum of the instantiation cost of VNF f_l and the cost of processing a unit traffic by the lth VNF in SC_k for each multicast request r_k in cloudlet v. That is, $w(\langle v'_{k,l}, v''_{k,l} \rangle) = \frac{c_l(v)}{b_k} + c(v_{f_l,r_k})$.

We also add a *widget source node* $ws_{l,v}$ and a *widget destination node* $wd_{l,v}$ for the widget for network function f_l and cloudlet $v \in V_{CL}$. Node $ws_{l,v}$ is connected to node $v'_{k,l}$ and the node f'_l for each existing instance of network function f_l that has enough computing resource to process the data traffic of r_k. In addition, node $v'_{k,l}$ and node f'_l for each existing instance of network function f_l are both connected with the widget destination node $wd_{l,v}$. The weights of those edges are set to zeros. It must be mentioned that widget source and destination nodes are used to guarantee that either a new instance for f_l is created or an existing VNF instance of f_l is selected to process the traffic of r_k, which will be proved in the algorithm analysis part.

The widgets are then added into the auxiliary graph G'.

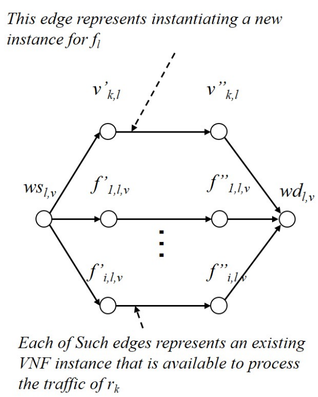


Figure 3: An example of the widget for the VNF f_l in SC_k and cloudlet $v \in V_{CL}$

We then connect the constructed widgets and other nodes in the auxiliary graph G' as follows.

- **s_k to widget source nodes:** There is an edge from source node s_k to each widget source node $ws_{l,v}$ of the widget for the first VNF f_1 of SC_k and every $v \in V_{CL}$. The weight of edge $\langle s_k, ws_{l,v}\rangle$ is set as the transmission cost of data traffic of r_k
- **widget destination to widget source nodes:** Since the data traffic of r_k may be processed by multiple cloudlets, there is an edge from the widget destination node of each widget for network function f_l to the widget source node of each widget for VNF f_{l+1}, for each l with $1 \le l \le L_k - 1$, i.e., $E' \leftarrow E' \cup \{\langle wd_{l,v}, ws_{l+1,u}\rangle\}$ for l with $1 \le l \le L_k - 1$ and v, u in V_{CL}. The weight of edge $\langle wd_{l,v}, ws_{l+1,u}\rangle$ is the transmission cost of a unit traffic along the shortest path from cloudlet v to cloudlet u
- **widget destinations of f_{L_k} to cloudlet nodes:** We finally connect each of the widgets that are created for the last VNF $f_{L_k} \in SC_k$ with the cloudlet node. Specifically, there is an edge from node $wd_{L_k,v}$ to cloudlet node u in V', i.e., $E' \leftarrow E' \cup \{\langle wd_{L_k,v}, u\rangle\}$. The weight of edge $\langle wd_{L_k,v}, u\rangle$ is the transmission cost of a unit traffic along the shortest path from cloudlet v to cloudlet u.

An example of the constructed auxiliary graph is shown in Fig. 4.

Problem transformation: We now reduce the original problem of G into the Steiner tree problem in the auxiliary graph G'. Recall that in the construction of G', the VNF processing and transmission costs are considered as the weights of edges. We thus find a Steiner tree that spans nodes in $\{s_k\} \cup D_k$ of the auxiliary graph G'. We then transfer the Steiner tree in auxiliary graph G' to routing paths for r_k in the original network G. Specifically, if a widget for $f_l \in SC_k$ of and cloudlet $v \in V_{CL}$ is included in the Steiner tree, either a newly created instance or an existing one in cloudlet v will be used to implement f_l, depending on which edge of the widget is included in the Steiner tree. Notice that the edges among widgets in G' correspond to the shortest paths of their endpoints of the edges in the original network G. We thus replace each of such edges with its shortest path in G. Notice that if there is cloudlet node along the shortest path, only its forwarding functionality will be adopted, and the traffic will not be forwarded to it for processing.

Algorithm 1 `Appro_NoDelay`

Input: $G = (V, E)$, V_{CL}, computing capacity C_v for each cloudlet $v \in V_{CL}$, and a multicast request $r_k = (s_k, D_k; b_k, SC_k)$.

Output: The locations for the VNFs of service chain SC_k of multicast request r_k and the multicast tree T_k to transfer its data.

1: Construct an auxiliary directed graph $G' = (V', E')$, as shown in Fig. 4;

2: Find a directed Steiner tree T in G' that spans nodes in $\{s_k\} \cup D_k$, using Charikar's algorithm [2];

3: For each path from the widget source node to the widget destination node of a widget in T, condense the path to a single node;

4: Replace each of all other edges in T with its corresponding shortest path in network G; /*The edges among widgets correspond to shortest paths in the original network G. */

4.2 A heuristic algorithm for the problem

So far, the proposed algorithm does not consider the delay requirement of each multicast request r_k. We now propose a heuristic for the NFV-enabled multicasting problem, by using **Algorithm** 1 as a subroutine. For each multicast request r_k, the proposed algorithm first ignores the delay requirement of r_k and invokes **Algorithm** 1 to find routing paths for the request to multicast its traffic to destinations in D_k. The obtained solution however may not meet the delay requirement of multicast request r_k.

We then adjust the obtained solution to make sure the end-to-end delay requirement is met. To this end, we observe that a longer delay will incur if the VNFs of SC_k are implemented in more cloudlets, because the data transfer among different cloudlets incurs delay. However, putting all VNFs into a single cloudlet may also incur a longer delay, since the selected cloudlet may be far away from the destinations of multicast request r_k. We thus find an appropriate number of cloudlets to implement the VNFs in SC_k, by adopting binary search to find a proper number. Specifically, let n'_k be the number of cloudlets that are used to implement the VNFs in SC_k in the current infeasible solution, and denote by n_k the appropriate number of cloudlets in the feasible solution. We first set $n_k = \lfloor \frac{|V_{CL}|+1}{2} \rfloor$. The proposed algorithm first tries to re-assign the VNFs in SC_k such that they are implemented in exactly n_k cloudlets. If $n_k < n'_k$, we identify a number of $(n'_k - n_k)$ cloudlets that implements VNFs of SC_k in the obtained infeasible solution from the Steiner tree in G' and have the longest average data transfer delay from it to the destinations in D_k. Let F' be the set of instances of VNFs in SC_k that are implemented in the identified cloudlets. The VNFs in F' are pre-consolidated to the rest n_k cloudlets in V' one by one, by selecting a cloudlet with the lowest implementation cost for each $f_l \in F_{v'}$. If the pre-consolidation makes the delay requirement of r_k being met, the algorithm terminates with a feasible solution. Otherwise, if the experienced delay of r_k is reduced but still greater

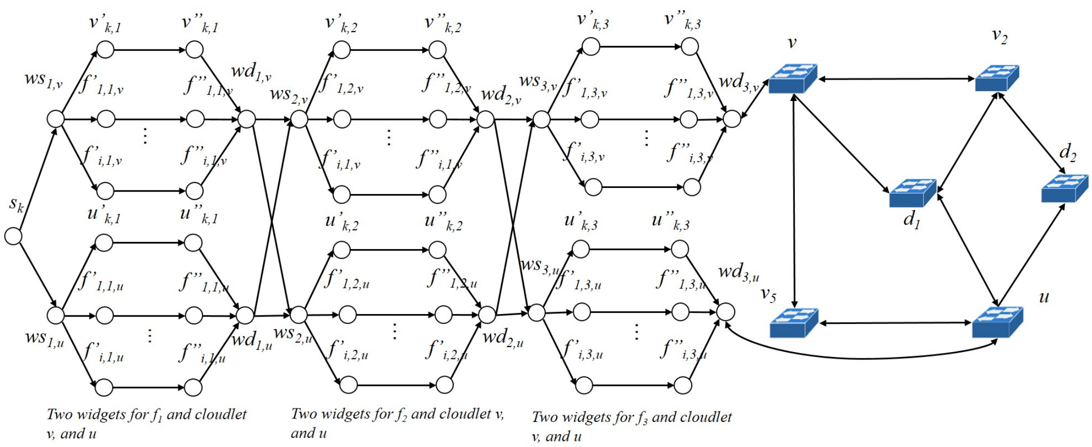


Figure 4: An example of the auxiliary graph $G' = (V', E')$ with two servers attached at node v and node u and multicast request r_k transfer its data to destinations in $D_k = \{d_1, d_2\}$.

than its requirement, we continue the above procedure by searching the appropriate number of cloudlets in the range of $[1, n_k]$. Instead, if the experienced delay is increased, we try to find the appropriate value for n_k in the range of $[n_k, |V_{CL}|]$. This means increasing the number of cloudlets for r_k may reduce the experienced delay of multicast request r_k. On the other hand, if $n_k > n'_k$, we need to find the additional $n_k - n'_k$ cloudlets that have the lowest implementation cost for VNFs of r_k, and pre-assign VNFs in F' to the cloudlets one by one. The above binary search procedure continues until a feasible solution is obtained or the multicast request is rejected.

4.3 Algorithm analysis

We analyze the solution feasibility and performance of the proposed algorithms.

We first show the feasibility of the algorithm 1. Intuitively, if a solution to the NFV-enabled multicasting problem for a single multicast request is feasible, it needs to satisfy the following conditions:

- **condition (1)**: each VNF $f_l \in SC_k$ will be assigned to one or multiple cloudlets by either creating a new instance or using an existing instance
- **condition (2)**: the traffic of r_k will be processed by VNFs as the specified order in SC_k
- **condition (3)**: the processed traffic by the VNFs in SC_k is forwarded to destinations in D_k of r_k.

For condition (1), we show that in each of the selected cloudlets for f_l, either a new instance is created or an existing instance is selected for it, in the following lemma.

Lemma 1. *If a cloudlet $v \in V_{CL}$ is selected for VNF $f_l \in SC_k$ of multicast request r_k, either an existing instance of f_l or a newly created instance is used to process the traffic of r_k.*

Proof. According to the construction of auxiliary graph G', showing the feasibility is to show that if the Steiner tree found in auxiliary graph G' has one path from $ws_{l,v}$ to $wd_{l,v}$ of each selected widget in it, the path will be the only path in the Steiner tree, and no other paths in the widget will be included. Let $W_{l,v}$ be the widget that is built for network function f_l in cloudlet $v \in V_{CL}$. Assume that widget $W_{l,v}$ is included into the Steiner tree for the subgraph, and let p be the path from $ws_{l,v}$ to $wd_{l,v}$ of $W_{l,v}$ in G' that is included in the Steiner tree. We prove by contradiction. Assume that there is another instance (either newly created or existing one)

Algorithm 2 Heu_Delay

Input: $G = (V, E)$, V_{CL}, computing capacity C_v for each cloudlet $v \in V_{CL}$, and a multicast request $r_k = (s_k, D_k; b_k, SC_k)$ and its delay requirement d_k^{req}.

Output: The locations for the VNFs of service chain SC_k of multicast request r_k and the multicast tree T_k to transfer its data.

```
Find a multicast tree for r_k without considering its delay requirement d_k^{req}, by invoking algorithm 1;
Let n'_k be the number of cloudlets that are used to implement VNFs in SC_k;
n_min ← 1;
n_max ← |V_CL|;
while n_min <= n_max do
    n_k ← ⌊(n_min + n_max)/2⌋;
    if n_k < n'_k then
        Identify the number of n'_k − n_k cloudlets that implements VNFs of SC_k in the obtained solution from the Steiner tree in G' and has the top-( n'_k − n_k) highest average data transfer delays from it to the destinations in D_k;
        Move the VNFs that were implemented in the n'_k − n_k cloudlets of the infeasible solution to the rest cloudlets one by one.
    else
        Find the additional n_k − n'_k cloudlets that have the lowest implementation cost for VNFs of r_k, and assign VNFs in F_{v'} to the cloudlets one by one.
    end if
    if the experienced delay of r_k is met then
        return;
    else
        if the experienced delay of r_k is decreased then
            n_max ← n_k;
        else
            n_min ← n_k;
        end if
    end if
end while
```

of f_l is used to process the traffic of r_k. Let the ith instance of f_l be such an additional instance. This means that edge $\langle f'_{i,l,v}, f''_{i,l,v}\rangle$ has to be included in the Steiner tree found in G'. To this end, edges $\langle ws_{l,v}, f'_{i,l,v}\rangle$ and $\langle f''_{i,l,v}, wd_{l,v}\rangle$ have to included, according to the structure of the widget; otherwise, edge $\langle f'_{i,l,v}, f''_{i,l,v}\rangle$ is a standalone edge that can be removed. Let p' be the path that consisting of edges $\langle ws_{l,v}, f'_{i,l,v}\rangle$, $\langle f'_{i,l,v}, f''_{i,l,v}\rangle$, and $\langle f''_{i,l,v}, wd_{l,v}\rangle$, as shown in Fig. 5. Paths p' and p however make it not a tree. Therefore, only one path from $ws_{l,v}$ to $wd_{l,v}$ will be included in

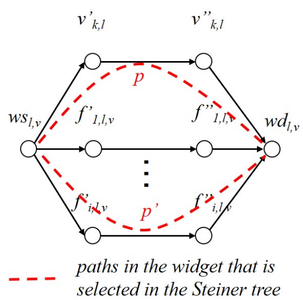


Figure 5: A widget and its paths from its source to destination nodes that are selected in the Steiner tree.

the Steiner tree for the subgraph of G' that is composed of source node s_k and the widgets, meaning that a newly created or existing instance of f_l will be selected in cloudlet $v \in V_{CL}$. The lemma holds. □

We condition (2) in the following lemma.

Lemma 2. *The traffic of r_k will be processed by the VNF instances in SC_k in the specified order.*

Proof. Assume that the traffic of r_k is not processed by the specified order in SC_k. We have the following two cases: (1) two instances of the same VNF f_l processed the traffic, and (2) the traffic of r_k is processed by a previous VNF f_{l-1} after being processed by f_l.

For case (1), the two instances must be in different cloudlets as shown in Lemma 1. This means that two widgets of the same VNF f_l is selected in the Steiner tree in G'. According to the construction of G' and Lemma 1, if the instances of f_l in two cloudlets are used, the source and destination nodes of the corresponding two widgets have to be included in the Steiner tree in G'; otherwise, the edges will be standalone edges that can be removed from the Steiner tree. Therefore, according to the problem transformation method of the algorithm, this will correspond to the processing of r_k's traffic by two instances of f_l in different cloudlets, rather than a sequence processing of the two instances. Similarly, for case (2), if the Steiner tree includes a widget for f_{l-1}, it has to include the source and destination nodes of the widget, which actually means a sequence process of the traffic by f_{l-1} and f_l. Therefore, these two cases are not possible according to the construction of G' and the problem transformation methods.

In addition, since each edge in G' may correspond to a shortest path in G, making the traffic being forwarded to a cloudlet more than once. this does not mean that the traffic is to be processed by the cloudlet twice. This is because we assume in such cloudlets will just forward the traffic instead of processing.

We thus conclude that the traffic of r_k will be processed by the VNFs in the specified order in SC_k. □

We now show condition (3) as follows.

Lemma 3. *The traffic of r_k will be forwarded to its destinations in D_k after being processed by the instances of its VNFs in SC_k.*

Proof. In the construction of the auxiliary graph G', we can see that the destination nodes of the widgets for the last VNF f_{L_k} is connected to its corresponding switch node in the original network. For each $W_{L_k,k}$ of such widgets, if its edges are included in the Steiner tree, edge $\langle wd_{L_k,k}, v\rangle$ has to be included in the Steiner tree. The reasons include (1) this is the only edge to the destination nodes in D_k, and (2) as shown in Lemma 2, the traffic cannot be processed sequentially by other cloudlets of the same VNF f_{L_k} or the instances of its previous VNFs in SC_k. The lemma holds. □

Theorem 1. *Given a mobile edge cloud $G = (V, E)$ with a set V_{CL} of cloudlets and a multicast request r_k $(= (s_k, D_k; b_k, SC_k))$ that requires to transfer an amount b_k of data from its source to a set D_k of destinations and process its traffic by the VNFs in SC_k. There is an approximation algorithm, i.e.,* **Algorithm** *1, for a special case of the NFV-enabled multicasting problem without the delay requirement, which delivers a feasible solution that has an approximation ratio of $i(i-1)|D_k|^{1/i}$ [2], and the time complexity of the approximation algorithm is $O((L_k \cdot |V| \cdot \frac{C_v}{C_{unit}(f_l)} + |V|)^i \cdot |D_k|^{2i})$, where L_k is the number of VNFs in the service chain SC_k of multicast request r_k, i.e., $L_k = |SC_k|$, and i is the level of the directed Steiner tree [2].*

Proof. From Lemmas 1, 2, and 3, we know that the solution obtained by finding a Steiner tree in the auxiliary graph G' is feasible. In the following, we analyze the approximation ratio and the running time of the proposed approximation algorithm.

Assume c^* is the optimal solution for the NFV-enabled multicasting problem. In algorithm 1, we find an approximate Steiner tree T' in the auxiliary graph G'. T' is then converted to routing paths for r_k in G by (1) selecting either an existing instance for a network function or a newly created instance of each VNF f_l in SC_k if the widget for f_l is included in the Steiner tree, and (2) replacing the edges between selected widgets using their corresponding shortest paths in G. In (1), the processing is determined according to which type of VNF instance is selected. In (2), the replaced auxiliary graph edge has the same weight as the total cost of its corresponding shortest path in G. Therefore, the cost do not change in the transfer from tree T' to the multicast tree T for multicast request r_k. Since the approximation ratio of the algorithm in [2] is $i(i-1)|D_k|^{1/i}$, the approximation of algorithm 1 is $i(i-1)|D_k|^{1/i}$ as well.

We now show the time complexity of algorithm 1. It can be seen that the most time consuming part of the algorithm is the finding of a Steiner tree in the auxiliary graph. The time complexity of Charikar's algorithm in auxiliary graph $G' = (V', E')$ is $O(|V'|^3)$ [15]. We can see that there are $O(\frac{C_v}{C_{unit}(f_l)})$ instances of VNF f_l in cloudlet $v \in V_{CL}$. According to the construction of the auxiliary graph, we thus have $O(\frac{C_v}{C_{unit}(f_l)} + 4) = O(\frac{C_v}{C_{unit}(f_l)})$ nodes for each widget. In total, we have $L_k \cdot |V_{CL}|$ widgets. Therefore, there are $O(L_k \cdot |V_{CL}| \cdot \frac{C_v}{C_{unit}(f_l)} + |V|)$ nodes in auxiliary graph G'. The time complexity thus is $O((L_k \cdot |V| \cdot \frac{C_v}{C_{unit}(f_l)} + |V|)^i \cdot |D_k|^{2i})$. □

We now show the performance of algorithm 2 in the following theorem.

THEOREM 2. *Given a mobile edge cloud $G = (V, E)$ with a set V_{CL} of cloudlets and a multicast request r_k (= $(s_k, D_k; b_k, SC_k)$) that requires to transfer an amount b_k of data from its source to a set D_k of destinations with an end-to-end delay requirement d_k^{req} and process its traffic by the VNFs in SC_k. There is a heuristic algorithm, i.e., algorithm 2, for the NFV-enabled multicasting problem for a single multicast request, which delivers a feasible solution in time $O(\lfloor \log V_{CL} + 1 \rfloor \cdot |V|^3 + (L_k \cdot |V| \cdot \frac{C_v}{C_{unit}(f_l)} + |V|)^i \cdot |D_k|^{2i})$, where L_k is the number of VNFs in the service chain SC_k of multicast request r_k, i.e., $L_k = |SC_k|$, and i is the level of the directed Steiner tree [2].*

PROOF. We first show the feasibility of the proposed heuristic by showing that the end-to-end delay requirement of r_k is met as well. Also, algorithm 2 adopts a binary search based heuristic to find the right number of cloudlets for each multicast request r_k until the end-to-end delay requirement of r_k is met or it is rejected. Therefore, as long as the request is admitted, its end-to-end delay requirement is met.

We then show the time complexity of the proposed heuristic. Clearly, in the worse case, the binary search can make $\lfloor \log V_{CL} + 1 \rfloor$ iterations. Within each iteration, the most time consuming parts include (1) the identification of cloudlets that involved finding the delays from cloudlets to destinations in D_k via all pair shortest paths, which take $O(|V|^3)$ time, and (2) the assignment of VNFs one by one, taking $O(|SC_k|)$ time. In total, the time complexity of the proposed heuristic is $O(\lfloor \log V_{CL} + 1 \rfloor \cdot |V|^3 \cdot |SC_k| + (L_k \cdot |V| \cdot \frac{C_v}{C_{unit}(f_l)} + |V|)^i \cdot |D_k|^{2i}) = O(\lfloor \log |V| + 1 \rfloor \cdot |V|^3 + (L_k \cdot |V| \cdot \frac{C_v}{C_{unit}(f_l)} + |V|)^i \cdot |D_k|^{2i})$, assuming that $|SC_k|$ is a small constant. □

5 SIMULATIONS

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

5.1 Environment settings

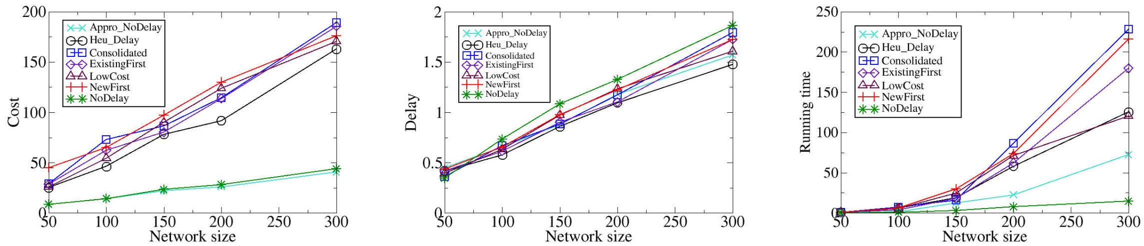
We consider a cloud network consisting of 50 to 250 nodes, where each network is generated using a graph generation tool GT-ITM [7]. The number of servers in each network is set to 10% of the network size, and they are randomly co-located with switches. We also use real network topologies, i.e., GÉANT [6] and an ISP network from [28]. There are nine cloudlets for the GÉANT topology as set in [8] and the number of data centers in the ISP networks are provided by [25]. The computing capacity of cloudlet varies from 40,000 to 120,000 MHz [9] (cloudlets with around tens of servers). Five types of network functions, i.e., Firewall, Proxy, NAT, IDS, and Load Balancing, are considered, and their computing demands are adopted from [8, 22]. 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 $[0.05, 0.2]$. The data of each request is randomly drawn from $[10, 200]$ Megabyte, and the delay requirement of transferring such data is randomly generated from $[0.05, 5]$ *seconds*. Notice that the transfer of larger amount of data can be divided into smaller amounts and transferred by multiple multicast requests. The running time of each algorithm is obtained based on a machine with a 3.70GHz Intel i7 Hexa-core CPU and 16 GiB RAM. Unless otherwise specified, these parameters will be adopted in the default setting.

Benchmark algorithm: We compare the performance of the proposed approximation and heuristic algorithms with the following benchmarks.

- We consider the case where the VNFs of each multicast request may be placed to multiple cloudlets for processing while there exist solutions that consolidate all VNFs of a multicast request into a single location. We thus compare our solutions with such solutions, which is referred to as algorithm `Consolidated`
- We evaluate the performance of the proposed approximation and heuristic algorithms against the one in [26] that does not consider the delay requirement of multicast requests, and we use `NoDelay` to represent the algorithm
- We also compare the performance of our algorithm against that of a greedy solution that prefers to select existing VNF instances for each multicast request r_k. Specifically, it finds the cloudlet that is closest to source node s_k and has an VNF instance for its first VNF in SC_k, if there does not exist such cloudlets a new VNF instance is created in the cloudlet that is closest. The procedure continues until all VNFs in SC_k are considered. This greedy is referred to as algorithm `ExistingFirst`
- Another greedy benchmark prefers to create new instances for each of the VNFs in SC_k, which is referred to as algorithm `NewFirst`
- The fifth benchmark selects the cloudlet that can achieve the lowest processing cost for each VNF in SC_k. For simplicity, it is referred to as algorithm `LowCost`. Specifically, algorithm `LowCost` finds the cloudlet that is closest to the source s_k and then packs as many VNFs in SC_k to the cloudlet until all existing VNF instances are used or no computing resource available to instantiate new ones. If there are still VNFs in SC_k that are not assigned, algorithm `LowCost` finds the next cloudlet that is the closest to the found cloudlets.

5.2 Performance evaluation of algorithms `Appro_NoDelay` and `Heu_Delay`

We first evaluate the performance of algorithm `Appro_NoDelay` and `Heu_Delay` against that of algorithms `Consolidated`, `NoDelay`, `ExistingFirst`, `NewFirst`, and `LowCost` in terms of average operational cost, average end-to-end delay, and running time, by varying the network size from 50 to 250 and fixing the number of requests to 100. Fig. 6 shows the result of the proposed algorithms. From Fig. 6 (a), we can see that algorithm `Heu_Delay` achieves the lower operational cost than algorithms `ExistingFirst`, `NewFirst`, and `LowCost`. The reason is that algorithm `Heu_Delay` jointly considers the use of existing VNF instances and newly instantiated ones. However, greedy approaches `NewFirst`, `ExistingFirst`, and `LowCost` only prefers new, existing, or low processing cost VNF instances.

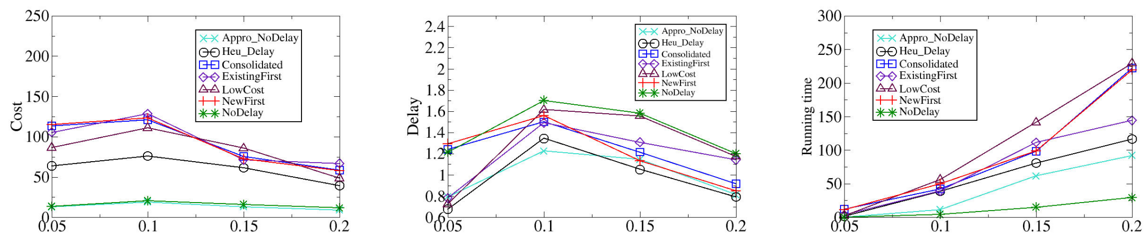


(a) Average cost of implementing a multicast request.

(b) Average delay experienced by a multicast request.

(c) Running times.

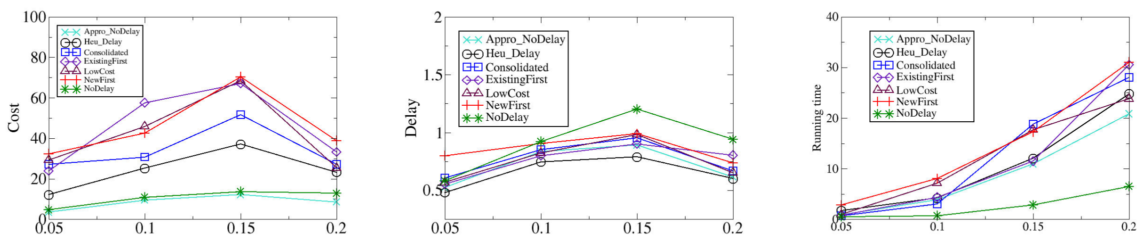
Figure 6: The performance of algorithms `Appro_NoDelay`, `Consolidated`, `NoDelay`, `ExistingFirst`, `NewFirst`, **and** `LowCost`.



(a) Average cost of implementing a multicast request in network AS1755.

(b) Average delay experienced by a multicast request in network AS1755.

(c) Running times in network AS1755.



(d) Average cost of implementing a multicast request in network AS4755.

(e) Average delay experienced by a multicast request in network AS4755.

(f) Running times in network AS4755.

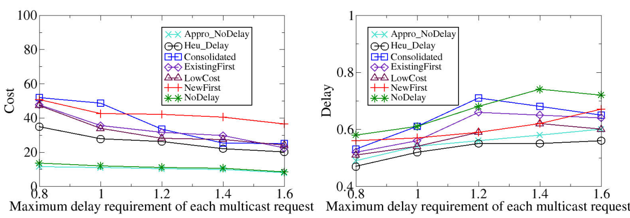
Figure 7: The performance of algorithms `Appro_NoDelay`, `Consolidated`, `NoDelay`, `ExistingFirst`, `NewFirst`, **and** `LowCost` **in networks AS1755 and AS4755.**

They unfortunately could miss the opportunities of further reducing the operational costs if the use of existing VNF instances can save processing cost, creating new VNF instances in close cloudlets may save transmission costs, or there exist cloudlets that can save data transmission costs, respectively. In addition, it can be seen in Fig. 6 (a) that algorithm `Heu_Delay` has a higher operational cost than algorithms `Appro_NoDelay` and `NoDelay`. The reason is that algorithms `Appro_NoDelay` and `NoDelay` does not consider the delay requirement of each multicast request, allowing it to choose cloudlets that can incur lower operational costs. Furthermore, as shown in Fig. 6 (b), the average delay experienced by each multicast request by algorithm `Heu_Delay` is much lower than its counterparts. Also, from Fig. 6 (c), we can see that the running time of algorithm `Heu_Delay` is around 50 seconds for network size 200, which is larger than those of algorithms `Appro_NoDelay` and `NoDelay` and smaller than `ExistingFirst`, `NewFirst`, and `LowCost`.

We then evaluate the performance of algorithm `Appro_NoDelay` and `Heu_Delay` against that of algorithms `Consolidated`, `NoDelay`, `ExistingFirst`, `NewFirst`, and `LowCost`, in real networks AS1755 and AS4755, by varying the ratio of the number of cloudlets and the number of switches, i.e., $|CL|/|V|$ from 0.05 to 0.2. Fig. 7 illustrate the results. From Fig. 7 (a) and (d), we can see that algorithms `Heu_Delay` and `Appro_NoDelay` achieve lower operational costs than algorithms `Consolidated`, `ExistingFirst`, and `NewFirst`, while algorithms `Appro_NoDelay` and `NoDelay` has the highest end-to-end delay for each admitted requests. We can also see that the average cost of implementing a multicast increases first when the ratio $|CL|/|V|$ increases from 0.05 to 0.1 and then decreases afterwards. The rationale behind is that each multicast request may be assigned to more cloudlets for processing with the increase of number of clouds, which increases the transmission cost from its source to the cloudlets and from the cloudlets to its destinations. However, with the further increase of cloudlets, it is more likely these cloudlets are deployed in locations that are close to the source and destinations of the multicast request. The transmission cost thus can be further reduced.

5.3 Impact of the maximum delay requirement

We finally investigate the impact of the maximum delay requirement of each multicast request on the algorithms performance in real network AS1755, by varying the maximum delay requirement of each multicast request from 0.8 seconds to 1.8 seconds with an increase of 0.2 seconds. It can be seen from Fig. 8 that the cost of implementing a multicast request is decreasing with the increase of the maximum delay requirement. The rationale behind is that a higher delay requirement of a request allows the algorithm to select cloudlets with lower costs but further from the source node of the request. Obviously, the experienced delay will be higher, as shown in Fig. 8.



(a) Average cost of implementing a multicast request.

(b) Average delay experienced by a multicast request.

Figure 8: The impact of the maximum delay requirement of each multicast request on the performance of algorithms `Appro_NoDelay`**,** `Consolidated`**,** `NoDelay`**,** `ExistingFirst`**,** `NewFirst`**, and** `LowCost`**.**

6 CONCLUSION

In this paper, we studied the problem of NFV-enabled multicasting in a mobile edge cloud, by exploring the sharing of VNF instances of requests. If cloudlets in the mobile edge cloud have sufficient accumulative computing resource to process the traffic of a multicast request while no delay requirement is considered, we proposed an approximate solution with an approximation ratio that guarantees how far the solution is from the optimal one; otherwise, we developed an efficient heuristic. We evaluated the performance of the proposed algorithms against state-of-the-arts, and the results show that the performance of our algorithms is promising.

REFERENCES

[1] S. M. Banik, S. Radhakrishnan, and C. N. Sekharan. Multicast routing with delay and delay variation constraints for collaborative applications on overlay networks. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 18, No.3, pp. 421 - 431, 2007.

[2] M. Charikar, C. Chekuri, T.-Y. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed Steiner problems. *Proc. of SODA*, IEEE, 1998.

[3] Y. Chen and J. Wu. NFV middlebox placement with balanced set-up cost and bandwidth consumption. *Proc. of ICPP*, ACM, 2018.

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

[5] 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.

[6] GÉANT. `http://www.geant.net`.

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

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

[9] Hewlett-Packard Development Company. L.P. Servers for enterprise ï£¡C bladeSystem, rack & tower and hyperscale. http://www8.hp.com/us/en/products/servers/, 2015.

[10] H. Huang, S. Guo, J. Wu, and J. Li. Service chaining for hybrid network function. To appear in *IEEE Transactions on Cloud Computing*, Vol. XX, IEEE, 2017.

[11] H. Huang, P. Li, and S. Guo. Traffic scheduling for deep packet inspection in software-defined networks. *Concurrency and computation: practice and experience*, Vol. 29, No.16, pp. e3967, Wiley, 2016.

[12] 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.

[13] 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.

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

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

[16] 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.

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

[18] D. H. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters*, Vol. 28, pp. 213-219, Elsevier, 2001.

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

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

[21] Y. Ma, W. Liang, Z. Xu, and S. Guo. Profit maximization for admitting requests with network function services in distributed clouds. To appear in *IEEE Transactions on Parallel and Distributed Systems*, Vol.XX, IEEE, 2018.

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

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

[24] M. Mongioví, A. K. Singh, X. Yan, B. Zong, and K. Psounis. Efficient multicasting for delay tolerant networks using graph indexing. *Proc. of INFOCOM*, IEEE, 2012.

[25] 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.

[26] B. Ren, D. Guo, G. Tang, X. Lin, and Y. Qin. Optimal service function tree embedding for NFV Enabled multicast. *Proc. of ICDCS'18*, IEEE, 2018.

[27] H. Soni, W. Dabbous, T. Turletti, and H. Asaeda. NFV-based scalable guaranteed-bandwidth multicast service for software-defined ISP networks. *IEEE Transactions on Network and Service Management*, Vol.14, No. 5, pp. 1157-1170, 2017.

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

[29] Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao. Task offloading with network function services in a mobile edge-cloud network. *IEEE Transactions on Mobile Computing*, Vol.XX, IEEE, 2018

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

[31] Z. Xu, W. Liang, M. Huang, M. Jia, S. Guo, and A. Galis. Approximation and online algorithms for NFV-enabled multicasting in SDNs. Proc of 37th IEEE Intl Conf on Distributed Computing Systems (ICDCS'17), 2017.

[32] Z. Xu, W. Liang, M. Huang, M. Jia, S. Guo, and A. Galis. Efficient NFV-enabled multicasting in SDNs. To appear in IEEE Transactions on Communications, Vol.XX, Acceptance date: November 8, 2018.

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

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