Efficient Algorithms for Delay-Aware NFV-Enabled Multicasting in Mobile Edge Clouds with Resource Sharing

Haozhe Ren, Zichuan Xu, *IEEE Member,* Weifa Liang *IEEE Senior Member,* Qiufen Xia, *IEEE Member,* Pan Zhou, *IEEE Member,* Omer F. Rana, *IEEE Senior Member,* Alex Galis, *IEEE Senior Member,* and Guowei Wu.

Abstract—Stringent delay requirements of many mobile applications have led to the development of mobile edge clouds, to offer low latency network services at the network edges. Most conventional network services are implemented via hardware-based network functions, including firewalls and load balancers, to guarantee service security and performance. However, implementing hardware-based network functions usually incurs both a high capital expenditure (CAPEX) and operating expenditure (OPEX). Network Function Virtualization (NFV) exhibits a potential to reduce CAPEX and OPEX significantly, by deploying software-based network functions in virtual machines (VMs) on edge-clouds. We consider a fundamental problem of NFV-enabled multicasting in a mobile edge cloud, where each multicast request has both service function chain and end-to-end delay requirements. Specifically, each multicast request requires chaining of a sequence of network functions (referred to as a service function chain) from a source to a set of destinations within specified end-to-end delay requirements. We devise an approximation algorithm with a provable approximation ratio for a single multicast request admission if its delay requirement is negligible; otherwise, we propose an efficient heuristic. Furthermore, we also consider admissions of a given set of the delay-aware NFV-enabled multicast requests, for which we devise an efficient heuristic such that the system throughput is maximized, while the implementation cost of admitted requests is minimized. We finally evaluate the performance of the proposed algorithms in a real test-bed, and experimental results show that our algorithms outperform other similar approaches reported in literature.

Index Terms—Mobile edge clouds, network function virtualization, multicasting, approximation algorithms, algorithm design.

1 Introduction

With increasing uptake and use of multimedia technologies, there is an associated increase in data being generated and transmitted (processed) over our network-based systems, often to multiple subscribers. Applications can include video-on-demand, high definition streaming, multimedia social networks (combing text, audio and video) and Internet-of-Things (IoTs). This paradigm of data transfer to multiple concurrent subscribers is referred to as *multicasting*, and can significantly stress our current networks. Multicasting not only requires use of various network functions such as firewalls, Intrusion Detection Systems (IDSs), proxies, and Wide Area Networks (WAN) optimizers to guarantee data transfer security, but also to meet stringent Quality-of-Service (QoS) requirements to ensure that the traffic is transferred on time. Considering that most multimedia data needs to be multicast to mobile users, Mobile Edge-Cloud Computing (MEC) [6], [15], [16], [18], [23], [27], [33], [50], [46], [47] has emerged as a promising platform to meet the QoS requirements of mobile users, by deploying data processing resources within the proximity of mobile users. Network Function Virtualization (NFV) moves network functions from dedicated hardware to (software-based) virtual machines (VMs) that can run on commodity hardware, thereby reducing the OPEX and CAPEX of network service providers. In this paper, we consider NFV-enabled multicasting in an MEC network, where each user request requires its traffic to pass through a sequence of network functions, referred to as a *service function chain*, before reaching its destination.

Provisioning NFV-enabled multicasting services in MEC networks poses many challenges. First, each cloudlet (resource hosting a software-based Virtual Network Function (VNF)) in an MEC network usually has limited computing resource to support VNFs. Allowing multicast requests to share existing VNF instances can significantly improve resource utilization in MEC networks and reduce service cost. It is however challenging to efficiently utilize existing VNF instances or create new VNF instances to maximize the number of multicast requests and minimize overall cost – subject to the computing capacity constraint on each cloudlet in the MEC network and the end-to-end delay

- *H. Ren, Z. Xu, and G. Wu are with the School of Software, Q. Xia is with the International School of Information Science and Engineering, Dalian University of Technology, and the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, China. E-mails: { z.xu, qiufenxia, wgwdut}@dlut.edu.cn and renhaozhe@mail.dlut.edu.cn.*
- *W. Liang is with Research School of Computer Science, the Australian National University, Canberra, ACT 2601, Australia. E-mail: wliang@cs.anu.edu.au*
- *P. Zhou is with the School of Cyber Science and Engineering, Huazhong University of Science and Technology, China. E-mail: panzhou@hust.edu.cn.*
- *O. F. Rana is with Cardiff University, United Kingdom. E-mail: RanaOF@cardiff.ac.uk.*
- *A. Galis is with University College London, United Kingdom. E-mail: a.galis@ucl.ac.uk.*

Corresponding author: Zichuan Xu. Email: z.xu@dlut.edu.cn

requirement of each admitted multicast request. The key challenge is to identify which cloudlets should be used to host VNFs required within a multicast request service chain, i.e. which existing VNF instances can be used for which request? Second, each NFV-enabled multicast request usually has a QoS requirement to guarantee that its traffic reaches the destinations within the specified end-to-end delay requirement. Identifying how to meet the end-to-end delay requirement of each admitted NFV-enabled multicast request is challenging. In this paper, we tackle the aforementioned challenges, by investigating efficient methods that investigate VNF sharing, service chaining, and routing that can meet QoS requirements of NFV-enabled multicast requests in an MEC network.

There are extensive studies on multicasting in conventional networks or software-defined networks, which do not consider service function chain requirements [17], [18], [51]. These solutions however cannot be directly applied to NFV-enabled multicasting. There are also recent investigations on NFV-enabled multicasting. However, these approaches do not consider end-to-end delay requirements [39], and they assume that only one service instance is included in the service function chain [51], or that the VNFs in each service chain are consolidated into a single location [47], [45]. For example, Zhang *et al.* [51] investigated the NFV-enabled multicast problem by assuming that there are sufficient computing and bandwidth resources in a Software Defined Network (SDN) to accommodate a multicast request. Xu *et al.* [47] investigated the problem of NFV-enabled multicasting, by devising an approximation algorithm with a provable approximation ratio for realizing a single NFV-enabled multicast request and an online algorithm with a guaranteed competitive ratio for the online NFV-enabled multicasting problem. Ren *et al.* [39] investigated the NFV-enabled multicasting in an SDN, by assuming that the traffic of each multicast request can be processed by multiple instances of the VNFs in its service chain. These methods are likely to increase the cost/delay of implementing such multicast requests, as placing VNFs into multiple cloudlets can lead to a greater delay to form a service function chain and incur a higher cost.

To the best of our knowledge, we are the first to consider the problem of delay-sensitive NFV-enabled multicasting problem in an MEC network, by designing both approximation algorithms and efficient heuristics. The main contributions of this paper are as follows.

- We study the NFV-enabled multicasting problem in an MEC network, with an aim to minimize the implementation cost of the request while meeting its delay requirement.
- We propose an efficient heuristic for the NFV-enabled multicasting problem. We also devise the very first approximation algorithm with an approximation ratio, if the delay requirement is neglected.
- We also consider a set of NFV-enabled multicast request admissions with the aim to maximize the weighted system throughput. We also propose a heuristic for this problem.
- We evaluate the performance of the proposed algorithms through experimental simulations in synthetic networks and within a real test-bed. Experimental results demonstrate that the proposed algorithms outperform existing reported approaches.

The rest of the paper is organized as follows. Section 2 reviews 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, and proposes an efficient heuristic for the problem with delay requirements using the proposed approximation algorithm as a subroutine. Section 5 devises an efficient heuristic algorithm for the the NFV-enabled multicasting problem with resource constraints on cloudlets. Section 6 evaluates the performance of the proposed algorithms experimentally in a real test-bed, and Section 7 concludes the paper.

2 Related Work

Recently, traffic steering has re-gained much attention due to the challenges introduced by software defined networking and network function virtualization [5], [6], [15], [16], [18], [23], [27], [33], [50], [46], [47]. Unicasting is one of the primary focus of existing studies. For example, Moens *et al.* [33] focused on hybrid networks with both hardware and software network functions. Cziva *et al.* [7] addressed the problem of the placement of virtual functions by minimizing the total number of VNF instances. Yu *et al.* [29] investigated profit maximization associated with placing VNFs onto a set of locations, and considered the delay requirement of each unicast request. Xu *et al.* [45] studied the offloading problem of delay-sensitive tasks with network function requirements in an MEC network, by proposing efficient heuristics and an online algorithm with a competitive ratio. Xie *et al.* [44] investigated the VNF sharing problem with an aim to improve resource utilization, by finding a common link for a set of service chains, so that the deployed service chains can be shared by all users. Kiji *et al.* [19] proposed a virtual network function placement and routing algorithm for multicast requests with service chain requests, through merging multiple service paths (MSC-M). Although there exist studies that consider the delay requirements of user requests [22], [29], [45], they only considered unicast requests and their solutions cannot be applied to the NFV-enabled multicasting problem, which is a generalization of the NFV-enabled unicasting problem. Chen and Wu [5] devised algorithms for the VNF placement to minimize the cost of implementing NFV-enabled unicast requests by balancing set-up and bandwidth consumption costs.

There are studies on multicasting in conventional networks [2], [24], [25], [14], [34], [43]. Recently, with the emergence 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 [18], [17]. For example, Huang *et al.* [18] studied online multicasting in software-defined networks with both node and link capacity constraints. Huang *et al.* [17] 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 MEC networks, because they did not consider the service chain requirements of multicast requests.

Investigations on NFV-enabled multicasting include [1], [39], [41], [47], [49], [31], [30], [51]. For instance, Zhang *et al.* [51] investigated the NFV-enabled multicasting problem in an SDN without resource capacity constraints, assuming that data traffic of each multicast request can only be processed by one server. Xu *et al.* [47], [48] 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 for the problems are proposed. They however assumed that the VNFs in each service chain is consolidated into a single data center. Ma *et al.* [31], [30] proposed an online algorithm for the NFV-enabled multicasting problem without taking into account the end-to-end delay requirement. Soni *et al.* [41] 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 however did not consider end-to-end delay requirements of multicast requests. Alhussein *et al.* [1] devised exact solutions for the problem of joint VNF placement and routing for multicast requests in 5G core networks, such that the cost of provisioning NFV-enabled multicast services is minimized, by formulating the problem into a mixed integer linear program (MILP). The delay requirement of NFV-enabled requests has not been considered and the MILP-based exact solutions might not be scalable for large problem sizes. Yi *et al.* [49] considered delay requirements of the NFV-enabled multicasting problem; however VNF sharing is not explored. To guarantee scalability and solution quality, Ren *et al.* [39] proposed approximation algorithm with an approximation ratio for the problem of embedding a service graph that consists of VNF instances into a substrate network, by assuming that the traffic of each multicast request can be processed by multiple instances of the VNFs in its service chain. The delay requirement of multicast requests however is not considered in the study. Similarly, the delay requirement of multicast requests is not considered [31], [30], and the authors only consider a single multicast request.

3 PRELIMINARIES

In this section, we first introduce the system model, notation and key concepts. We then define the problem being considered more precisely.

3.1 System model

We consider a mobile edge cloud (MEC) network $G = (V, E)$ with a set V of switches, a set of cloudlets and a set E of links between switches and cloudlets. Each cloudlet is attached to a switch in V via an optical fiber, 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 that are within the proximity of mobile users. Due to space limitation of installing cooling equipment in those places, each cloudlet is usually equipped with (a small number of) servers and thus has computing resource capacity to implement VNF instances. We denote by C_v the computing capacity of the cloudlet attached to switch node $v \in V_{CL}$. In addition, transferring data through links in E incurs a communication latency. Let d_e be the delay associated with transmitting a unit of 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 MEC network G. Fig. 1 is an illustrative example of an MEC network.

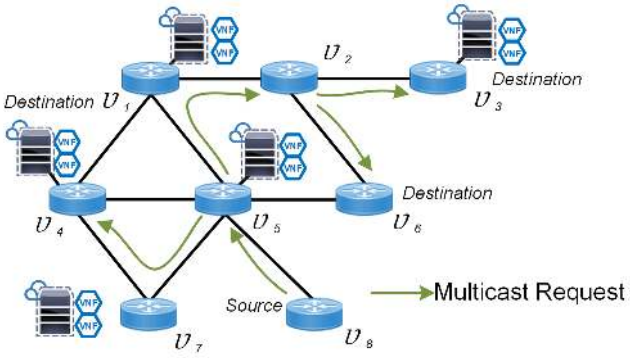


Fig. 1. An MEC network G.

3.2 NFV-enabled multicast requests and service chains

A delay-aware NFV-enabled multicast request is a request that transfers an amount of data traffic from a source to a set of destinations. The data traffic must be processed by a sequence of VNFs before reaching their destinations, while also meeting delay constraints.

Let r_k be a delay-aware NFV-enabled multicast request, 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 consider that the data traffic b_k of request r_k is given (derived from historical information).

Let $\mathcal{F}$ be the set of VNFs provided by the network service provider in G. A VNF $f_l \in \mathcal{F}$ can be needed by request r_k to form its service function chain SC_k. Assume that there are L_k VNFs 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 G. Due to the resource capacity constraints on cloudlets, we allow the instances of VNF f_l to be shared among different requests.

To admit request r_k, all data traffic from source s_k of r_k needs to be processed through an instance of each VNF $f_l \in SC_k$ prior to reaching destinations in D_k, as illustrated in Fig. 2. An existing instance must therefore 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. Existing or newly created VNF instances of each service chain 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 it may lack sufficient computing resources to create new instances for all VNFs in SC_k.

Each multicast request needs a certain amount of computing resource to process its data traffic. Let $C_{unit}(f_l)$ be the

number of computing resource needed to process a unit amount of data traffic. If f_l is implemented as a newly created instance, the total number of computing resources that should be assigned to the new instance to process the data traffic of request r_k 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 resources to meet the demand of r_k.

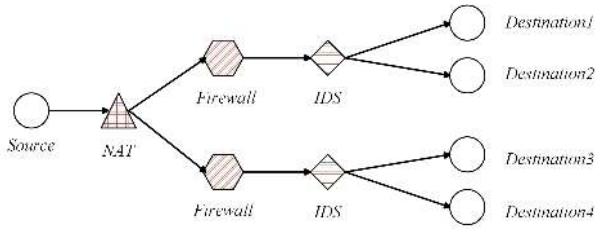


Fig. 2. A service chain ⟨ NAT, Firewall, IDS ⟩ with one instance of NAT and two instances of Firewall and IDS.

3.3 Delay requirements of multicast requests

The end-to-end delay of implementing a multicast request plays a vital role in guaranteeing the quality of services of users. We thus consider that each multicast request has a delay requirement, which specifies the maximum delay it can tolerate for transmitting its data from its specified source to its destinations. For a delay-aware NFV-enabled multicast request, its experienced delay 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 proportional 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.

The accumulative processing delay incurred due to the traffic processing by network functions in SC_k of r_k 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, where each path $p_m \in \mathcal{P}_k$ denotes a routing path from s_k to a 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 $\mathcal{P}_k$. We denote by d^t_k the transmission delay of request r_k, which can be defined as follows.

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

The delay experienced by multicast request r_k thus is

$$d_k = d^p_k + d^t_k, \tag{4}$$

which needs no greater than the specified delay requirement D_k, i.e.,

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

3.4 Cost models

As the network service provider of an MEC network G charges user requests on a pay-as-you-go basis, the major concern of the service provider is its *operational cost*, which 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. Denote by $n_{l,v}$ the number of existing instances of f_l in v that are used to process the traffic of r_k.

The operational cost of admitting a delay-aware NFV-enabled multicast request r_k can be specified as:

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

where V_{CL,r_k} is the set of cloudlets that are used to implement the instances of VNFs in SC_k of request r_k, and T_k is the obtained multicast tree that is used to route the data traffic of r_k.

3.5 The directed Steiner tree [4]

The Steiner tree problem is defined as follows: given a graph $G = (V, E)$ with a cost function c on the edges, and a subset of terminals $X \subset V$, the goal is to find a minimum cost tree that includes all the terminals in X. The found minimum cost tree is referred to as the *Steiner tree*.

3.6 Problem definition

We consider a mobile edge cloud (MEC) network $G = (V, E)$ with a set V_{CL} of cloudlets with $V_{CL} \subset V$, and a set of multicast requests R. Given a snapshot of the MEC at a given time instant and a NFV-enabled multicast request r_k, understanding how request r_k can be realised across a set of VNFs remains a key challenge. We thus first consider the problem of admitting a single multicast request r_k, such that its operational cost is minimized. Further, considering that the accumulated computing resources in an MEC may be insufficient to implement all requests, another question is identifying how to carry out admission control for multicast requests to maximize weighted throughput. In the following we define these two optimization problems precisely.

Problem 1: Assuming that each multicast request can be implemented using the computing resources assigned to existing VNF instances, the *NFV-enabled multicasting problem with a single multicast request* in MEC network 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, such that the operational cost (i.e., Eq.(6)) of implementing r_k is minimized, while meeting the end-to-end delay requirement D_k of r_k and capacity constraint on each cloudlet $v \in V_{CL}$.

Problem 2: Assuming that the computing resource in each cloudlet in the MEC network G has available capacity. For each request in R, the network may or may not have enough resources to admit it, the *NFV-enabled multicasting problem* in an MEC network G for a given set R of NFV-enabled multicast requests is to maximize the system throughput while minimizing the operational cost, subject to computing capacity on each cloudlet, where the system throughput is defined as the total amount of data that is processed and transferred by the system for admitted multicast requests. Let ST be the weighted throughput and R_{ad} the set of admitted multicast requests, then

$$ST = \sum_{r_k \in R_{ad}} b_k. \tag{7}$$

The NFV-enabled multicasting problems are NP-hard, as its special case – the traditional multicast problem without NFV service chain constraints is NP-hard [8].

For clarity, the symbols used in this paper are summarized in Table 1.

4 Algorithms for the Admission of a Single NFV-enabled Multicast Request

In this section, we deal with NFV-enabled multicasting for a single NFV-enabled multicast request admission. We first propose an efficient heuristic for the problem. We then consider a special case of the problem without delay requirements, by devising an approximation algorithm.

4.1 An efficient heuristic

The basic idea of the proposed heuristic is based on an observation that a feasible solution to the problem needs to meet the capacity constraints on cloudlets, service function chain requirements, and the end-to-end delay requirement of each multicast requests r_k. We thus adopt a two-phase heuristic that progressively considers the mentioned constraints and requirements.

Phase one: we first propose an algorithm to jointly consider the capacity constraint and the service chain requirement, by ignoring the delay requirement of r_k. The proposed algorithm smartly explores existing VNF instances in each cloudlet that can be shared with the VNF instances of r_k. Notice that the solution may not be feasible to the NFV-enabled multicasting problem, because the delay requirement of r_k is not considered in this phase. For the sake of clarity, we describe the proposed algorithm for the problem without delay requirement in the next subsection, which is referred to as `Appro_NoDelay`. By now, we assume we already obtained the multicast tree for r_k in G without considering its delay requirement.

Phase two: we refine the obtained multicast tree into a feasible solution to meet the delay requirement of r_k. In particular, we observe that a longer delay will be the result if the VNFs of SC_k are implemented in multiple cloudlets. This is because that if the VNFs are distributed into different cloudlets, the data traffic transmission among two consecutive VNFs has to be performed by inter-cloudlet links, which incurs higher delays than those by intra-cloudlet data transfers. 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 r_k. This means that a large or a small value for the number of cloudlets of a request may not be proper to meet the delay requirement of r_k. We thus adopt a binary search to narrow down the choices of the proper number of cloudlets for r_k, making the delay requirement of r_k being met quickly. 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 proper number of cloudlets in the feasible solution. We first set

$$n_k = \Big\lfloor \frac{|V_{CL}| + 1}{2} \Big\rfloor. \tag{8}$$

The proposed algorithm first tries to re-assign the VNFs in service function chain 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 [4] (i.e., multicast 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 than its requirement, we continue the above procedure by searching the appropriate number of cloudlets in the range of $[1, n_k]$. The rationale is that the number of cloudlets in the multicast tree is still too many, and the inter-cloudlet communication leads to the delay requirement violation. The number of cloudlets still needs to be reduced. 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. The detailed heuristic is described in Algorithm 1 and its basic idea is shown in Fig. 3. For simplicity, this algorithm is referred to as algorithm `Heu_Delay` in the rest of this paper.

4.2 An approximation algorithm for the problem without delay requirements

The proposed approximation algorithm for the problem without delay requirements is to reduce the problem in 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 function chain SC_k can be implemented in multiple cloudlets or consolidated into a single cloudlet

TABLE 1
Symbols

Symbols	Meaning
$G = (V, E)$	a mobile edge cloud (MEC) network with a set V of switches and a set E of links
R	a set of delay-aware NFV-enabled multicast requests
V_{CL}	the set of switches with attached cloudlets, and clearly $V_{CL} \subseteq V$
C_v	the computing capacity of the cloudlet attached to a switch node $v \in V_{CL}$
$r_k = (s_k, D_k; b_k, SC_k)$	a delay-aware NFV-enabled multicast request, 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.
$\mathcal{F}$ and f_l	the set of VNFs provided by the network service provider in G and a VNF f_l
L_k	The number of VNFs in SC_k
$C_{unit}(f_l)$	the amount of computing resource needed to process a unit amount of data traffic
$d^p_{k,l}$	the processing delay of each multicast request r_k by VNF f_l
α_l	a given proportional factor of VNF f_l
d^p_k	the accumulative processing delay incurred due to the traffic processing by network functions in SC_k of r_k
$\mathcal{P}_k$	the set of routing paths from source s_k to destinations in D_k
$p_m \in \mathcal{P}_k$	a routing path from s_k to a destination $t_m \in D_k$
d^t_k	the transmission delay of request r_k
d_k and D_k	the delay experienced by multicast request r_k and its delay requirement
$c(e)$ and $c(v)$	the usage costs of one unit of bandwidth and computing resources at link $e \in E$ and cloudlet $v \in V_{CL}$
$c_l(v)$	the cost of instantiating an instance of network function f_l in cloudlet $v \in V_{CL}$
$n'_{l,v}$	the number of newly created instances for network function f_l in cloudlet v
$n_{l,v}$	the number of existing instances of f_l in v that are used to process the traffic of r_k
c_k	the operational cost of admitting a delay-aware NFV-enabled multicast request r_k
V_{CL,r_k}	the set of cloudlets that are used to implement the instances of VNFs in SC_k of request r_k
T_k	the obtained multicast tree that is used to route the data traffic of r_k
ST and R_{ad}	the weighted throughput and the set of admitted multicast requests
n'_k	the number of cloudlets that are used to implement the VNFs in SC_k in the current infeasible solution
n_k	the proper number of cloudlets in the feasible solution of algorithm `Heu_Delay`
n_{max} and n_{min}	the minimum and maximum bounds of the binary search range in algorithm `Heu_Delay`
$G' = (V', E')$	the auxiliary graph constructed in algorithm `Appro_NoDelay`
$f'_{i,l,v}$ and $f''_{i,l,v}$	the pair of virtual VNF nodes for the ith VNF instance of f_l in cloudlet $v \in V_{CL}$
$w(f'_{i,l,v}, f''_{i,l,v})$	the weight of edge $\langle f'_{i,l,v}, f''_{i,l,v} \rangle$ in the auxiliary graph G'.
$v'_{k,l}$ and $v''_{k,l}$	a pair of virtual cloudlets for the lth VNF and cloudlet v in G'
$W_{l,v}$	the widget that is built for network function f_l in cloudlet $v \in V_{CL}$
$ws_{l,v}$ and $wd_{l,v}$	a widget source node and a widget destination node for the widget for network function f_l and cloudlet $v \in V_{CL}$ in auxiliary graph G'
c^*	the optimal solution for the NFV-enabled multicasting problem
L_{max}	the maximum length of the service chains of the requests in R, i.e., $L_{max} = \arg\max_{r_k \in R} \lvert SC_k \rvert$.
L_{com} and $R(L_{com})$	the number of common VNFs that requests have in their service chains VNFs in common of their service chains, and the set of such requests.

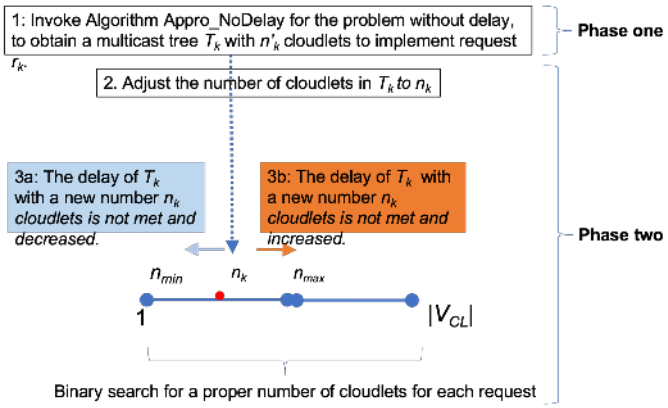


Fig. 3. An illustration of the algorithm `Heu_Delay`.

to save the communication cost due to the transmissions between different cloudlets. To ensure 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 G' based on the sub-network of G.

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.

Recall that VNFs in SC_k of multicast request r_k can be assigned to existing VNFs or newly instantiated VNF instances. To determine whether making use of existing VNF instances or creating new ones, we create a *widget* for each cloudlet $v \in V_{CL}$ and each 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. There is a widget for each

Algorithm 1 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.

/*Phase one: find cloudlets and routing paths for r_k by considering its service chaining requirement and cloudlet capacity constraints.*/
Find a multicast tree for r_k without considering its delay requirement d_k^{req}, by invoking algorithm Appro_NoDelay;
Let n'_k be the number of cloudlets that are used to implement VNFs in SC_k of the found multicast tree;
/*Phase two: adjust the multicast tree to meet the delay requirement of r_k.*/
$n_{min} \leftarrow 1$;
$n_{max} \leftarrow |V_{CL}|$;
while $n_{min} <= n_{max}$ **do**
  $n_k \leftarrow \lfloor \frac{n_{min}+n_{max}}{2} \rfloor$;
  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.
  if the experienced delay of r_k is met **then**
    return;
  else
    if the experienced delay of r_k is decreased **then**
      $n_{max} \leftarrow n_k$;
    else
      $n_{min} \leftarrow n_k$;

pair of cloudlet and VNF, which actually means a possible placement of a VNF to a cloudlet.

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. 4. Let $v'_{k,l}$ and $v''_{k,l}$ be such a pair of virtual cloudlets for the lth VNF and cloudlet v. To jointly consider the processing and transmission costs, 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 become part of the auxiliary graph G'.

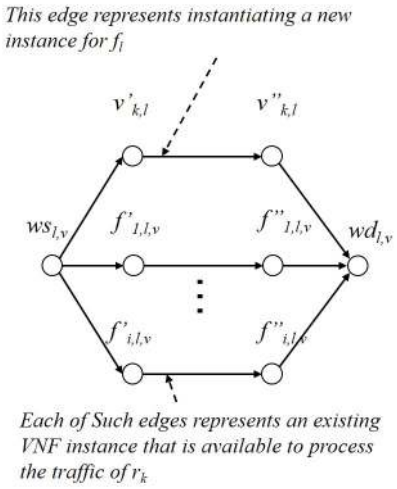


Fig. 4. An example of the widget for the VNF f_l in SC_k and cloudlet $v \in V_{CL}$

We then connect the 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 $1 \leq l \leq L_k - 1$, i.e., $E' \leftarrow E' \cup \{\langle wd_{l,v}, ws_{l+1,u} \rangle\}$ for l with $1 \leq l \leq L_k - 1$ 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. 5.

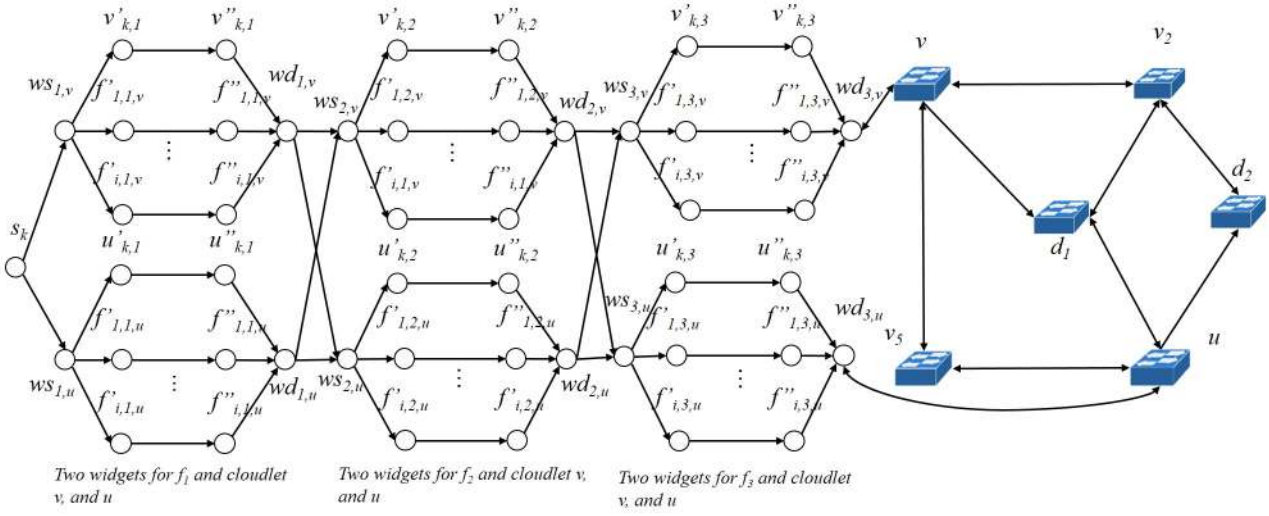


Fig. 5. An example of the auxiliary graph $G' = (V', E')$ with two servers attached at node v and node u and multicast request r_k transfer its data to destinations in $D_k = \{d_1, d_2\}$. Note that there is a widget for each pair of VNF f_l and cloudlet v, corresponding to a possible assignment of f_l. The original switches that attach the two cloudlets will just serve as normal forwarding switches.

Problem reduction We now reduce the NFV-enabled multicasting problem without delay requirements in G to the Steiner tree problem in the directed 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 directed Steiner tree in G' that spans nodes in $\{s_k\} \cup D_k$. We then transfer the Steiner tree in 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 VNF 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 the widgets in G' correspond to the shortest paths of their endpoints of the edges in G. We thus replace each of such edges with its shortest path in G.

Algorithm 2 `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. 5;
2: Find a directed Steiner tree T in G' that spans nodes in $\{s_k\} \cup D_k$, using Charikar's algorithm [4];
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. */

4.3 Algorithm analysis

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

We first show the feasibility of the solution delivered by algorithm 2. Intuitively, if a solution to the NFV-enabled multicasting problem, it needs to satisfy the following three 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 Following the construction of G', showing the feasibility of the solution is to show that if the Steiner tree found in G' has one path from $ws_{l,v}$ to $wd_{l,v}$ of each selected widget, 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) 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'. 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 stand alone 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. 6. Paths p' and p however make it not a tree. Therefore, only one path from $ws_{l,v}$ to

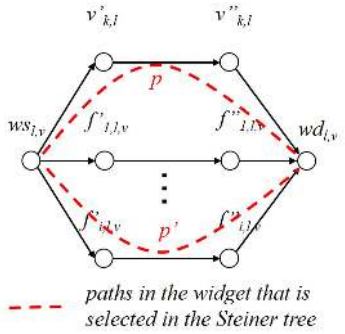


Fig. 6. A widget and its paths from its source to destination nodes that are selected in the Steiner tree.

$wd_{l,v}$ will be included in 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 consider 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 stand alone 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.

Case (2) can be dealt with similarly. Therefore, these two cases are not possible according to the construction of G'.

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 an MEC network $G = (V, E)$ 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** 2, for a special case of the NFV-enabled multicasting problem without delay requirements, which delivers a feasible solution that has an approximation ratio of $i(i-1)|D_k|^{1/i}$ [4], and the time complexity of $O((L_k \cdot |V| \cdot \frac{C_v}{C_{unit}(f_l)} + |V|)^i \cdot |D_k|^{2i})$, 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 [4].

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

Assume c^* is the optimal solution for the NFV-enabled multicasting problem. In **Algorithm** 2, 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 [4] is $i(i-1)|D_k|^{1/i}$, the approximation of **Algorithm** 2 is $i(i-1)|D_k|^{1/i}$ as well.

We now show the time complexity of **Algorithm** 2. 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 $O(|V'|^3)$ [21]. 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 finally analyze the performance of Algorithm 1 the following theorem.

Theorem 2. Given an MEC network $G = (V, E)$ 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** 1, 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 [4].

Proof We first show the solution feasibility of the proposed heuristic by showing that the end-to-end delay requirement of r_k is met. Algorithm 1 adopts a binary search based heuristic to find the proper number of cloudlets 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 analyze 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 Algorithm for Admissions of a Set of NFV-enabled Multicasting Requests

In this section, we consider a set of multicast request admissions. Given a set of NFV-enabled multicast request, we admit as many as requests in the set such that the weighted system throughput is maximized, while the accumulated implementation cost of all admitted requests is minimized, subject to computing capacities on cloudlets in an MEC.

5.1 Overview

Recall that we proposed both approximate and heuristic solutions for the NFV-enabled multicasting problem for the admission of a single multicast request, a simple method for the NFV-enabled multicasting problem is to consider algorithm `Heu_Delay` as a black-box and admit each request one by one invoking algorithm `Heu_Delay` iteratively. This method however may miss the opportunities of sharing VNFs among the requests, if the consecutively admitted requests do not have common VNFs in their service chains. Further, the constructed auxiliary graph G' in algorithm `Heu_Delay` for a request may no longer useful for the other. This consequently may lead to a prohibitively long time to make decisions of request admissions.

The basic idea behind the proposed algorithm is as follows. We observe that some requests have the same service chain requirements, and the VNFs in their service chains can be shared with high opportunities. Fig. 7 illustrates this idea, from which we can see that requests are classified into different *categories*, with each category having a set of requests that share a number of VNFs. Specifically, the algorithm first considers the category in which multicast requests the maximum number of common VNFs of their service chains. Then, the requests in this category, we start with the requests with smaller data traffic, and admit the requests one-by-one. This procedure continues until no more requests can be admitted in the category.

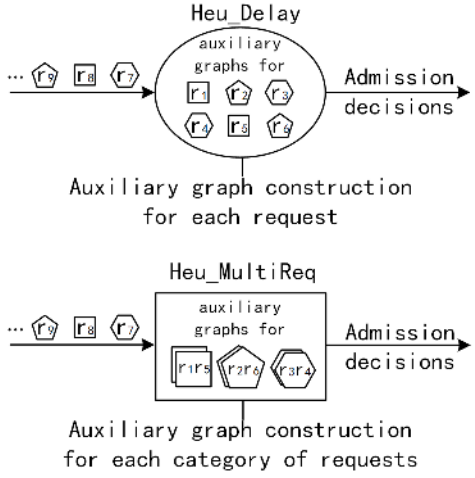


Fig. 7. The basic idea of the proposed heuristic for the NFV-enabled multicasting problem.

5.2 Heuristic algorithm

We propose an efficient heuristic for the NFV-enabled multicasting problem for a set of requests with different service chain requirements, based on **Algorithm** 1.

Specifically, the heuristic consists of a number of iterations within each iteration, a set of requests with the same number of VNFs in common are processed. Let L_{com} be the number of common VNFs that requests have in their service chains. Let L_{max} be the maximum length of the service chains of the requests in R, i.e., $L_{max} = \arg\max_{r_k \in R} |SC_k|$. Initially, $L_{com} = L_{max}$. It decreases by one in each iteration of the algorithm until $L_{com} = 0$.

Within each iteration, we first find the requests that have L_{com} VNFs in common of their service chains. Denote by $R(L_{com})$ the set of such requests. We then rank the requests in $R(L_{com})$ in increasing order of their data traffic. For each request $r_k \in R(L_{com})$, we invoke the proposed approximation algorithm in 2. Notice that the requests in $R(L_{com})$ may have different source nodes and different destination sets. We thus need to adjust the auxiliary graph after the admission of each multicast request, by removing the source node for the previous request, and add the source node of the current request. This means that, before admitting the next multicast request r_{k+1}, we make adjustments of the constructed auxiliary graph G' instead of constructing a new one. Specifically, the widgets that are built for the L_{com} VNFs are updated accordingly, if multicast request r_k is admitted. Also, the widgets for the VNFs that are not among the L_{com} of request r_{k+1} is added to the auxiliary graph. This iteration continues until no more requests can be admitted within this category. The steps of this algorithm are detailed in Algorithm 3.

Algorithm 3 `Heu_MultiReq`

Input: $G = (V, E)$, V_{CL}, C_e for each $e \in E$, C_v for each $v \in V_{CL}$, and a set of multicast requests with each multicast request being denoted by $r_k = (s_k, D_k; b_k, SC_k)$.

Output: The system throughput achieved by the admitted requests in R.

$N_{ad} \leftarrow 0$;
for $L_{com} \leftarrow 0, 1..., L_{max}$ **do**
  Find the maximum number of requests in R that have L_{com} common VNFs in their service chains, and let $R(L_{com})$ be the set of requests;
  Rank the multicast requests in $R(L_{com})$ according to their data traffic;
  for each request $r_k \in R(L_{com})$ **do**
    $T \leftarrow \emptyset$;
    while G is (s_k-D_k)-connected OR r_k is admitted **do**
      Construct auxiliary graph $G' = (V', E')$, by creating $L_k \cdot |V_{CL}|$ widgets, adding all the switch nodes in V of the original network G, and interconnecting the added nodes as shown in Fig. 5, or adjust the auxiliary graph if it is already constructed in the admission of previous requests;
      Find a Steiner tree T for in auxiliary graph G';
      if the delay of each branch of T is smaller than d_k^{req} **then**
        Admit multicast request r_k;
      else
        Find the branches of T that violate delay requirement d_k^{req};
        For each of such found branch, identify an edge with the maximum delay;
        Remove the identified edges from graph G;
    if $T \neq \emptyset$ **then**
      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;
    The widgets that are built for the L_{com} VNFs are updated according to the resource availabilities after admitting r_k;
    if $k + 1 < |R(L_{com})|$ **then**
      The widgets for the VNFs that are not among the L_{com} of request r_{k+1} is added to the auxiliary graph;

We now analyze the feasibility of **Algorithm** `Heu_MultiReq` in the following theorem.

Theorem 3. Given an MEC network $G = (V, E)$ with a set V_{CL} of cloudlets, a set R of NFV-enabled multicast requests with each 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 an efficient algorithm, Algorithm 3, for the NFV-enabled multicasting problem.

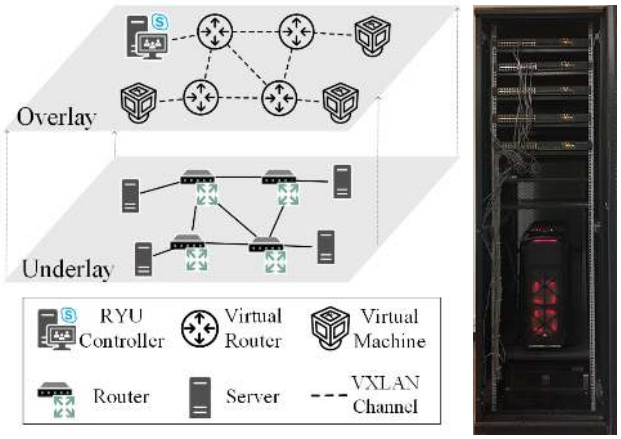
Proof To show the solution delivered by algorithm 3 is feasible, we need to show the classification of requests does not affect the solution feasibility of algorithm 2. Assume that the algorithm currently considers request r_{k+1}. If its previous request r_k is admitted, the widgets of the corresponding cloudlets that implement the VNFs of r_k are then updated, since the resource availabilities of these cloudlets or statuses of their existing VNF instances changed. Otherwise, there is not any change of the widgets in the auxiliary graph. Considering that the feasibility of admitting one request by Algorithm 2 can be shown by Lemma 2, Algorithm 1 delivers a feasible solution when multiple requests are considered.

6 Performance Evaluation

In this section we evaluate the performance of the proposed algorithms in a real testbed.

6.1 Test-bed setup

We build a test-bed consisting of both an underlay network with hardware switches and an overlay network with virtual switches, as shown in Fig. 8 The physical underlay consists of five H3C S5560X-30S-EI switches [12], with the support for VXLAN for virtual tunnel building and SDN capabilities. It has also one server with E5 Gold 5218 CPU, 128G RAM and four PCs with i7-8700 CPU, 16G RAM. Netconf and SNMP protocols are used to manage the switches and the links that interconnect them [35], [3]. We considered a design approach that uses the VXLAN functionality provided by the switch, where VXLAN is a widely used overlay technology [37]. The H3C S5560X-30S-EI switch implements a VXLAN tunnel based on hardware, which can greatly improve performance compared to traditional methods. The overlay mechanism provides connectivity within, and potentially across multiple testbed sites as it can transit any routed layer-3 underlay. We use VXLAN as a point-to-point tunneling mechanism (VXLAN VNI identifies a single link between two nodes [37]). SDN-capable switches can also perform encapsulation and decapsulation of VXLAN tunnels, each tunnel corresponds to a port in the switch. Using VXLAN, we build an overlay network with a number of Open vSwitch (OVS) [36] nodes and VMs. The overlay network is built following the topology generated using a graph generation tool GT-ITM [10] and the real network topologies AS1755, AS4755. Its OVS nodes and VMs are controlled by a Ryu [40] controller. The proposed algorithms are implemented as Ryu applications.

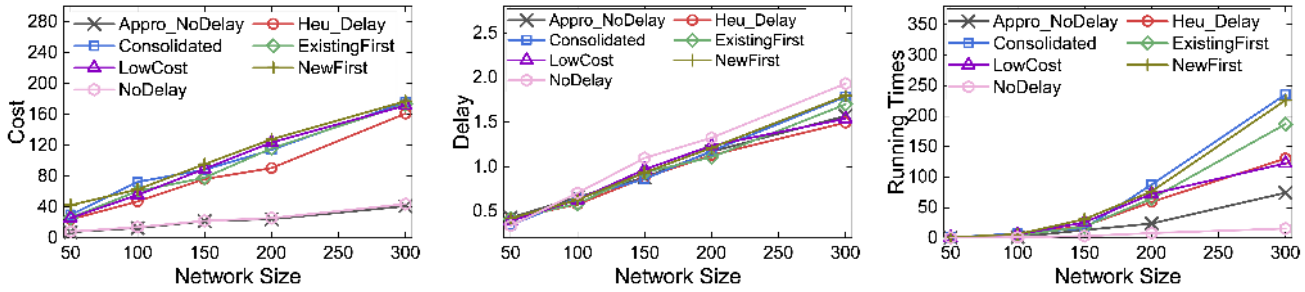


(a) The underlay and overlay of the test-bed. (b) The hardware switches and servers.

Fig. 8. A test-bed with both hardware switches and virtual resources.

6.2 Environment settings

We consider an MEC network consisting of the number of nodes from 50 to 250. The number of servers in each network



(a) Average cost of implementing a multicast request. (b) Average delay experienced by a multicast request. (c) Running times.

Fig. 9. The performance of algorithms `Appro_NoDelay`, `Consolidated`, `NoDelay`, `ExistingFirst`, `NewFirst`, and `LowCost`.

is set to 10% of the network size, and the servers are randomly co-located with the switches. We also use real network topologies, i.e.,the GÉANT [9] and an ISP network from [42]. There are nine cloudlets for the GÉANT topology as set in [11] and the number of data centers in the ISP networks are provided by [38]. The computing capacity of cloudlet varies from 40,000 to 120,000 MHz [13] (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 [11], [32]. 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. Unless otherwise specified, these parameters will be adopted in the default setting.

We compare the performance of the proposed approximation and heuristic algorithms against the following benchmarks.

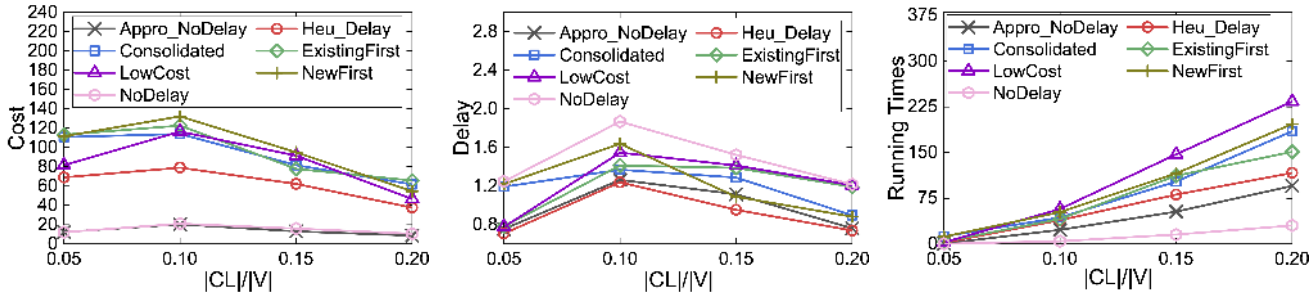
- We consider the case where the VNFs of each multicast request may be placed to multiple cloudlets for processing while there exist solutions that consolidate all VNFs of a multicast request into a single location. We thus compare our solutions with such a solution, which is referred to as algorithm `Consolidated`.
- We evaluate the performance of the proposed approximation and heuristic algorithms against the one in [39] 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 the 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 closest cloudlet. The procedure continues until all VNFs in SC_k are considered. This greedy algorithm 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 the closest to the source s_k and then places 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 have not been assigned, it finds the next cloudlet that is the closest to the found cloudlets.

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

We first evaluate the performance of algorithms `Heu_Delay` and `Appro_NoDelay` against that of algorithms `Consolidated`, `NoDelay`, `ExistingFirst`, `NewFirst`, and `LowCost`, in terms of the average operational cost, the average end-to-end delay, and the running time, by varying the network size from 50 to 250 while fixing the number of requests at 100. Fig. 9 shows the results of the proposed algorithms.

From Fig. 9 (a), we can see that Algorithm `Heu_Delay` achieves a lower operational cost than these of algorithms `ExistingFirst`, `NewFirst`, and `LowCost`. The reason is that Algorithm `Heu_Delay` jointly considers existing VNF instances and newly instantiated ones. However, the greedy approaches `NewFirst`, `ExistingFirst`, and `LowCost` only prefer new, existing, or low processing cost VNF instances. They unfortunately could miss the opportunities of further reducing the operational cost. Specifically, if the use of existing VNF instances can save the processing cost, `NewFirst` has a higher cost due to creating new instances. Also, there are some cases when creating new VNF instances can save transmission costs, which can be missed by algorithm `ExistingFirst`. In addition, it can be seen from Fig. 9 (a) that Algorithm `Heu_Delay` has a higher operational cost than algorithms `Appro_NoDelay` and `NoDelay`. This is because algorithms `Appro_NoDelay` and `NoDelay` do not consider the delay requirement of requests, making it choose cloudlets with lower operational costs.

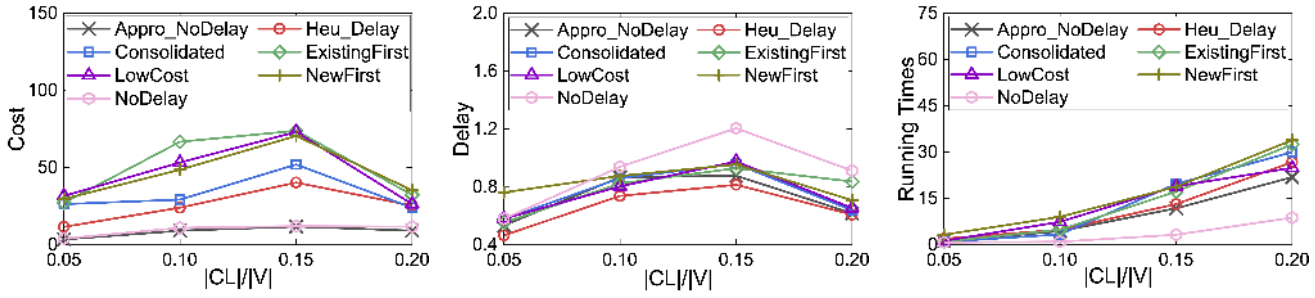
As shown in Fig. 9 (b), the average delay experienced by each multicast request by Algorithm `Heu_Delay` is



(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.

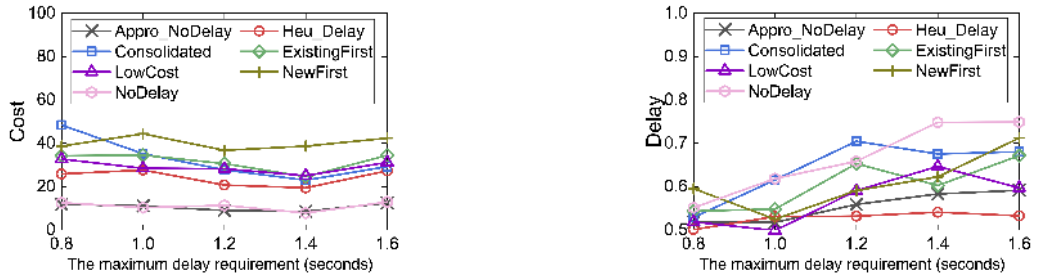


(d) Average cost of implementing a multicast request in network AS4755.

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

(f) Running times in network AS4755.

Fig. 10. The performance of algorithms `Appro_NoDelay`, `Consolidated`, `NoDelay`, `ExistingFirst`, `NewFirst`, and `LowCost` in networks AS1755 and AS4755.



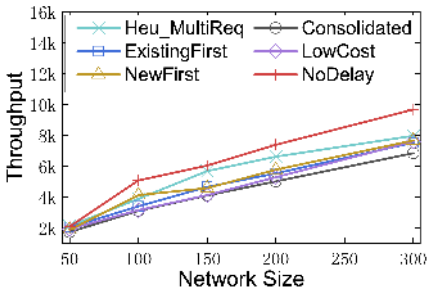
(a) Average cost of implementing a multicast request.

(b) Average delay experienced by a multicast request.

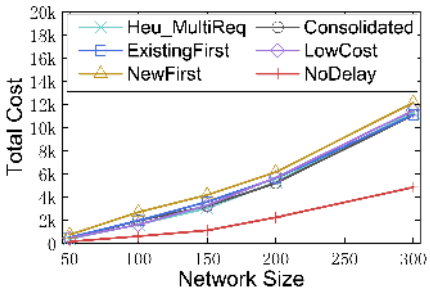
Fig. 11. The impact of the maximum delay requirement of each multicast request on the performance of algorithms `Appro_NoDelay`, `Consolidated`, `NoDelay`, `ExistingFirst`, `NewFirst`, and `LowCost`.

much lower than its comparison counterparts. The reason is that Algorithm `Heu_Delay` carefully finds a trade-off between the delay and cost of implementing a NFV-enabled request. Also, from Fig. 9 (c), we can see that the running time of Algorithm `Heu_Delay` is around 50 seconds for network size 200, which is slightly larger than those of algorithms `Appro_NoDelay` and `NoDelay` and smaller than algorithms `ExistingFirst`, `NewFirst`, and `LowCost`. The reason is that `Heu_Delay` has an additional process of binary search to find a proper number of cloudlets for each request r_k. Algorithm `NoDelay` has a lower running time compared with algorithm `Appro_Delay` because the delay requirement of requests is not considered, which reduces the solution space.

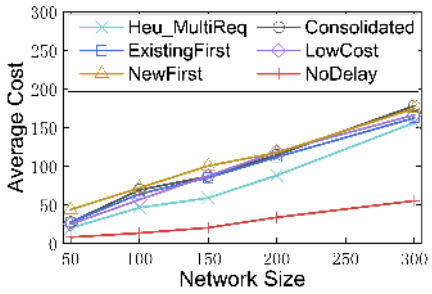
We then evaluate the performance of algorithms `Heu_Delay` and `Appro_NoDelay` against that of algorithms `Consolidated`, `NoDelay`, `ExistingFirst`, `NewFirst`, and `LowCost`, in real networks AS1755 and AS4755, by varying the ratio of the number of cloudlets to the number of switches, i.e., $|\mathcal{CL}|/|V|$ from 0.05 to 0.2. Fig. 10 illustrates the results. Fig. 10(a) and (d) show 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 delay. We can also see that the average cost of implementing a multicast increases first when the ratio $|\mathcal{CL}|/|V|$ increases from 0.05 to 0.1 and then decreases afterwards. The rationale behind is that VNFs of each multicast request may be assigned to more cloudlets with the increase of number of cloudlets, thereby pushing up 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 that these cloudlets are deployed



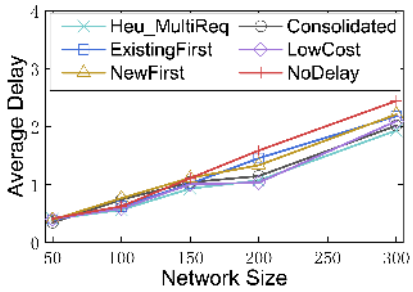
(a) System throughput.



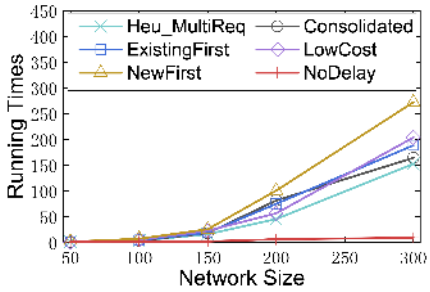
(b) Total cost of implementing a multicast request.



(c) Average cost of implementing a multicast request.

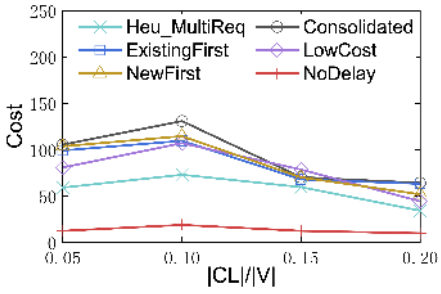


(d) Average delay experienced by a multicast request.

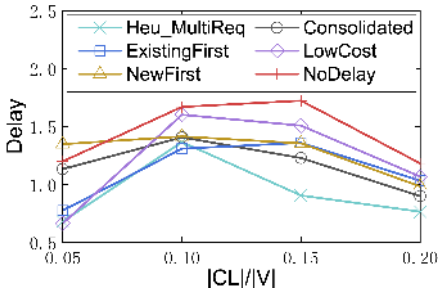


(e) Running times.

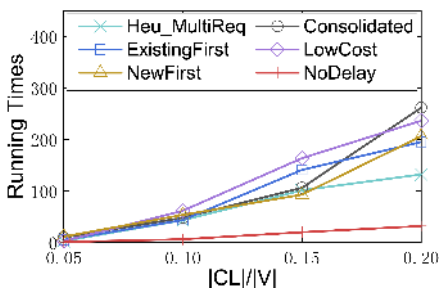
Fig. 12. The performance of algorithms `Heu_Multicast`, `Consolidated`, `NoDelay`, `ExistingFirst`, `NewFirst`, and `LowCost`.



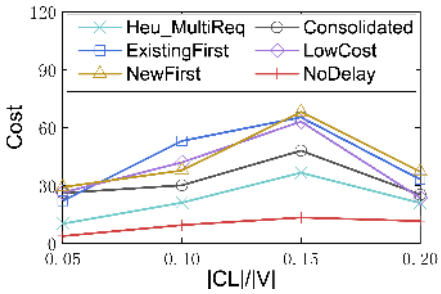
(a) Average cost of implementing a multicast request.



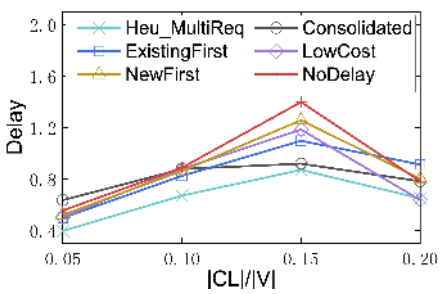
(b) Average delay experienced by a multicast request.



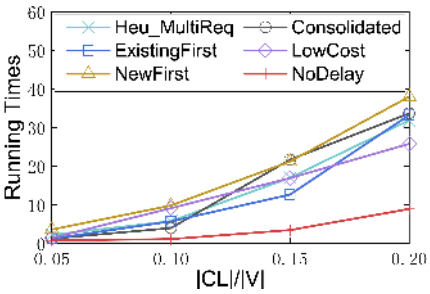
(c) Running times.



(d) Average cost of implementing a multicast request.



(e) Average delay experienced by a multicast request.

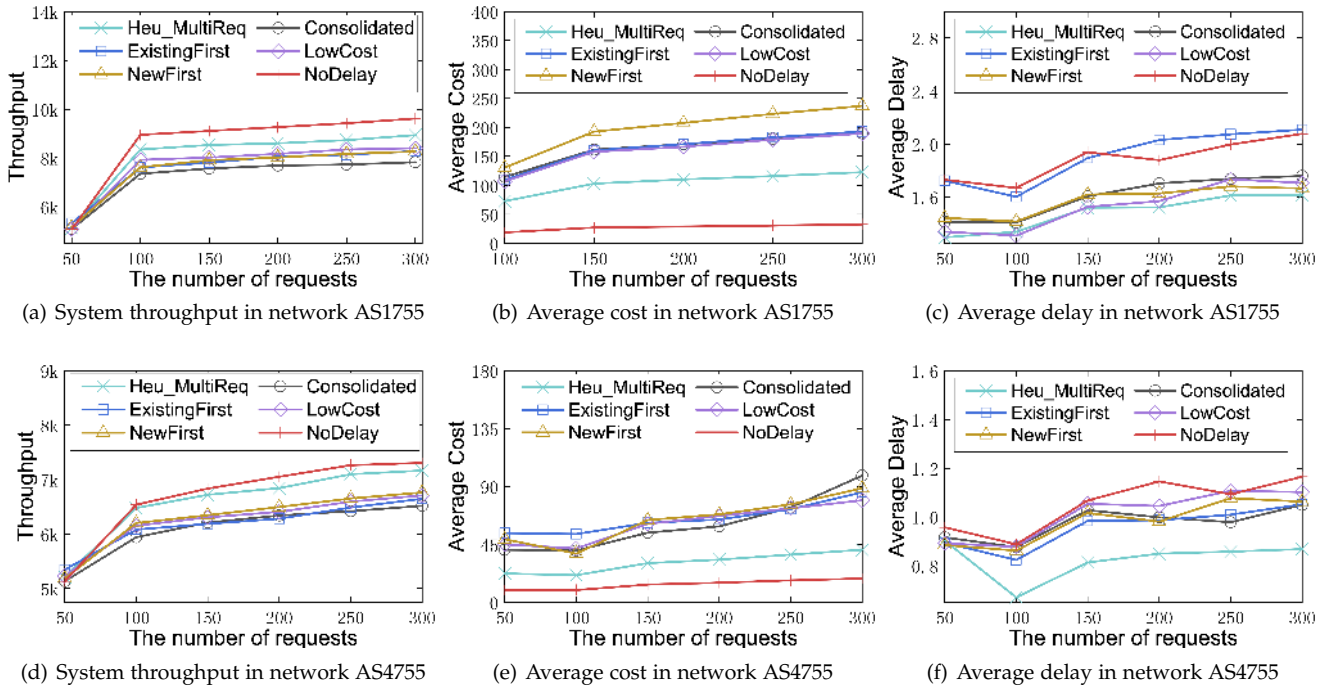


(f) Running times.

Fig. 13. The performance of algorithms `Heu_Multicast`, `Consolidated`, `NoDelay`, `ExistingFirst`, `NewFirst`, and `LowCost`.

in locations that are close to the source and destinations of the multicast request. The transmission cost then can be reduced afterwards.

We then investigate the impact of the maximum delay requirement on algorithm performance in the real network AS1755, by varying the maximum delay requirement of each multicast request from 0.8 seconds to 1.8 seconds with an increment of 0.2 seconds. Fig. 11 illustrates that the cost of implementing a multicast request is decreasing with the increase of the maximum delay requirement. The rationale



(a) System throughput in network AS1755 (b) Average cost in network AS1755 (c) Average delay in network AS1755

(d) System throughput in network AS4755 (e) Average cost in network AS4755 (f) Average delay in network AS4755

Fig. 14. The performance of algorithms `Heu_Multicast`, `Consolidated`, `NoDelay`, `ExistingFirst`, `NewFirst`, and `LowCost`.

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. 11.

6.4 Performance evaluation of algorithm `Heu_MultiReq`

We now compare the performance of Algorithm `Heu_MultiReq` against that of algorithms `Consolidated`, `NoDelay`, `ExistingFirst`, `NewFirst`, and `LowCost`, in terms of the system throughput, the total operational cost, the average end-to-end delay, and the running time, by varying the network size from 50 to 250 and fixing the number of requests to 100. Results are shown in Fig. 12, from which we can see that Algorithm `Heu_MultiReq` achieves around 30%, 30%, 35% higher system throughput than algorithms `ExistingFirst`, `NewFirst`, `LowCost`, and `Consolidated` when the network size is 200. The rationale behind is that algorithms `ExistingFirst`, `NewFirst`, and `LowCost` prefer existing, newly instantiated, and low processing cost VNF instances for each multicast request, and the cloudlets for those VNF instances may not have sufficient computing resource to implement the request, thereby leading to its rejection. Further, from figures 12 (a) and 12 (b), it can be seen Algorithm `NoDelay` has a higher end-to-end delay than that of Algorithm `Heu_MultiReq`, although it delivers a slight higher system throughput. Similar results can be observed from Fig. 13 when the performance of Algorithm `Heu_MultiReq` is evaluated against that of algorithms `Consolidated`, `NoDelay`, `ExistingFirst`, `NewFirst`, and `LowCost`, in real networks AS1755 and AS4755.

We then investigate the impact of the number of requests on the performance of algorithms `Heu_MultiReq`, `Consolidated`, `NoDelay`, `ExistingFirst`, `NewFirst`, and `LowCost`, in terms of system throughput, average operational cost, average end-to-end delay, and running time, by varying the number of requests from 50 to 300 while fixing the network size to 100. Fig. 14 shows that the system throughput increases first with the growth on the number of requests from 50 to 100, and then keeps stable afterwards, because the cloudlet capacities are saturated. We can also see that the average cost of implementing a multicast increases with the growth of request number. The rationale behind is that each multicast request may be assigned to more cloudlets for processing with the increase of number of requests, considering that the resources in cloudlets are saturated and may not be enough to implement all VNFs of a service chain. This eventually increases the transmission cost for each multicast request.

7 Conclusion and Future Work

In this paper, we study the problem of delay-aware, NFV-enabled multicasting in a mobile edge cloud network, by exploring the sharing of VNF instances of requests. If cloudlets have sufficient computing resource to process traffic of a multicast request, with no delay requirement, we proposed an approximate solution with a provable approximation ratio; otherwise, we developed an efficient heuristic. We also considered a set of NFV-enable multicast request admissions with the aim to maximize the weighted system throughput, for which we proposed an efficient heuristic. We finally evaluate the performance of the proposed algorithms against state-of-the-arts approaches in a real test-bed, and the results show that the performance of our algorithms is promising.

In this paper we considered the sharing of idle VNFs that have been released by other requests. The requests with the same service chain requirements may share resources with high probability. However, requests may have dynamic resource demands, and may share resources with others as long as they have complimentary demands. Understanding how to learn such dynamic complimentary resource demands among requests is challenging. Therefore, we consider the adoption of machine learning methods to classify requests with complimentary demands as our future research study – akin to existing efforts in *interference-aware* scheduling in cloud-based data centers. Existing efforts that make use of an interference index to characterize these competing/ complementary workloads can also be utilized in the proposed environment. Another is to explore the dynamic admission of NFV-enabled delay-aware requests, taking account of uncertainty (variability) of processing and transmission delays. The admission of requests in the current time slot can impact the admission of future requests. Understanding how online learning algorithms can adapt to support such admission control remains another potential research topic.

ACKNOWLEDGEMENTS

We would like to thank the three anonymous referees and the associate editor for their expertise comments and constructive suggestions, which have helped us improve the quality and presentation of the paper greatly. The work of Zichuan Xu, Qiufen Xia, and Guowei Wu is partially supported by the National Natural Science Foundation of China (Grant No. 61802048 and 61802047), the fundamental research funds for the central universities in China (Grant No. DUT17RC(3)061, DUT17RC(3)070, DUT19RC(4)035, and DUT19GJ204), and the "Xinghai Scholar Program" in Dalian University of Technology, China. The work by Weifa Liang is supported by the Australian Research Council Discovery Project (Grant No. DP200101985). The work by Pan Zhou is supported by the National Natural Science Foundation of China (Grant No. 61972448).

REFERENCES

[1] O. Alhussein, P. T. Do, J. Li, Q. Ye, W. Shi, W. Zhuang, X. Shen, X. Li, and J. Rao. Joint VNF placement and multicast traffic routing in 5G core networks. *Proc. of ICC*, IEEE, 2018.

[2] 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.

[3] J. Case *et al.* A Simple Network Management Protocol (SNMP). RFC 1098, IETF, *https://tools.ietf.org/html/rfc1157*, 1990.

[4] 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.

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

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

[7] R. Cziva, C. Anagnostopoulos, D. P. Pezaros Dynamic Latency-Optimal vNF Placement at the Network Edge. *Proc. of INFOCOM*, IEEE, 2018.

[8] 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.

[9] GÉANT. `http://www.geant.net`, accessed in Feb. 2020.

[10] E. W. Zegura, K. Calvert and S. Bhattacharjee. How to Model an Internetwork. *Proc. of IEEE INFOCOM*, IEEE, 1996.

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

[12] H3C SDN Switches. http://www.h3c.com/en/Product_Technology/Enterprise_Products/Switches/Campus_Switches/H3C_S5560X-EI/, accessed in Feb. 2020.

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

[14] K. Han, Y. Liu, and J. Luo. Duty-cycle-aware minimum-energy multicasting in wireless sensor networks. *IEEE/ACM Transactions on Networking*, Vol. 21, No. 3, pp. 910 – 923, 2013.

[15] H. Huang, S. Guo, J. Wu, and J. Li. Service chaining for hybrid network function. *IEEE Transactions on Cloud Computing*, Vol. 7, No.4, pp. 1082 –1094, 2019.

[16] 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.

[17] 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.

[18] 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.

[19] N. Kiji, T. Sato, R. Shinkuma, and E. Oki Virtual network function placement and routing model for multicast service chaining based on merging multiple service paths. *Proc. of HPSR*, IEEE, 2019.

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

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

[22] 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.

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

[24] D. Li, M. Xu, Y. Liu, X. Xie, Y. Cui, J. Wang, and G. Chen. Reliable multicast in data center networks. *IEEE Transactions on Computers*, Vol. 63, No. 8, pp. 2011 – 2024, IEEE, 2014.

[25] W. Liang. Approximate minimum-energy multicasting in wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, Vol. 5, No. 4, pp. 377 – 387, 2006.

[26] 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.

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

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

[29] Y. Ma, W. Liang, Z. Xu, and S. Guo. Profit maximization for admitting requests with network function services in distributed clouds. *IEEE Transactions on Parallel and Distributed Systems*, Vol.30, No.5, pp.1143–1157, 2019.

[30] Y. Ma, W. Liang, J. Wu, and Z. Xu. Throughput maximization of NFV-enabled multicasting in mobile edge cloud networks. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 31, No. 2, pp. 393 – 407, 2020.

[31] Y. Ma, W. Liang, and J. Wu. Online NFV-enabled multicasting in mobile edge cloud networks. *Proc. of ICDCS*, IEEE, 2019.

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

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

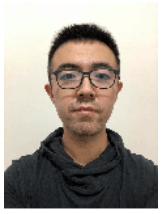
[34] M. Mongioví, A. K. Singh, X. Yan, B. Zong, and K. Psounis. Efficient multicasting for delay tolerant networks using graph indexing. *Proc. of INFOCOM*, IEEE, 2012.

[35] Netconf Working Group. *https://datatracker.ietf.org/wg/netconf/about/*.

[36] Open vSwtich. *https://www.openvswitch.org*

[37] M. Mahalingam et al. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. RFC 7348, IETF, *https://tools.ietf.org/html/rfc7348*.

[38] 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.
[39] 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.
[40] Ryu SDN Controller. *https://osrg.github.io/ryu/*
[41] 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.
[42] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with rocketfuel. *Proc. of SIGCOMM*, ACM, 2002.
[43] J. M. Vella and S. Zammit. A Survey of multicasting over wireless access networks. *IEEE Communications Surveys & Tutorials*, Vol. 15, No. 2, pp. 718 – 753, IEEE, 2013.
[44] K. Xie, X. Zhou, T. Semong, and S. He Multi-source multicast routing with QoS constraints in network function virtualization. *Proc. of ICC*, IEEE, 2019.
[45] 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.18, No.11, pp.2672 – 2685, 2019.
[46] Z. Xu, W. Liang, A. Galis, and Y. Ma. Throughput maximization and resource optimization in NFV-enabled networks. *Proc. of ICC'17*, IEEE, 2017.
[47] 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.
[48] Z. Xu, W. Liang, M. Huang, M. Jia, S. Guo, and A. Galis. Efficient NFV-enabled multicasting in SDNs. *IEEE Transactions on Communications*, Vol.67, No.3, pp. 2052 – 2070, 2019.
[49] B. Yi, X. Wang, M. Huang, and A. Dong. A multi-stage solution for NFV-enabled multicast over the hybrid infrastructure. *IEEE Communication Letters*, vol. 21, no. 9, pp. 2061–2064, 2017.
[50] Y. Zhang *et. al*. StEERING: A software-defined networking for inline service chaining. *Proc. of ICNP*, IEEE, 2013.
[51] S. Q. Zhang, Q. Zhang, H. Bannazadeh, and A. L. Garcia. Network function virtualization enabled multicast routing on SDN. *Proc. of ICC*, IEEE, 2015.



Haozhe Ren received the B.Sc degree from the University of Science and Technology Beijing in China, in 2012, and received the ME degree from the Xinjiang Normal University in China in 2018. He is currently pursuing his PhD degree in the School of Software, Dalian University of Technology. His current research interests include network function virtualization, software-defined networking, algorithmic game theory, and optimization problems.

Zichuan Xu (M'17) received his PhD degree from the Australian National University in 2016, ME degree and BSc degree from Dalian University of Technology in China in 2011 and 2008, all in Computer Science. From 2016 to 2017, he was a Research Associate at Department of Electronic and Electrical Engineering, University College London, UK. He is currently an Associate Professor in School of Software at Dalian University of Technology. He is also a 'Xinghai Scholar' in Dalian University of Technology. His research interests include cloud computing, network function virtualization, software-defined networking, wireless sensor networks, routing protocol design for wireless networks, algorithmic game theory, and optimization problems.

Weifa Liang (M'99–SM'01) received the PhD degree from the Australian National University in 1998, the ME degree from the University of Science and Technology of China in 1989, and the BSc degree from Wuhan University, China in 1984, all in computer science. He is currently a Full Professor in the Research School of Computer Science at the Australian National University. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, mobile edge computing and cloud computing, Network Function Virtualization, Software-Defined Networking, design and analysis of parallel and distributed algorithms, approximation algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE.

Qiufen Xia received her PhD degree from the Australian National University in 2017, the ME degree and BSc degree from Dalian University of Technology in China in 2012 and 2009, all in Computer Science. She is currently a lecturer at the Dalian University of Technology. Her research interests include mobile cloud computing, query evaluation, big data analytics, big data management in distributed clouds, and cloud computing.

Pan Zhou (S07M14) received the B.S. degree in the Advanced Class of Huazhong University of Science and Technology (HUST), Wuhan, China, in 2006, and the Ph.D. degree from the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, in 2011. He is currently an Associate Professor with the School of Electronic Information and Communications, HUST, Wuhan, China. He was a Senior Technical Member with Oracle, Inc., America, from 2011 to 2013, Boston, MA, USA. His current research interests include security and privacy, machine learning and big data analytics, and information networks.

Omer F. Rana received the B.S. degree in information systems engineering from the Imperial College of Science, Technology and Medicine, London, U.K., the M.S. degree in microelectronics systems design from the University of Southampton, Southampton, U.K., and the Ph.D. degree in neural computing and parallel architectures from the Imperial College of Science, Technology and Medicine. He is a Professor of performance engineering with Cardiff University, Cardiff, U.K. His current research interests include problem solving environments for computational science and commercial computing, data analysis and management for large-scale computing, and scalability in high performance agent systems.

Alex Galis is a Professor in Networked and Service Systems at University College London. He has co-authored 10 research books and more that 250 publications in the Future Internet areas: system management, networks and services, networking clouds, 5G virtualisation and programmability. He was a member of the Steering Group of the Future Internet Assembly (FIA) and he led the Management and Service-aware Networking Architecture (MANA) working group. He acted as TPC chair of 14 IEEE conferences. He is also a co-editor of the IEEE Communications Magazine feature topic on Advances In Networking Software. He acted as a Vice Chair of the ITU-T SG13 Group on Future Networking. He is involved in IETF and ITU-T SG13 network slicing activities and he is also involved in IEEE SDN initiative.

Guowei Wu received his Ph.D degree from Harbin Engineering University in 2003,PR China.He is now a professor at the School of Software,Dalian University of Technology(DUT) in China. His research interests include embedded real-time system,cyber-physical systems(CPS), and smart edge computing. He has published over 100 papers in Journal and Conference.