Online NFV-Enabled Multicasting in Mobile Edge Cloud Networks

Yu Ma†, Weifa Liang†, and Jie Wu‡
† The Australian National University, Canberra, ACT 2601, Australia
‡ Temple University, Philadelphia, PA, USA

***Abstract*—Mobile Edge Computing (MEC) reforms the cloud paradigm by bringing unprecedented computing capacity to the vicinity of mobile users at the mobile network edge. This provides end-users with swift and powerful computing, energy efficiency, storage capacity, mobility- and context-awareness support. Furthermore, provisioning virtualized network services in MEC can improve user service experience, simplify network service deployments, and ease network resource management. However, user requests usually arrive into the system dynamically and different user requests may have different resource demands. How to optimize and guarantee the performance of MEC is of significant importance and challenging. In this paper, we study the problem of online NFV-enabled multicasting in an MEC network with resource capacity constraints on both cloudlets and links. We first devise an approximation algorithm for the cost minimization problem for a single NFV-enabled multicast request admission. We then propose an online algorithm with a provable competitive ratio for the online throughput maximization problem where NFV-enabled multicast requests arrive one by one without the knowledge of future request arrivals. We admit the requests through placing or sharing VNF instances of network functions in their service chains to meet their computing and bandwidth resource demands, and we introduce a novel cost model to capture the dynamic usages of different resources and perform network resource allocations based on the proposed cost model. We finally evaluate the performance of the proposed algorithms through experimental simulations. Simulation results demonstrate that the proposed algorithms are promising.**

I. Introduction

Mobile devices, including smart phones and tablets, gain increasing popularity as communication tools of users for their business, social networking, and personal entertainment. However, their computing, storage and battery capacity is very limited, due to their portal size. Leveraging by rich computing and storage resources in public and private clouds, mobile devices can offload their tasks to remote clouds for processing and storage. However, such rich-resource clouds are usually far away from users. The response delay to user requests may not be tolerable for some real-time applications, and their availability and security are also concerned [12]. Instead, a new network paradigm, Mobile Edge Computing (MEC) is emerged, which can provide cloud-computing capabilities at the edge of pervasive Radio Access Network (RAN) in close proximity to mobile users [1]. It can significantly shorten the response delay, ensure highly efficient network operation and service delivery, and offer an improved user experience. MEC thus is an ideal platform to meet ever-growing resource demands of mobile users for their applications, by enhancing mobile device capabilities with a realtime manner for autonomous vehicles, e-Health, Internet of Things (IoT), virtualized/augmented reality, etc. In addition to MEC, Network Function Virtualization (NFV) [15] has been envisaged as another key technology to the next-generation networking paradigm that enables fast service deployments, and inexpensive and error-free service provisioning in future communication networks [5]. It replaces resource demanding service applications, such as object recognition, voice control, or virtual reality, with software components in servers or cloudlets that provide the same capability. Each network function runs in a virtual machine, referred to as a virtualized network function instance, hosted in a cloudlet.

Although implementing network functions as VNF instances in cloudlets is a promising technology for simplifying network service deployments, easing network resource management, and improving user service experience, it poses several fundamental challenges. One major challenge is the limited resources on both cloudlets and links of mobile edge cloud networks compared to powerful centralized data center networks. It is of paramount importance to optimize the performance of an MEC network through judicious allocation of its limited resources. In addition, each NFV-enabled multicast request has a requirement of a service function chain. How to steer the data traffic of the request to go through each network function in its service function chain correctly? Furthermore, the implementation of a request may share an existing network function instance with the other request implementations or create a new VNF instance. How to make a decision to create a new VNF instance or make use of an existing VNF instance to minimize the operational cost of service providers? Finally, how to deal with request admissions providing that multicast requests arrive into the system dynamically without the knowledge of future request arrivals. In this paper, we will address the aforementioned challenges.

The novelties of this work lie in proposing an approximation algorithm for a single NFV-enabled multicast request admission, through constructing an auxiliary graph and reducing the problem to a minimum cost steiner tree in the auxiliary graph. Furthermore, an online algorithm with a provable competitive ratio for a sequence of NFV-enabled multicast request admissions is also proposed, through the development of a novel cost model to accurately capture the usage costs of different resources in cloudlets and links.

The main contributions of this paper are summarized as fol-

lows. We study the NFV-enabled multicast request admissions in mobile edge cloud networks with the aim to either minimize the admission cost of a single NFV-enabled multicast request admission, or maximize the network throughput through admitting a sequence of NFV-enabled multicast requests dynamically, by taking both resource capacity constraints on cloudlets and links, and the service chain of each request into consideration. We first propose an approximation algorithm for the cost minimization of a single multicast request admission. We then devise an online algorithm with a provable competitive ratio for dynamic multicast request admissions. We finally evaluate the performance of the proposed algorithms through experimental studies. The simulation results reveal that the proposed algorithms are very promising.

The rest of the paper is organized as follows. Section II reviews related work. Section III introduces notions, notations, and problem definitions. Section IV devises an approximation algorithm for the NFV-enabled multicasting cost minimization problem. Section V develops an efficient online algorithm with a provable competitive ratio for dynamic NFV-enabled multicast request admissions. Section VI evaluates the proposed algorithms empirically, and Section VII concludes the paper.

II. Related work

As a key-enabling technology of 5G, MEC networks have gained tremendous attention from the research community recently [13]. Also, with the emergence of complicated and resource-hungry mobile applications, implementing user tasks in cloudlets of a nearby mobile edge-cloud network is becoming an important approach to reduce mobile device energy consumption and improve user experience. There are extensive studies on resource allocations in MEC networks [2], [3], [6], [7], [9], [10], [13]. For example, Jia *et al.* [8] considered the assignment of user requests to different cloudlets in a Wireless Metropolitan Area Network with the aim to minimize the maximum delay among offloaded tasks, by developing heuristics for the problem. Feng *et al.* [6] proposed an algorithm with performance guarantee for placing VNFs in distributed cloud networks and routing service flows among the placed VNFs under the constraints of service function chains of the requests.

All the aforementioned studies assumed that each task will be allocated with dedicated computing resources. There is no consideration for whether there are existing VNF instances in cloudlets to serve them. However, many tasks usually request for the same type of network services. If the VNF instance for a specified service has already been instantiated and its workload has not reached its capacity, the other tasks that requested for the service can make use of the VNF instance. Several recent works started exploring VNF instance sharing [7], [10]. For example, He *et al.* [7] recently studied the joint service placement and request scheduling in order to optimally provision edge services while taking into account the demands of both sharable and non-sharable resources. They aim to maximize the network throughput, by developing heuristic algorithms. There are several studies of NFV-enabled multicasting in MEC environments [17], [18], [19]. For example, Zhang *et al.* [19] investigated the NFV-enabled multicasting problem in SDNs. They assumed that there are sufficient computing and bandwidth resources to accommodate all multicast requests, for which they provided a 2-approximation algorithm if only one server is deployed for implementing the service chain of each multicast request. Xu *et al.* [18] considered the cost minimization of admitting a single NFV-enabled multicast request with the QoS requirement in MEC, where the implementation of the service chain of each request will be consolidated into a single cloudlet. They developed both approximation and heuristic algorithms for the problem, by placing no more than constant numbers of VNF instances of the service chain of the request in different branches of the found pseudo-multicast tree for the request.

III. Preliminaries

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

A. System model

We consider a mobile edge cloud (computing) network (MEC) in a metropolitan region, which is modelled by an undirected graph $G = (V, E)$, where V is a set of *access points* (APs) located at different locations in the metropolitan region, e.g., shopping centers, airports, restaurants, bus stations, and hospitals. A cloudlet is co-located with each AP node $v \in V$ via a high-speed optical cable, which implies that the communication delay between them is negligible due to plenty of bandwidth on the cable. For simplicity, an AP node and the co-located cloudlet will be used interchangeably if no confusion arises. Each cloudlet has computing capacity C_v for implementing various virtualized network functions (VNFs) requested by mobile users. E is the set of links between APs. Each link $e \in E$ has a bandwidth capacity B_e. We assume that each AP node covers a certain area, in which mobile users can access the MEC wirelessly through it. In case a mobile user located at an overlapping coverage region by multiple APs, the mobile user can choose connecting to its nearest AP (or an AP with the strongest signal strength). Fig. 1 is an example of an MEC network.

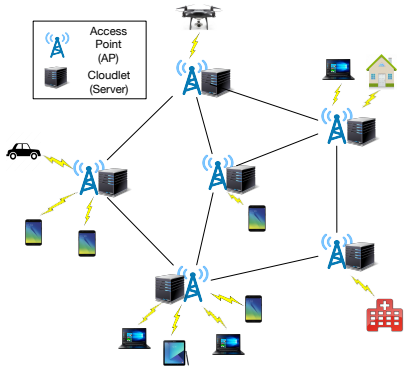


Fig. 1. An example of an MEC network with 6 APs and each is attached with a cloudlet.

B. NFV-enabled multicast requests with service function chain requirements

Consider *an NFV-enabled multicast request* $r_j = (s_j, D_j, SFC_j, \rho_j)$ that transfers data traffic from a source node $s_j \in V$ to a given set of destination nodes $D_j \subseteq V$ with a specified packet rate ρ_j. Each packet in the data traffic must pass through a sequence of network functions of its specified type of *service function chain* $SFC_j = \langle f_{j,1}, \ldots, f_{j,l}, \ldots, f_{j,L_j} \rangle$ before reaching its destinations, where L_j is the length of the service function chain.

We assume that resources in cloudlets are virtualized, using container-based lightweight virtualization technologies, and thus can be allocated and shared flexibly. Each instance of a *virtualized network function* (VNF) is a lightweight virtual machine in a cloudlet. Implementing VNF instances in cloudlets consumes computing resource of cloudlets. Without loss of generality, we assume that different types of VNFs in service function chains of all requests can be classified into K different types. Denote by $f^{(k)}$ and $C(f^{(k)})$ the virtualized network function of type k and the amount of computing resource consumed for its implementation in a cloudlet respectively, $1 \leq k \leq K$. Suppose each VNF instance of $f^{(k)}$ has a maximum processing capacity $\mu^{(k)}$. Furthermore, if the residual processing capacity of a VNF instance is sufficient to process the data traffic of a request, this VNF instance can be shared by the request. Otherwise, a new VNF instance needs to be instantiated in a cloudlet if it has sufficient residual computing resource in the admission of the request.

To implement an NFV-enabled multicast request r_j, each packet of its data traffic is enforced to go through an instance of each network function in its service function chain SFC_j prior to reaching its destinations in D_j. Denote by $T(j)$ the multicast tree that transfers the data traffic of NFV-enabled multicast request r_j from the source s_j to destinations in D_j. Fig. 2 is an illustrative example of a multicast request implementation. To this end, an existing VNF instance (with sufficient residual processing capacity) must be selected or a new VNF instance must be instantiated in a cloudlet, for each network function $f_{j,l}$ in its service function chain SFC_j. Without loss of generality, we assume that the VNF instances of service function chain SFC_j can be placed at different cloudlets, i.e., the VNF instances of a service function chain are not necessarily consolidated into a single cloudlet only.

C. Admission cost of an NFV-enabled multicast request

The operational cost of an MEC network for NFV-enabled multicast requests mainly consists of three components, the VNF instance processing cost for processing request data packets, the VNF instance instantiation cost for instantiating VNF instances in cloudlets, and the bandwidth usage cost for routing data traffic of the request along links. Instantiating VNF instances at cloudlets consumes their computing and storage resources, thus incurs the VNF instantiation cost. Denote by $c_{ins}(f^{(k)}, v)$ the instantiation cost a VNF instance of network function $f^{(k)}$ in a cloudlet v, and $\rho_j \cdot c_{proc}(f^{(k)}, v)$ the processing cost of data traffic of a request r_j at a VNF instance of $f^{(k)}$ at cloudlet v, where $c_{proc}(f^{(k)}, v)$ is the cost of processing a packet by a VNF instance $f^{(k)}$ at cloudlet v and ρ_j is the demanded packet rate of r_j. Notice that the processing cost of a packet $c_{proc}(f^{(k)}, v)$ of different VNF instances at different cloudlets may be significantly different, since different VNF instances consume different amounts of computing resources, and servers in different cloudlets have different amounts of energy consumptions.

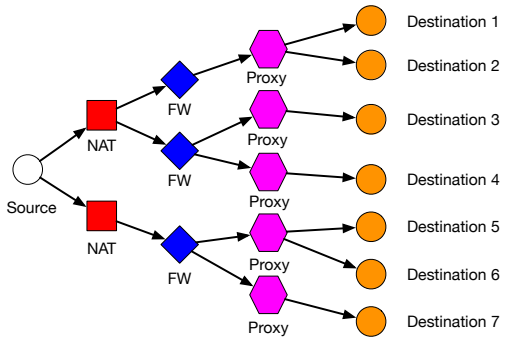


Fig. 2. An example of an NFV-enabled multicast request with a service function chain consists of three network functions, Network Address Translation (NAT), Firewall (FW), and Proxy. Data traffic of the multicast request is transferred from the source node $Source$ to a set of 7 destination nodes. Each packet of the request must pass through an instance of network function in its service function chain.

In addition to the processing cost of its data traffic at VNF instances, the data traffic of request r_j is routed along a pseudo-multicast tree $T(j)$ (it may not be a multicast tree) in the network from the source node s_j to the destination nodes in D_j, which incurs the communication cost. The routing cost of data packets of r_j along multicast tree $T(j)$ thus is $\rho_j \cdot c_{bw}(T_j) = \rho_j \cdot \sum_{e \in T(j)} c_e$, where c_e is the unit transmission cost on link $e \in E$, and $c_{bw}(T_j)$ is the cost of transferring a packet along the multicast tree $T(j)$.

D. Problem definitions

In this paper, we consider two NFV-enabled multicast request admission problems. We first consider the cost minimization problem for a single NFV-enabled multicast request admission; we then consider the online multicast request admissions in which requests arrive one by one without the knowledge of future request arrivals, we aim to maximize the network throughput. The precise definitions of the two problems are given below.

Definition 1: Given an MEC network $G = (V, E)$ with a set V of cloudlets, each $v \in V$ having computing capacity C_v, let B_e be the bandwidth capacity of each link $e \in E$, an NFV-enabled multicast request $r_j = (s_j, D_j, SFC_j, \rho_j)$, *the NFV-enabled multicasting problem* is to find a pseudo-multicast tree for r_j to route its data traffic from the source node s_j to each destination node in D_j while each packet of its data traffic must pass through each VNF instance in its service function chain SFC_j, such that its implementation cost is minimized, subject to the computing and bandwidth capacities on both cloudlets and links of the network.

Definition 2: Given an MEC network $G = (V, E)$ with a set V of cloudlets, each $v \in V$ has computing capacity

C_v, and each link $e \in E$ has bandwidth capacity B_e. Let $r_1, r_2, \ldots, r_j$ be a sequence of NFV-enabled multicast requests that arrive into the system one by one without the knowledge of future request arrivals, *the online multicasting throughput maximization problem* in G is to maximize the number of requests admitted, subject to resource capacities on both cloudlets and links of the network.

The defined two problems are NP-hard, and their NP-hard proofs are omitted due to space limitation.

IV. An approximation algorithm for the NFV-enabled multicasting problem

In this section, we deal with the NFV-enabled multicasting problem. We first devise an approximation algorithm for the problem, and then analyze the performance of the proposed algorithm.

A. Algorithm overview

Given an MEC $G = (V, E)$ and a multicast request r_j, we aim to minimize the operational cost of the service provider by steering the data traffic of request r_j from a source s_j to a set of destinations in D_j while each packet must pass through a sequence of network functions in the service function chain SFC_j demanded by the request. There are two important challenges to tackle the problem. One is the resource availability. Given a multicast request r_j, whether it should be admitted or rejected is determined by the availability of its demanded resources in the network G; the other is which cloudlets should be selected to implement which network functions in its service function chain, and whether new VNF instances to be instantiated or existing VNF instances can be shared. Addressing these two challenges is essential for delivering a feasible and cost-efficient solution to the problem.

The basic idea of the proposed approximation algorithm for the problem is to reduce the cost minimization problem in an auxiliary acyclic graph for the multicast request r_j. Then, if a multicast tree exists in the auxiliary graph, there will be sufficient resources in G to meet the resource demands of the request, and a pseudo-multicast tree $T(j)$ in G for r_j can be derived from the found multicast tree in the auxiliary graph.

B. Approximation algorithm

For a given multicast request r_j, we can either make use of existing network function instances as long as their residual processing capacities are sufficient to admit request r_j. Also, if there is sufficient available computing resource in a cloudlet, a new instance for that type of network function can be instantiated. Thus, there can be multiple candidate instances of the lth network function $f_{j,l}$ in its service function chain SFC_j in G with $1 \leq l \leq L_j$.

Define a function $\lambda(j, l) = k$, with $1 \leq k \leq K$, to represent type k of the lth network function $f_{j,l}$ in SFC_j of request r_j. Denote by $F_v^{(k)}$ the set of VNF instances of type k instantiated in cloudlet v. Let μ_i^{re} be the residual processing capacity of VNF instance $i \in F_v^{(k)}$. And let C_v^{re} be the residual computing capacity of cloudlet $v \in V$. Denote by $N_{l,v}$ the set of VNF instances that can be employed as the lth network function $f_{j,l}$ in SFC_j in cloudlet v, including both existing network function instances with sufficient residual processing capacities, i.e., $\mu_i^{re} \geq \rho_j$ with $i \in F_v^{(\lambda(j,l))}$, as well as a new VNF instance i' to be created providing sufficient computing resource in cloudlet v, i.e, $C_v^{re} \geq C(f^{(\lambda(j,l))})$. Then N_l is the set of VNF instances that can be employed as the lth network function $f_{j,l}$ in SFC_j among all cloudlets in V, i.e., $N_l = \cup_{v \in V} N_{l,v}$. We assume that the number of VNF instances of the same type in each cloudlet is a small constant.

To this end, we construct an auxiliary directed graph $G'_j = (V'_j, E'_j)$ from G for request r_j as follows. We first remove all links from G if their residual bandwidth is less than ρ_j. The node set V'_j of G'_j is the union on sets N_l of VNF instances for $1 \leq l \leq L_j$, with the source node s_j, the destination node set D_j of multicast request r_j, i.e., $V'_j = \cup_{l=1}^{L_j} N_l \cup \{s_j\} \cup D_j$. In order to make sure that network functions of $SFC_j = \langle f_{j,1}, \ldots, f_{j,l}, \ldots, f_{j,L_j} \rangle$ are traversed in this specified order, we add a directed edge from a node $x \in N_{l-1}$ to each node $y \in N_l$ for $2 \leq l \leq L_j$ if a path in G between x and y exists, and a weight $w(x, y)$ is assigned to this edge, which is the communication cost along the shortest path between the cloudlets that implementing VNF instances x and y, and the processing and VNF instance instantiation cost of network function y. Notice that if the VNF instance is an existing instance, the VNF instance instantiation cost is 0; and if two network functions x, y resides in the same cloudlet, the communication cost between them is 0. We then add a directed edge from s_j to each node $y \in N_1$ if such a path in G exists. A weight assigned to it is the cost of communication cost along the path and the processing and VNF instance instantiation cost of the network function y. Notice that the type of network function y is $\lambda(j, 1)$. Also, we add a directed edge from a node $x \in N_L$ to a node $y \in D_j$, and set the communication cost along the shortest path from a cloudlet that implement network function x to AP node y as its weight if such a path exists in G. Thus, $E'_j = \cup_{l=2}^{L_j} \{\langle x, y \rangle \mid x \in N_{l-1}, y \in N_l\} \cup \{\langle s_j, y \rangle \mid y \in N_1\} \cup \{\langle x, y \rangle \mid x \in N_{L_j}, y \in D_j\}$. In order to make sure a multicast request can be admitted without violating computing capacity of any cloudlet, we adopt a conservative strategy such that only if the residual computing capacity of a cloudlet is sufficient to create all necessary VNF instances (VNF instances that do not have enough residual processing capacity or do not exist in this cloudlet), this cloudlet will be selected to create new VNF instances. Fig. 3 shows the construction of G'_j.

Having the constructed auxiliary graph G'_j, the problem is reduced to find a directed multicast tree $T'(j)$ in G'_j rooted at s_j and spanning all nodes in D_j, such that the weighted sum of edges in $T'(j)$ is minimized. This is the classic Directed Steiner Tree problem, which is NP-hard. There is an approximate solution which is $|D_j|^\epsilon$ times of the optimal [4], where ϵ is a constant with $0 < \epsilon \leq 1$.

If a multicast tree $T'(j)$ in G'_j exists, a pseudo-multicast tree $T(j)$ in G rooted at s_j and spanning all nodes in D_j

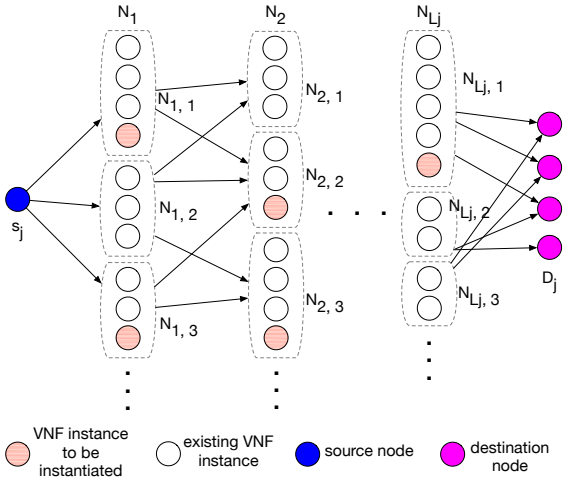


Fig. 3. A constructed auxiliary graph G'_j with L_j+2 layers for NFV-enabled multicast request r_j. Layer 0 is the source node s_j. Layer L_j+1 consists of destination nodes in D_j. Each layer l, with $1 \leq l \leq L_j$, consists of VNF instances of type $\lambda(j,l)$ that can be employed to process data traffic of request r_j in each cloudlet $v \in V$. If there is sufficient residual computing resource in a cloudlet, a new VNF instance of that type can be instantiated. Notice that, if there is a path between a VNF instance implemented in cloudlet $u \in V$ and a VNF instance implemented in cloudlet $v \in V$, then there will be a path between any pair of VNF instances implemented in the two cloudlets respectively. For simplicity, we use an edge in the graph to represent a set of edges between each pair of VNF instances resides in the two cloudlets respectively.

can then be derived, where *a pseudo-multicast tree* in fact is a graph in which nodes and links can appear multiple times. Specifically, we expand each directed edge in the multicast tree $T'(j)$ to a set of edges in the corresponding shortest path of G. The detailed description of the algorithm for the NFV-enabled multicasting problem is given in `Algorithm` 1.

Algorithm 1 Finding a minimum-cost multicast tree in G for request r_j

Input: An MEC network $G=(V,E)$ with a set V of cloudlets, a multicast request $r_j=(s_j, D_j, SFC_j, \rho_j)$.
Output: Admit or reject request r_j. If r_j is admitted, a pseudo-multicast tree $T(j)$ in G is delivered.

1: Remove all edges in G with residual bandwidth less than ρ_j;
2: Compute all pairs shortest paths between each pair of AP nodes in G;
3: Construct the directed auxiliary graph G'_j from G, and assign a cost weight on each of its edges;
4: Find an approximate multicast tree $T'(j)$ in G'_j rooted at s_j, and spanning all nodes in D_j, by applying the algorithm due to Charikar *et al.* [4];
5: **if** $T'(j)$ in G'_j exists **then**
6: A pseudo-multicast tree $T(j)$ in G is derived, by replacing each edge in $T'(j)$ with its corresponding set of edges of a shortest path in G;
7: If a selected VNF instance is to be instantiated, create a new VNF instance in its cloudlet;
8: Update residual resource capacities of links, cloudlets, and VNF instances in G;
9: **else**
10: Reject multicast request r_j.
11: **end if**

C. Algorithm analysis

In the following, we show the correctness of the proposed algorithm, `Algorithm` 1, and analyze its approximation ratio and time complexity.

Theorem 1: Given an MEC network $G=(V,E)$ with a set V of APs each attached with a cloudlet, and a multicast request $r_j=(s_j, D_j, SFC_j, \rho_j)$, there is an approximation algorithm, `Algorithm` 1, for the NFV-enabled multicasting problem with an approximation ratio of $|D_j|^{\epsilon}$, which takes $O((L_j \cdot |V|)^{\frac{1}{\epsilon}}|D_j|^{\frac{2}{\epsilon}} + |V|^3)$ time, where L_j is the length of service function chain SFC_j of request r_j, and ϵ is a constant with $0 < \epsilon \leq 1$.

Proof We first show the solution obtained by the proposed algorithm is feasible. Following the construction of G'_j, G'_j is a layered directed acyclic graph, node s_j is at layer 0, each node $x \in N_l$ is at layer l, with $1 \leq l \leq L_j$, and each node $x \in D_j$ is at layer L_j+1, assuming that $|SFC_j| = L_j$. If $T'(j)$ is the found multicast tree in G'_j, it can be seen that there is a corresponding pseudo-multicast tree in G rooted at s_j and spanning all nodes in D_j. Since G'_j is a layered directed acyclic graph, the multicast tree $T'(j)$ passes through a cloudlet in layer l for implementing network function $f_{j,l}$ in its service function chain SFC_j, with $1 \leq l \leq L_j$. The VNF instance for $f_{j,l}$ can be an existed VNF instance with sufficient residual processing capacity or a new VNF instance to be instantiated.

The admission cost of multicast request r_j consists of three components, the VNF instance processing cost, the VNF instance instantiation cost, and the bandwidth usage cost. Each packet of request r_j is transferred from the source node s_j to each destination node in D_j while passing through each VNF instance in its service function chain SFC_j. The use of each VNF instance $f_{j,l}$ in layer N_l for processing data packets incurs the VNF instance processing cost, and if the VNF instance is newly instantiated, there is VNF instance instantiation cost. When data packets are transferred from a cloudlet (VNF instance) to the next cloudlet (VNF instance), there is communication cost in links. The summation of all these costs are assigned to each directed edge in E'_j. Thus, the cost of the minimum steiner tree $T'(j)$ found in G'_j from s_j while spanning all nodes in D_j, is the minimum admission cost of r_j in G. Following [4], the approximation ratio of the proposed algorithm for NFV-enabled multicast problem for a multicast request r_j is $|D_j|^{\epsilon}$, where ϵ is a constant with $0 < \epsilon \leq 1$.

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

V. An Online Algorithm for the Online Multicasting Throughput Maximization Problem

In this section, we study the online multicasting throughput maximization problem, by assuming that requests arrive one by one without the knowledge of future request arrivals. We first propose an efficient online algorithm for the problem, by building a novel cost model to capture the dynamic resource

consumptions in G and performing resource allocation to admit requests based on the cost metric. We then analyze the competitive ratio and time complexity of the proposed online algorithm.

A. Resource usage cost model

The basic idea of the proposed online algorithm is to regulate an online admission control policy to respond to each incoming NFV-enabled multicast request by either admitting or rejecting it, depending on the availability of its demanded resources. We still make use of the constructed auxiliary graph G'_j for the NFV-enabled multicasting problem in the previous section. However, the weight assigned to each node and each edge in G'_j will be dynamically determined.

As a VNF instance can be shared by multiple requests provided that the sum of packet rates going through the instance is no greater than its processing capacity, we treat the VNF instance processing capacity of each VNF instance as a type of resource, as well as computing resource in cloudlets, and bandwidth along links. We start by introducing a resource usage cost model to measure the resource consumption by each VNF instance (processing capacity), each cloudlet (computing resource) and each link (bandwidth) when admitting multicast requests, in which if a specific type of resource has been highly utilized, it should not be encouraged to be used in the near future, and the use of it will result in a higher cost. Otherwise, if a resource that has rarely been used should be encouraged to use by assigning it a lower cost. Thus, the resources in the network can be optimally utilized among user requests to maximize the network throughput. We here treat processing capacity of VNF instances and computing resource in cloudlets for creating VNF instances as different type of resources.

The proposed online algorithm examines each arrived multicast request one by one. When a request r_j arrives, the resource availability of VNF instances, cloudlets and links will determine whether r_j should be admitted. Recall that $F_v^{(k)}$ the set of VNF instances of type k in cloudlet v. If there is sufficient computing resource in cloudlet v, a new VNF instance of type k can be instantiated. Here, let $F_v^{(k)}$ include the new VNF instance of type k to be created. Denote by $\mu_{v,i}^{(k)}(j)$ the residual processing capacity of VNF instance i of type k in cloudlet v when request r_j arrives, with $\mu_{v,i}^{(k)}(0) = \mu^{(k)}$, $i \in F_v^{(k)}$. If request r_j is admitted and its packets is processed by the VNF instance, then $\mu_{v,i}^{(k)}(j+1) = \mu_{v,i}^{(k)}(j) - \rho_j$; otherwise, the residual computing capacity is unchanged, i.e., $\mu_{v,i}^{(k)}(j+1) = \mu_{v,i}^{(k)}(j)$. Similarly, denote by $C_v(j)$ and $B_e(j)$ the residual computing capacity at cloudlet v and residual bandwidth in link e, when request r_j arrives.

To capture the resource usage of r_j, we use an exponential function to model the cost $W_{v,i}^{(k)}(j)$ of processing packets of r_j by VNF instance $i \in F_v^{(k)}$ as follows,

$$W_{v,i}^{(k)}(j) = \mu^{(k)}(\alpha^{1-\frac{\mu_{v,i}^{(k)}(j)}{\mu^{(k)}}} - 1), \tag{1}$$

where α (> 1) is a tuning parameter to be decided later, and $1 - \frac{\mu_{v,i}^{(k)}(j)}{\mu^{(k)}}$ is the processing capacity utilization ratio in VNF instance i when request r_j is considered.

Similarly, the cost $W_v(j)$ of instantiating a new VNF instance at cloudlet $v \in V$ and the cost $W_e(j)$ of using bandwidth resource at link $e \in B$ are defined, respectively,

$$W_v(j) = C_v(\beta^{1-\frac{C_v(j)}{C_v}} - 1), \tag{2}$$

$$W_e(j) = B_e(\gamma^{1-\frac{B_e(j)}{B_e}} - 1), \tag{3}$$

where β (> 1) and γ (> 1) are tuning parameters to be decided later, and $1 - \frac{C_v(j)}{C_v}$ and $1 - \frac{B_e(j)}{B_e}$ are the resource utilization ratios in cloudlet v and in link e, respectively, when request r_j is considered. In order to encourage the sharing of VNF instances among multicast requests, we assume that the cost of creating a new VNF instance much higher than the cost of processing capacity usage, i.e., $\beta \gg \alpha$.

We then define *the normalized usage cost* of each VNF instance $i \in F_v^{(k)}$ in cloudlet v for request r_j as,

$$\omega_{v,i}^{(k)}(j) = W_{v,i}^{(j)}(j)/\mu^{(k)} = \alpha^{1-\frac{\mu_{v,i}^{(k)}(j)}{\mu^{(k)}}} - 1. \tag{4}$$

Similarly, the normalized usage costs $\omega_v(j)$ at each cloudlet $v \in V$ and $\omega_e(j)$ at each link $e \in E$ for request r_j are defined as follows,

$$\omega_v(j) = W_v(j)/C_v = \beta^{1-\frac{C_v(j)}{C_v}} - 1, \tag{5}$$

$$\omega_e(j) = W_e(j)/B_e = \gamma^{1-\frac{B_e(j)}{B_e}} - 1. \tag{6}$$

For each request r_j, we construct an auxiliary graph $G'_j = (V'_j, E'_j)$ similar as the one for the NFV-enabled multicasting problem. The difference is the weight assigned to each directed edge in E'_j is the sum of three normalized usage costs defined in (4), (5), (6), respectively. Similar as before, if the VNF instance is an existing instance, the VNF instance instantiation cost is 0; and if two network functions resides in the same cloudlet, the communication cost between them is 0.

To avoid admitting requests that consume too much resources, thereby undermining the performance of the MEC, we adopt the following admission control policy: If (i) the sum of normalized usage costs of the VNF instances in its service function chain of request r_j is greater than σ_1, i.e., $\sum_{v \in V} \sum_{l=1}^{L_j} \sum_{i \in F_v^{(\lambda(j,l))}} \omega_{v,i}^{(\lambda(j,l))}(j) > \sigma_1$, where $L_j = |SFC_j|$; or (ii) the sum of normalized usage costs of VNF instantiation for request r_j is greater than σ_2, $\sum_{v \in V} \omega_v(j) > \sigma_2$; or (iii) the sum of normalized usage costs of links for request r_j is greater than σ_3, $\sum_{e \in E} \omega_e(j) > \sigma_3$, r_j will be rejected, where σ_1, σ_2, σ_3 are the admission control thresholds of resource usages in VNF instances, cloudlets, and links, respectively, where $\sigma_1 = \sigma_2 = \sigma_3 = n$, and $n = |V|$. The detailed algorithm for the online multicasting throughput maximization problem is detailed in `Algorithm` 2.

B. Algorithm analysis

We now analyze the competitive ratio and time complexity of the proposed online algorithm, `Algorithm` 2. We first

Algorithm 2 Online algorithm for the online multicasting throughput maximization problem

Input: An MEC network $G = (V, E)$ with a set V of APs each $v \in V$ attached with a cloudlet with computing capacity C_v, a sequence of multicast requests $r_j = (s_j, D_j, SFC_j, \rho_j)$.

Output: A solution to maximize the network throughput, by admitting or rejecting each arriving multicast request r_j. If r_j admitted, a routing multicast tree for r_j from source node s_j to a set of destination nodes in D_j will be delivered.

while request r_j arrives **do**
  Remove all edges with residual bandwidth less than ρ_j;
  Construct the auxiliary graph $G'_j = (V'_j, E'_j)$ for request r_j, assign weight to each edge in E'_j as stated;
  Find an approximate multicast tree $T'(j)$ in G'_j rooted at s_j and spanning all nodes in D_j, by applying the algorithm due to Charikar *et al.* [4];
  if $T'(j)$ does not exist **then**
    Reject multicast request r_j;
  else
    Determine whether r_j should be accepted or not by the admission control policy;
    if r_j is admitted **then**
      A pseudo-multicast tree $T(j)$ in G is derived, by replacing each edge in $T'(j)$ with its corresponding set of edges in G;
      If a selected VNF instance is to be instantiated, create a new VNF instance;
      Update residual resource capacities of VNF instances, links and cloudlets in G;
    end if
  end if
end while

show the upper bound on the total cost of admitted requests. We then provide a lower bound on the cost of a rejected request by `Algorithm 2` but admitted by an optimal offline algorithm. We finally derive the competitive ratio of `Algorithm 2`.

As for each network function in service function chain SFC_j of r_j, a new VNF instance can be instantiated or an existing VNF instance can be shared, we introduce a binary variable $x_v^{(\lambda(j,l))}$ with $x_v^{(\lambda(j,l))} = 1$ if the lth VNF instance is newly instantiated in cloudlet v; otherwise, it is 0.

Lemma 1: Given an MEC network $G = (V, E)$ with a set V of APs that each $v \in V$ is attached with a cloudlet with computing capacity C_v and link bandwidth capacity B_e for each link $e \in E$, denote by $\mathcal{A}(j)$ the set of NFV-enabled multicast requests admitted by the algorithm, `Algorithm 2`, until the arrival of request r_j. Then, the cost sums of VNF instances, cloudlets, and links when multicast request r_j arrives are

$$\sum_{v\in V}\sum_{l=1}^{L_j}\sum_{i\in F_v^{(\lambda(j,l))}} W_{v,i}^{(\lambda(j,l))}(j) \le 2n\log\alpha\cdot\mathbb{B}(j), \quad (7)$$

$$\sum_{v\in V} W_v(j) \le 2nL_{max}\log\beta\cdot|\mathcal{A}(j)|\cdot C(f_{max}), \quad (8)$$

$$\sum_{e\in E} W_e(j) \le 2n\log\gamma\cdot\mathbb{B}(j), \quad (9)$$

respectively, provided that the maximum length of any service function chain is no greater than n, i.e., $L_{max} = \max_{1\le j'\le j}|SFC_{j'}| \le n$, and $\rho_{j'} \le \frac{\min_{1\le l\le L_{j'}}\mu^{(\lambda(j',l))}}{\log\alpha}$, $\sum_{l=1}^{L_{j'}} C(f^{(\lambda(j',l))})\cdot x_v^{(\lambda(j',l))} \le \frac{\min_{v\in V}C_v}{\log\beta}$, $\rho_{j'} \le \frac{\min_{e\in E}B_e}{\log\gamma}$ with $1 \le j' \le j$, where $\sum_{l=1}^{L_{j'}} C(f^{(\lambda(j',l))})\cdot x_v^{(\lambda(j',l))}$ is the computing resource being occupied by newly instantiated VNF instances in cloudlet v for request $r_{j'}$, $\mathbb{B}(j)$ is the accumulative bandwidth resource being occupied by the admitted requests, i.e., $\mathbb{B}(j) = \sum_{r_{j'}\in\mathcal{A}(j)}\rho_{j'}$, and $C(f_{max})$ is the maximum computing resource required among all VNF instance types, i.e., $C(f_{max}) = \max_{1\le k\le K}\{C(f^{(k)})\}$.

Proof Consider a request $r_{j'} \in \mathcal{A}(j)$ admitted by `Algorithm 2`. For any VNF instance $i \in F_v^{(k)}$, we have

$$\begin{aligned}
&W_{v,i}^{(k)}(j'+1) - W_{v,i}^{(k)}(j') \\
&= \mu^{(k)}(\alpha^{1-\frac{\mu_{v,i}^{(k)}(j'+1)}{\mu^{(k)}}} - 1) - \mu^{(k)}(\alpha^{1-\frac{\mu_{v,i}^{(k)}(j')}{\mu^{(k)}}} - 1) \\
&= \mu^{(k)}\alpha^{1-\frac{\mu_{v,i}^{(k)}(j')}{\mu^{(k)}}}(\alpha^{\frac{\mu_{v,i}^{(k)}(j')-\mu_{v,i}^{(k)}(j'+1)}{\mu^{(k)}}} - 1) \\
&= \mu^{(k)}\alpha^{1-\frac{\mu_{v,i}^{(k)}(j')}{\mu^{(k)}}}(\alpha^{\frac{\rho_{j'}}{\mu^{(k)}}} - 1) \\
&= \mu^{(k)}\alpha^{1-\frac{\mu_{v,i}^{(k)}(j')}{\mu^{(k)}}}(2^{\frac{\rho_{j'}}{\mu^{(k)}}\log\alpha} - 1) \\
&\le \mu^{(k)}\alpha^{1-\frac{\mu_{v,i}^{(k)}(j')}{\mu^{(k)}}}\cdot\frac{\rho_{j'}}{\mu^{(k)}}\cdot\log\alpha \quad (10) \\
&= \alpha^{1-\frac{\mu_{v,i}^{(k)}(j')}{\mu^{(k)}}}\cdot\rho_{j'}\cdot\log\alpha, \quad (11)
\end{aligned}$$

where Ineq. (10) holds due to that $2^a - 1 \le a$ for $0 \le a \le 1$.

Similarly, for any cloudlet $v \in V$, we have $W_v(j'+1) - W_v(j') \le \beta^{1-\frac{C_v(j')}{C_v}}(\sum_{l=1}^{L_{j'}} C(f^{(\lambda(j',l))})\cdot x_v^{(\lambda(j',l))})\log\beta$ and for any link $e \in E$, we have $W_e(j'+1) - W_e(j') \le \gamma^{1-\frac{B_e(j')}{B_e}}\cdot\rho_{j'}\cdot\log\gamma$.

We then calculate the cost sum of all VNF instances when admitting request $r_{j'}$. The difference of the cost sum of VNF instances before and after admitting request $r_{j'}$ is

$$\begin{aligned}
&\sum_{v\in V}\sum_{l=1}^{L_{j'}}\sum_{i\in F_v^{(\lambda(j',l))}} W_{v,i}^{(k)}(j'+1) - W_{v,i}^{(k)}(j') \\
&= \sum_{l=1}^{L_{j'}} W_{v,i}^{(k)}(j'+1) - W_{v,i}^{(k)}(j') \quad (12) \\
&\le \sum_{l=1}^{L_{j'}} \alpha^{1-\frac{\mu_{v,i}^{(\lambda(j',l))}(j')}{\mu^{(\lambda(j',l))}}}\cdot\rho_{j'}\cdot\log\alpha, \quad \text{by Ineq. (11)} \\
&= \rho_{j'}\cdot\log\alpha\left(\sum_{l=1}^{L_{j'}}(\alpha^{1-\frac{\mu_{v,i}^{(\lambda(j',l))}(j')}{\mu^{(\lambda(j',l))}}} - 1) + \sum_{l=1}^{L_{j'}} 1\right) \\
&= \rho_{j'}\cdot\log\alpha\left(\sum_{l=1}^{L_{j'}}\omega_{v,i}^{(k)}(j') + L_{j'}\right) \le 2n\rho_{j'}\cdot\log\alpha \quad (13)
\end{aligned}$$

Eq. (12) holds due to that for each network function $f_{j',l}$, only one VNF instance is employed to process data traffic of

request $r_{j'}$. Ineq. (13) holds due to the fact that if request $r_{j'}$ is admitted, the admission control policy is met, i.e., $\sum_{l=1}^{L_{j'}} \omega_{v,i}^{(\lambda(j',l))}(j') \leq \sigma_1 = n$, and the length of service function chain of request $r_{j'}$ is less than the number of APs, i.e., $|SFC_{j'}| = L_{j'} \leq L_{max} \leq n$.

Similarly, the difference of the cost sum of cloudlets before and after admitting request $r_{j'}$ is $\sum_{v\in V} W_v(j'+1) - W_v(j') \leq 2nL_{j'} \cdot C(f_{max}) \cdot \log\beta$, where $C(f_{max})$ is the maximum computing resource consumption of any VNF instance $f^{(k)}$, $1 \leq k \leq K$ in the system. And the difference of the cost sum of links before and after admitting request $r_{j'}$ is $\sum_{e\in E} W_e(j'+1) - W_e(j') \leq 2n\rho_{j'} \cdot \log\gamma$.

The cost sum of VNF instances for request admissions when r_j arrives thus is

$$\begin{aligned}
&\sum_{v\in V}\sum_{l=1}^{L_j}\sum_{i\in F_v^{(k)}} W_{v,i}^{(k)}(j) \\
&= \sum_{j'=1}^{j-1}\sum_{v\in V}\sum_{l=1}^{L_{j'}}\sum_{i\in F_v^{(k)}} W_{v,i}^{(k)}(j'+1) - W_{v,i}^{(k)}(j') \qquad (14)\\
&= \sum_{r_{j'}\in\mathcal{A}(j)}\sum_{v\in V}\sum_{l=1}^{L_{j'}}\sum_{i\in F_v^{(\lambda(j',l))}} (W_{v,i}^{(k)}(j'+1) - W_{v,i}^{(k)}(j')) \\
&\leq \sum_{r_{j'}\in\mathcal{A}(j)} 2n\rho_{j'}\cdot\log\alpha = 2n\log\alpha\cdot\mathbb{B}(j), \quad \text{by Ineq. (13)}
\end{aligned}$$

where Eq. (14) follows from the fact that if a request is not admitted, none of the processing capacity of any VNF instance will be consumed.

Similarly, the cost sum of cloudlets for request admissions when r_j arrives is $\sum_{v\in V} W_v(j) \leq 2nL_{max}\log\beta \cdot |\mathcal{A}(j)| \cdot C(f_{max})$. And, the cost sum of links for request admissions when r_j arrives is $\sum_{e\in E} W_e(j) \leq 2n\log\gamma\cdot\mathbb{B}(j)$.

We now provide a lower bound on the weight of a rejected request by `Algorithm 2` but admitted by an optimal offline algorithm denoted by OPT. Before we proceed, we choose appropriate values for α, β, and γ prior to the arrival of any request r_j and VNF instance k, $1 \leq k \leq K$ as follows.

$$2n+2 \leq \alpha \leq \min_{1\leq k\leq K}\{2^{\frac{\mu^{(k)}}{\rho_j}}\} \qquad (15)$$

$$2n+2 \leq \beta \leq \min_{1\leq k\leq K}\min_{v\in V}\{2^{\frac{C_v}{C(f^{(k)})}}\} \qquad (16)$$

$$2n+2 \leq \gamma \leq \min_{e\in E}\{2^{\frac{B_e}{\rho_j}}\} \qquad (17)$$

Lemma 2: Let $\mathcal{T}(j)$ be the set of requests that are rejected by `Algorithm 2` but admitted by the optimal offline algorithm OPT prior to the arrival of request r_j. Then, for any request $r_{j'} \in \mathcal{T}(j)$, we have $\sum_{v\in V}\sum_{l=1}^{L_{j'}}\sum_{i\in F_v^{(\lambda(j',l))}} \omega_{v,i}^{(\lambda(j',l))}(j') + \sum_{v\in V}\omega_v(j') + \sum_{e\in E}\omega_e(j') > \min\{\sigma_1,\sigma_2,\sigma_3\} = n$.

The proof of Lemma 2 is omitted, due to space limitation.

Theorem 2: Given an MEC network $G=(V,E)$ with a set V of APs in which each $v\in V$ is attached with a cloudlet with computing capacity C_v, each link $e\in E$ has bandwidth capacity B_e, there is an online algorithm, `Algorithm 2`, with competitive ratio of $O(\log n)$ for the online multicasting throughput maximization problem, and the algorithm takes $O((L_j\cdot|V|)^{\frac{1}{\epsilon}}|D_j|^{\frac{2}{\epsilon}})$ time to admit each request r_j when $\alpha=\beta=\gamma=O(n)$, where $n=|V|$, $L_j=|SFC_j|$, and ϵ is a constant with $0<\epsilon\leq 1$.

Proof Denote by D_{max} and ρ_{max} the maximum cardinality of destination set $D_{j'}$ and the maximum bandwidth requirement of request $r_{j'}$ among all requests respectively, prior to the arrival of request r_j, i.e., $D_{max} = \max_{1\leq j'\leq j}\{D_{j'}\}$, and $\rho_{max} = \max_{1\leq j'\leq j}\{\rho_{j'}\}$. We first analyze the competitive ratio of the proposed online algorithm. We here abuse the notation OPT to denote the optimal offline algorithm OPT and the number of requests admitted by it. Let $\mathcal{A}(j)$ be the set of admitted requests when request r_j arrives, we have

$$\begin{aligned}
&\frac{n}{D_{max}^{\epsilon}}(OPT - |\mathcal{A}(j)|) \\
&\leq \frac{n}{D_{max}^{\epsilon}}\sum_{r_{j'}\in\mathcal{T}(j)} 1 \leq \sum_{r_{j'}\in\mathcal{T}(j)} n \qquad (18)\\
&\leq \sum_{r_{j'}\in\mathcal{T}(j)}\left(\sum_{v\in V}\sum_{l=1}^{L_{j'}}\sum_{i\in F_v^{(\lambda(j',l))}}\omega_{v,i}^{(\lambda(j',l))}(j') + \sum_{v\in V}\omega_v(j') + \sum_{e\in E}\omega_e(j')\right) \\
&\leq \sum_{r_{j'}\in\mathcal{T}(j)}\sum_{v\in V}\sum_{l=1}^{L_j}\sum_{i\in F_v^{(\lambda(j,l))}}\omega_{v,i}^{(\lambda(j,l))}(j) + \\
&\quad \sum_{r_{j'}\in\mathcal{T}(j)}\sum_{v\in V}\omega_v(j) + \sum_{r_{j'}\in\mathcal{T}(j)}\sum_{e\in E}\omega_e(j), \qquad (19)\\
&= \sum_{r_{j'}\in\mathcal{T}(j)}\sum_{v\in V}\sum_{l=1}^{L_j}\sum_{i\in F_v^{(\lambda(j,l))}}\frac{W_{v,i}^{(\lambda(j,l))}(j)}{\mu^{(\lambda(j,l))}} + \\
&\quad \sum_{r_{j'}\in\mathcal{T}(j)}\sum_{v\in V}\frac{W_v(j)}{C_v} + \sum_{r_{j'}\in\mathcal{T}(j)}\sum_{e\in E}\frac{W_e(j)}{B_e} \\
&= \sum_{v\in V}\sum_{l=1}^{L_j}\sum_{i\in F_v^{(\lambda(j,l))}} W_{v,i}^{(\lambda(j,l))}(j)\sum_{r_{j'}\in\mathcal{T}(j)}\frac{1}{\mu^{(\lambda(j,l))}} + \\
&\quad \sum_{v\in V}W_v(j)\sum_{r_{j'}\in\mathcal{T}(j)}\frac{1}{C_v} + \sum_{e\in E}W_e(j)\sum_{r_{j'}\in\mathcal{T}(j)}\frac{1}{B_e} \qquad (20)\\
&\leq \sum_{v\in V}\sum_{l=1}^{L_j}\sum_{i\in F_v^{(\lambda(j,l))}} W_{v,i}^{(\lambda(j,l))}(j) + \sum_{v\in V}W_v(j) + \sum_{e\in E}W_e(j) \qquad (21)\\
&\leq 2n\mathbb{B}(j)\log\alpha + 2nL_{max}C(f_{max})\log\beta\cdot|\mathcal{A}(j)| + 2n\mathbb{B}(j)\log\gamma \\
&\leq 2n|\mathcal{A}(j)|\big(\rho_{max}\log\alpha + L_{max}\cdot C(f_{max})\log\beta + \rho_{max}\log\gamma\big) \qquad (22)
\end{aligned}$$

Ineq. (18) holds since $D_{max} \geq 1$, and $0 < \epsilon \leq 1$, thus $D_{max}^{\epsilon} \geq 1$. Ineq. (19) holds since the resource utilization ratio does not decrease and thus the usage cost of each VNF instance, each cloudlet, and each link does not decrease with more request admissions. Ineq. (20) holds because $\sum_{i=1}^{m}\sum_{j=1}^{n} A_i\cdot B_j \leq \sum_{i=1}^{m} A_i \cdot \sum_{j=1}^{n} B_j$, for all $A_i \geq 0$ and $B_j \geq 0$. Ineq. (21) holds because all algorithms, including the optimal offline algorithm OPT, the accumulated usage of resources in any VNF instance, cloudlet and link is no greater than its capacity. Recall that $\mathcal{A}(j)$ is the set

of requests admitted by Algorithm 2, and $\mathcal{T}(j)$ is the set of requests rejected by Algorithm 2 but accepted by the optimal offline algorithm OPT. We have $\frac{OPT-|\mathcal{A}(j)|}{|\mathcal{A}(j)|} \leq 2D^{\epsilon}_{max}(\rho_{max}\log\alpha + L_{max}\cdot C(f_{max})\log\beta + \rho_{max}\log\gamma)$.

Thus, we have $\frac{OPT}{|\mathcal{A}(j)|} \leq 2D^{\epsilon}_{max}(\rho_{max}\log\alpha + L_{max}\cdot C(f_{max})\log\beta + \rho_{max}\log\gamma) + 1 = O(\log n)$ when $\alpha = \beta = \gamma = O(n)$.

The time complexity analysis of Algorithm 2 is omitted, due to space limitation.

VI. Performance evaluation

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

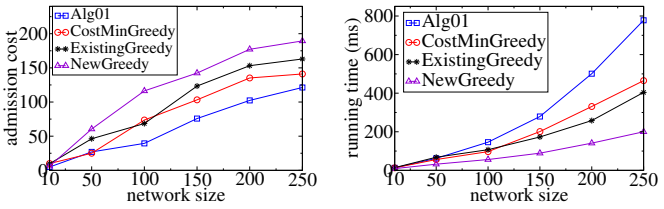
A. Experiment settings

We consider an MEC network $G = (V, E)$ consisting of from 10 to 250 APs (cloudlets). The computing capacity of each cloudlet ranges from $3,000$ MHz to $6,000$ MHz [10], and the bandwidth capacity of each link varies between $2,000$ $Mbps$ and $20,000$ $Mbps$ [11]. The number K of different types of network functions is set at 20. The computing resource demand of each network function is set from 300 MHz to 600 MHz, and their processing rate is randomly drawn from 50 to 100 data packets per millisecond [14]. Given a cloudlet, the instantiation cost of a VNF instance in it is randomly drawn from $[0.50, 2.0]$, while the processing cost of per packet data traffic by a VNF instance is a random value drawn from $[0.01, 0.1]$ [16]. The routing cost per data packet along a link is a value drawn from $[0.01, 0.1]$. For each generated request r_j, one AP node in V is randomly selected as its source s_j, and a set of AP nodes in V are randomly selected as its destinations D_j. Its data packet rate is drawn from 2 to 10 packets per millisecond, where each packet is of size $64KB$. The length of its service function chain is set from 1 to 10, and each network function is randomly drawn from the K types. The value in each figure is the mean of the results out of 30 MEC instances of the same size. The running time of an algorithm is obtained on a machine with 3.4GHz Intel i7 Quad-core CPU and 16GB RAM. Unless otherwise specified, these parameters will be adopted in the default setting.

In the following, we first evaluate the performance of Algorithm 1 for the NFV-enabled multicasting problem against three baseline heuristics CostMinGreedy, ExistingGreedy, and NewGreedy. Algorithm CostMinGreedy considers network functions in the service function chain one by one for each arriving request, it always choose the cloudlet that can achieve the the minimum cost (including the processing cost, instance instantiation cost, and routing cost) for the next network function. Algorithm ExistingGreedy considers network functions one by one and it tries to admit the request by existing VNF instances with the minimum admission cost as long as there is a VNF instance with sufficient residual processing capacity, while algorithm NewGreedy always tries to create new VNF instance for the request providing sufficient computation resource in a cloudlet. We then evaluate the performance of Algorithm 2 against a benchmark OnlineLinear for online request admissions, where for each request, algorithm OnlineLinear first removes all VNF instances, cloudlets and links that do not have sufficient residual resources to support the admission of the request, and then assign a cost of one to each VNF instance, each cloudlet, and each link. It then constructs an auxiliary graph and finds a single-source shortest path tree rooted at the source node and spanning all destination nodes of the multicast request.

B. Performance evaluation

We first investigate the performance of Algorithm 1 against that of three baseline heuristics CostMinGreedy, ExistingGreedy, and NewGreedy, for the NFV-enabled multicasting problem for a single request admission, by varying the network size from 10 to 250. Fig. 4 illustrates the admission cost and running time of the four mentioned algorithms. From Fig. 4 (a), we can see that Algorithm 1 achieves a much lower admission cost than its three benchmarks in all cases. Specifically, it can incurs 48.0%, 25.3%, and 14.1% less admission cost than that by NewGreedy, ExistingGreedy, and CostMinGreedy, respectively, when the network size is 250. The reason behinds is that Algorithm 1 jointly considers the placement of VNF instances and data traffic routing for multicast request admission, as well as makes a fine decision between using an existing VNF instance or creating a new instance. Fig. 4 (b) demonstrates the running time of the four algorithms. It can be seen that algorithm NewGreedy achieves the least running time, as it gives priority to create new VNF instances in cloudlets, thus much smaller solution space is explored. Algorithm 1 achieves the highest running time due to the fact that Algorithm 1 strives for finding a multicast tree with the least cost while passing through VNFs in its service function chain at the same time, while the counter parts only try to place its VNF instances in a greedy way, then routing data packets to the destinations.

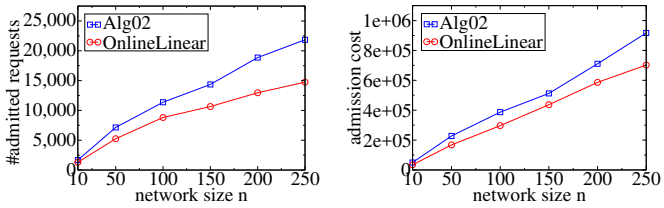


(a) The admission cost (b) The running time

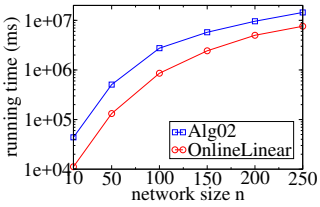
Fig. 4. Performance of Algorithm 1, CostMinGreedy, ExistingGreedy, and NewGreedy.

We then evaluate the performance of Algorithm 2 by varying the network size from 10 to 250 for a sequence of $40,000$ requests. Fig. 5 plots the performance curves of different algorithms, from which we can see that Algorithm 2 outperforms the baseline algorithm OnlineLinear in all cases. Specifically, Algorithm 2 admits 35.4% more requests than that by algorithm OnlineLinear when the net-

work size is 200. The rationale behinds is that Algorithm 2 applies a cost model to assign higher cost to over-utilized resources and assign lower cost to under-utilized resources, thus allocating resources more evenly, while algorithm OnlineLinear does not take into account the utilization of resources on each cloudlet and each link, thus leading to overloads on some links and cloudlets. Fig. 5 (c) shows the running time of the two algorithms. We can see that the running time of Algorithm 2 is higher than that of algorithm OnlineLinear, this is because Algorithm 2 admits more requests, and assigns exponential weights to cloudlets and links when each request arrives.



(a) The network throughput (b) The admission cost



(c) The running time

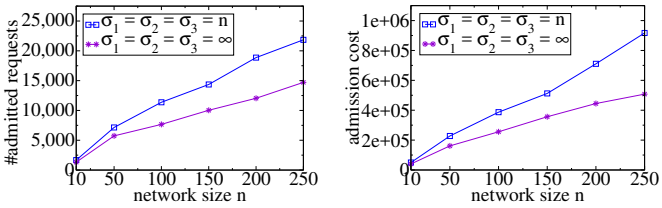
Fig. 5. Performance of Algorithm 2 and OnlineLinear by varying the network size from 10 to 250.

C. Parameter impact on the algorithm performance

We finally study the impact of the admission control variables σ_1, σ_2, and σ_3 on the performance of Algorithm 2. Fig. 6 demonstrates the performance curves of Algorithm 2 with and without the admission control thresholds, from which it can be seen that less requests can be admitted if no admission control policy is applied. We can see that when the network size is 100, Algorithm 2 can admit 38.4% more requests than that if no admission control is applied. Furthermore, the performance gap of Algorithm 2 with and without the thresholds becomes larger and larger with the increase in network size. This is due to that in larger networks, the size of the destination set of a request can be very large, and the distance between the source and a destination node can be very long, thus consuming much more bandwidth resource for routing its data traffic. Under the admission control policy, Algorithm 2 is able to reject those requests beyond the given threshold, thereby enable to admit more future requests and achieving a higher throughput. Fig. 6 (b) shows the admission cost of Algorithm 2 with and without admission control thresholds.

VII. Conclusion

In this paper, we studied the online NFV-enabled multicast request admissions in a mobile edge cloud network. We first proposed an approximation algorithm for finding a minimum cost multicast tree for a single request. We then devised an online algorithm with a provable competitive ratio for the online throughput maximization problem where NFV-enabled multicast requests arrive one by one without the knowledge of future request arrivals. We finally evaluated the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising, and outperform their theoretical counterparts.



(a) The network throughput (b) The admission cost

Fig. 6. Impact of the admission control policy on the performance of Algorithm 2.

References

[1] N. Abbas *et al.* Mobile edge computing: a survey. *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450 – 465, 2018.

[2] X. Chen, L. Jiao, W. Li, and X. Fu. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795 – 2808, 2016.

[3] A. Ceselli *et al.* Mobile edge cloud network design optimization. *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1818 – 1831, 2017.

[4] M. Charikar *et al.* Approximation algorithms for directed Steiner problems. *J. Algorithms*, vol. 33, no. 1, pp. 73 – 91, Elsevier, 1998.

[5] N. Chowdhury and R. Boutaba. Network virtualization: state of the art and research challenges. *IEEE Commu. Maga.*, pp. 20 – 26, 2009.

[6] H. Feng *et al*. Approximation algorithms for the nfv service distribution problem. *Proc. of INFOCOM*, IEEE, 2017.

[7] T. He *et al*. It's hard to share: joint service placement and request scheduling in edge clouds with sharable and non-sharable resources. *Proc. of ICDCS*, IEEE, 2018.

[8] M. Jia, J. Cao, and W. Liang. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 725 – 737, 2017.

[9] M. Jia, W. Liang, Z. Xu, and M. Huang. Cloudlet load balancing in wireless metropolitan area networks. *Proc. of INFOCOM*, IEEE, 2016.

[10] M. Jia *et al.* QoS-aware task offloading in distributed cloudlets with virtual network function services. *Proc. of MSWiM*, ACM, 2017.

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

[12] P. Mach and Z. Becvar. Mobile edge computing: a survey on architecture and computation offloading. *IEEE Commu. Surveys & Tutorials*, vol. 19, no. 3, pp. 1628 – 1656, Jun. 2017.

[13] Y. Mao *et al*. A survey on mobile edge computing: the communication perspective. *IEEE Commun. Surv. Tutor.*, vol. 19, pp. 2322 – 2358, 2017.

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

[15] S. V. Rossem *et al.* Deploying elastic routing capability in an SDN/NFV-enabled environment. *2015 IEEE NFV-SDN*, pp. 22 – 24, Nov. 2015.

[16] Z. Xu *et al*. Throughput maximization and resource optimization in NFV-enabled networks. *Proc. of ICC'17*, IEEE, 2017.

[17] Z. Xu *et al*. Approximation and online algorithms for NFV-enabled multicasting in SDNs. *Proc. of ICDCS'17*, IEEE, 2017.

[18] Z. Xu *et al*. Efficient NFV-enabled multicasting in SDNs. *IEEE Transactions on Communications*, vol. 67, no. 3, pp. 2052 – 2070, 2019.

[19] S. Q. Zhang *et al*. Network function virtualization enabled multicast routing on SDN. *Proc. of ICC*, IEEE, 2015.