Cost-Efficient NFV-Enabled Mobile Edge-Cloud for Low Latency Mobile Applications

Binxu Yang, Wei Koong Chai, *Member, IEEE,* Zichuan Xu, *Member, IEEE,* Konstantinos V. Katsaros, *Member, IEEE,* and George Pavlou, *Fellow, IEEE*

***Abstract*—Mobile edge-cloud (MEC) aims to support low latency mobile services by bringing remote cloud services nearer to mobile users. However, in order to deal with dynamic workloads, MEC is deployed in a large number of fixed-location micro-clouds, leading to resource wastage during stable/low workload periods. Limiting the number of micro-clouds improves resource utilization and saves operational costs, but faces service performance degradations due to insufficient physical capacity during peak time from nearby micro-clouds. To efficiently support services with low latency requirement under varying workload conditions, we adopt the emerging Network Function Virtualization (NFV)-enabled MEC, which offers new flexibility in hosting MEC services in any virtualized network node, e.g., access points, routers, etc. This flexibility overcomes the limitations imposed by fixed-location solutions, providing new freedom in terms of MEC service-hosting locations. In this paper, we address the questions on *where* and *when* to allocate resources as well as *how many* resources to be allocated among NFV-enabled MECs, such that both the low latency requirements of mobile services and MEC cost efficiency are achieved. We propose a dynamic resource allocation framework that consists of a fast heuristic-based incremental allocation mechanism that dynamically performs resource allocation and a reoptimization algorithm that periodically adjusts allocation to maintain a near-optimal MEC operational cost over time. We show through extensive simulations that our flexible framework always manages to allocate sufficient resources in time to guarantee continuous satisfaction of applications' low latency requirements. At the same time, our proposal saves up to 33% of cost in comparison to existing fixed-location MEC solutions.**

***Index Terms*—Mobile edge-cloud, low latency applications, dynamic resource allocation, approximation algorithm.**

I. Introduction

OVER the last decade, advances in wireless access technologies (e.g., WiFi and LTE) have enabled an explosion of resource-hungry mobile applications, challenging current mobile devices' processing ability. In particular, mobile multimedia services with stringent latency requirements (in the order of hundreds of milliseconds [1]), such as augmented reality (AR), high-definition video streaming, gaming and face recognition, are computationally expensive for today's mobile devices; resulting in fast exhaustion of battery life and long processing delays [2]. Conventional cloud solutions [3], where users exploit preallocated service instances from data center-based clouds to process computationally expensive tasks, address the issue of computational resources, but suffer from long network latencies [4]. On the other hand, mobile edge-cloud (MEC) (also known as *cloudlet* [4], *fog computing* [5], *Telco cloud* [6], *follow-me cloud* [7]) mitigates the long network latency issue by deploying dedicated micro-clouds along with service instances at network locations that are closer to users, e.g., access points (APs), routers, etc.

However, since the micro-clouds are deployed at *fixed* locations and have limited physical resources (especially compared to data center-based clouds), they are deployed to large number of APs with MEC service instances in each micro-cloud [8]. This achieves low latency at the expense of significant operational costs due to break of data center (DC) consolidation [2], [3]. Limiting the number of micro-clouds can save operational costs, but faces challenges in dynamically supporting low latency services with limited resources at static network locations. For instance, current resource allocation techniques to deal with workload elasticity, such as auto-scaling [9], [10], could only scale up to the physical capacity limit of micro-clouds. Subsequently, if there is no micro-cloud in the vicinity of the overloaded one that can provide more computational resources for load balancing, users' tasks would accumulate, leading to the violation of the required service response time (e.g., time spent in network and edge clouds).

Recently, Network Function Virtualization (NFV) was proposed to facilitate network function deployment for Internet service providers (ISPs) [11]. It decouples network functions from the underlying hardware and implements them as software in virtual machines (VMs) hosted in commodity servers. The advent of NFV promotes the emerging concept of NFV-enabled MEC (e.g., [12], [13]) whereby services can be hosted at any network location that has virtualized resources, e.g., provided by commodity servers. Such NFV-enabled MEC model enables real-time instantiation (e.g., VM instantiation time for Unikernel [14] and ClickOS [15] are in the order of tens of milliseconds) of MEC at new network locations to host edge services, and also allows MEC to scale up/down computational resources to accommodate user demand variations. As a result, the MEC can be dynamically instantiated at network locations that efficiently utilize ISPs' virtual network infrastructures and thereby maintaining low operational costs overtime. However, such flexibility in resource allocation faces challenges in:

- *Dynamically deriving the MEC service-hosting locations,*

B. Yang and G. Pavlou are with the Department of Electronic and Electrical Engineering, University College London, London, WC1E 7JE, U.K. (e-mails: binxu.yang.13@ucl.ac.uk, g.pavlou@ucl.ac.uk).

Z. Xu is with the School of Software, Dalian University of Technology, Dalian, Liaoning, China, 116620 (e-mail: z.xu@dlut.edu.cn).

W. K. Chai is with the Department of Computing and Informatics, Bournemouth University, Dorset, BH12 5BB, U.K. (e-mail: wchai@bournemouth.ac.uk).

K. V. Katsaros is with Intracom Telecom, Athens, Greece. (e-mail: konkat@intracom-telecom.com).

amount of resources and the corresponding network paths to mobile users such that the resulting network access latencies are within the network latency requirements and the ISPs' virtualized network resources are optimally utilized.
- *Determining the appropriate time instance* to perform dynamic resource allocation in order to avoid computation congestion at VMs due to peak load [16].
- *Performing resource allocation in a timely manner* such that the time spent in deriving a resource allocation decision does not affect low latency MEC services.

In this work, we take into account the flexibility afforded by NFV along with the abovementioned challenges, and study the problem of dynamic resource allocation in MEC, aiming at minimizing operational costs while satisfying users' low latency service response time requirements.

For the above problem, we propose a novel dynamic resource allocation framework for NFV-enabled MEC that consists of an online heuristic-based incremental allocation mechanism and a global resource reoptimization algorithm to address the trade-off between cost efficiency and low latency requirement. In particular, our online heuristic-based incremental allocation mechanism aims to efficiently allocate resources to tackle local MEC computation congestion due to (sudden) increase of workload in a timely manner. It consists of (1) an initial offline MEC resource allocation based on expected workload that achieves the desired service response time with the minimum required computational resources, (2) an auto-scaling and load balancing (ALB) mechanism that accommodates workload variations, (3) a *capacity violation detection* (CVD) mechanism that derives the projected time when ALB fails to cope with service elasticity and (4) a network latency constraint greedy (NLCG) algorithm of polynomial complexity to derive a new NFV-enabled node as MEC service-hosting node which supports the stringent latency requirement. Since our online allocation mechanism computes local MEC resource allocation, we also design a set cover partition approximation (SCPA) algorithm that operates in parallel with NLCG to *globally* reoptimize the locations and allocated resources while achieving a guaranteed operational cost. Given user demands, this cost is no more than $\ln(N)$ times of optimal MEC operational cost, where N is the largest number of APs that are served by a MEC service-hosting node among all instantiated MECs.

To demonstrate the effectiveness of our proposed framework, we carry out an extensive simulations with realistic three-layer cellular network setup [17]. We use real mobility traces from [18] to show the cost reduction brought by NFV-enabled flexible MEC instantiation compared to fixed-location MEC. Further, we conduct an in-depth cost efficiency impact factor analysis to give detailed insights into the design of online MEC resource allocation framework under various network topologies, latency requirements and server capacities.

Our study here is based on our preliminary work in [19]. The main contributions of this study are as follows.

1) We formulate and solve the dynamic MEC resource allocation problem as an integer linear programming (ILP) problem taking into account the flexibility in the determination of MEC locations enabled by NFV (see Section III-B) and the trade-off between service response time and operational costs. To the best of our knowledge, this is the first study focusing on the dynamic MEC resource allocation taking into account the possibility of NFV-enabled MEC service instantiations.
2) We design a dynamic resource allocation framework consisting of a fast heuristic-based incremental allocation mechanism and a SCPA reoptimization algorithm for low-cost MEC resource allocation framework (see Section IV). Both NLCG and SCPA algorithms are general in nature and applicable to any online edge cloud systems (e.g., for different hosted services, edge cloud capacities and VM technologies). In addition, we mathematically prove that given user demands, our SCPA algorithm results in no more than $\ln(N)$ times of optimal MEC operational cost in polynomial time.
3) We demonstrate the effectiveness of our framework (see Section V) through extensive simulations. We show that our framework achieves 33% cost reduction compared to fixed-location MEC overprovisioning solutions. Further, our in-depth impact factor analysis shows that SCPA achieves cost efficiency within 20% of the lower bound of the optimal solution, under different network size, services' latency requirements and MEC server capacities.

II. Related Work

Cost efficiency in cloud computing is an important topic that has received wide attention. One branch of studies in this regard focuses on energy efficiency in DCs [3], [20]. In particular, various dynamic workload-to-VM placement algorithms have been proposed to minimize system operational costs by minimizing the number of active physical machines. However, these work do not consider the deadlines of computational tasks, which is a key requirement for low latency services [21]. In addition, DCs are considered to be rich in terms of computational resources at a single location, whereas MECs are distributed and have limited computational resources. Thus, these solutions are not suitable for our problem.

Early work targeting specifically MEC focused on offloading technologies [4], [22] which later shifted to problems on dynamic state migration between micro-clouds [8], [23] whereby the investigations focused on the decision on whether and where to migrate user states in VMs due to user mobility. More recently, the research focus in MEC further shifted to resource allocation and micro-cloud placement. These work can be categorized into offline (e.g., static) and online (e.g., dynamic) problems. Specifically, [17], [24], [25] consider static network planning problems in metropolitan area networks where the authors investigated the optimal static placement of micro-clouds with objectives such as minimizing the system costs or the end-to-end latencies. For instance, [24], [25] formulated the static micro-cloud placement problem into a K-median problem such that the average end-to-end latency of all users is minimized. In addition, they proposed an online user request assignment algorithm that dynamically decides the routes between users and the K micro-clouds.

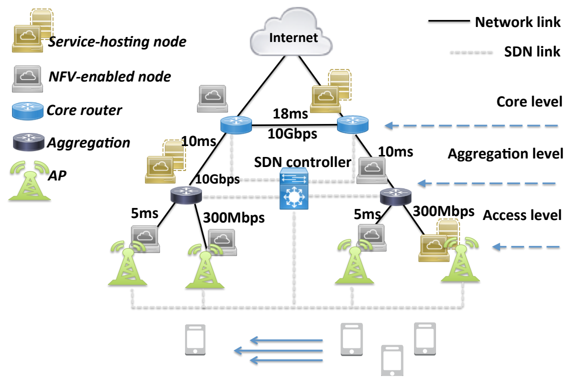


Fig. 1: (Color Online) Hierarchical MEC system model.

Similar to [24], [25], work in [17] studied a static micro-cloud placement problem while additionally taking potential migrations into account. The authors also investigated the dynamic routing problem given the derived initial micro-cloud locations. Nonetheless, these early work related to the placement of micro-clouds did not consider the possibility of flexible MEC service instantiations enabled by NFV. Clearly, the performance improvement achieved through online routing and load balancing [26] is limited by the fixed number and locations of micro-clouds. Finally, [13] considered flexible micro-cloud instantiations where content distribution network providers dynamically discover edge locations in different ISP networks to improve the performance and reduce MEC costs. Nevertheless, application latency requirements are ignored. Moreover, this investigation looked into a different resource allocation granularity whereby the allocation is performed at autonomous system level.

Online resource allocation proposals are investigated either in the form of online admission control or the online service placement problem in MEC. [27], [28] and [29] considered a resource-constrained MEC scenario where they devised online resource allocation schemes to determine how much resources to allocate to each user or which users to serve. In particular, they considered pre-determined micro-cloud locations that have fixed amount of overall resources to be allocated. In contrast, our proposal can be seen as an alternative solution to their problem, as we increase the overall allocated resources at new network locations (rather than selecting which users to serve) such that all users can be served.

Apart from offloading computational tasks to fixed network locations (e.g., our approach), computational tasks can also be offloaded to nearby mobile devices, known as mobile ad-hoc clouds [30]. The primary advantage of using mobile ad-hoc clouds is to exploit its intrinsic mobility to enable a flexible on-demand resource provisioning by scheduling devices to move to certain geographic locations [30]. However, unlike conventional mobile edge clouds where clouds are fixed and managed by cloud operators, the discovery and management of dynamic mobile ad-hoc clouds would introduce extra system complexity. In addition, the fully distributed cloud architecture would result in a worse cost efficiency than our approach due to the complete break of DC consolidation [2].

III. System Model and Problem Formulation

A. System Model

We consider a typical three-layer hierarchical wireless metropolitan area network [17] that consists of APs, aggregation nodes and metropolitan level mobile core network nodes (illustrated in Fig. 1). Each AP is connected to a single aggregation node which is connected to one mobile core node. Furthermore, the connectivity between mobile core nodes depends on the actual mobile core network's topology. For most of real-world topologies, a mobile core node has at least one network link towards other mobile core nodes (e.g., a topology example is shown in Fig. 2). We use $G = (V, E)$ to denote this network, where V is the set of network nodes and E is the set of links. Further, let B denote the set of APs, $b \in B$, which is a subset of network nodes ($B \subset V$). We consider that each network node is equipped with a commodity server [11], which has limited computational resources, k_v (e.g., CPU[1]) to host application service providers' services as software via VMs. Such support of MEC services with NFV-enabled nodes necessitates NFV commodity servers to be active (e.g., active servers are shown in Fig. 1 as service-hosting nodes) and hence, incurs operational costs (e.g., energy consumption) [3]. For the rest of paper, we consider MEC nodes to be any NFV-enabled network nodes on which MEC services can be hosted with allocated VMs.

Given the NFV-enabled MEC, mobile users upload raw files at discrete time, $t \in T$, through their associated APs to MEC nodes for processing rather than executing service instances locally in their mobile devices. The user requests from an AP are served by VMs at a single MEC node through the same path, $p_{bv} \in P_{bv}$, between AP b and node v (v is the selected node to host the required service)[2], where P is the set of paths between pair of nodes in V and $P_{bv} \subseteq P$. We use A_b^t to denote the total load incurred by mobile users at AP, b at time t, which results in bandwidth consumption, w_b^t, of flows departing from AP b. At the same time, user flows consume computational resources from MEC nodes, which depend on the AP-to-MEC assignment.

We consider stateless mobile services (e.g., AR, etc.) to be pre-installed as software into NFV-enabled nodes [1], [31]. That is, user requests can be seamlessly served by VMs at different MEC node without requiring service state migration since the services are stateless. In addition, NFV-enabled nodes that are not serving as MEC nodes can instantiate VMs to support stateless MEC services in a timely manner. This is due to the latest advances in VM technology such as Unikernel [14] and ClickOS [15], whereby the VM instantiation time could be reduced to tens of milliseconds (e.g., 30ms [3]). We summarize the notations used in this paper in Table I.

B. Problem Definition

Given the abovementioned system model and the flexible instantiation of MEC nodes, we consider the MEC opera-

[1]We only consider CPU as computational resources in this work.

[2]Multiple network paths between b and v could exist due to connectivity between mobile core nodes (see Fig. 2).

[3]Unikernel, designed for edge computing environment, achieves 30ms by exploiting a shared memory channel to optimize the VM instantiation time.

TABLE I: Notations

Symbol	Notations
V, E, B	Set of NFV-enabled nodes, edges and APs
P,P_{bv}	Set of paths, set of paths between b and v
k_v	Resource capacity at node v
w_b	Bandwidth consumption at AP b
BW_e	Bandwidth capacity at network link e
A_b^t	User computational resource demand from AP b at time t
D	Maximum network latency (hops) constraint
d_{bv}	Network hop distance between AP b and node v
N_b	The set of v that are located less than D network hops to b $N_b = \{v \mid d_{bv} \le D\}$
$\mathcal{AP}_v$	The set of APs covered by network node v
$\mathcal{AP}_{v'v}$	The set of APs covered by network node v' and v
L_v	The excess workload from node v
$X_{p_{bv}}$	The path decision variable for $p_{bv} \in P_{bv}$
Y_v	The MEC node decision variable for v

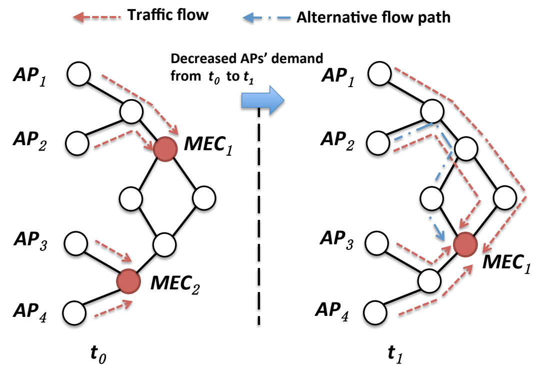


Fig. 2: (Color Online) Example of MEC operational cost minimization problem.

tional cost minimization problem for stateless low latency mobile services, whereby the network locations that host MEC services and the corresponding network paths can be dynamically controlled to efficiently utilize ISPs' resources. To better illustrate this scenario, an example is given in Fig. 2. We can see that two MEC nodes are instantiated among all NFV-enabled nodes together with its selected network paths at t_0. In contrast, only one MEC node is instantiated for operational cost minimization at t_1 in response to the decreased demands from APs. Meanwhile, the network paths are accordingly changed at t_1.

In this work, we aim to concurrently answer four primary questions: given a time varying workload, resource-constrained distributed NFV-enabled network nodes and capacitated network links, (1) *where* and (2) *when* to allocate resources, (3) *how many* resources to be allocated among NFV-enabled nodes and (4) *which* network paths to use (e.g., between APs and MECs) such that the low latency requirements of mobile services are always satisfied while incurring the least operational cost. Without loss of generality, we assume in this work that all NFV-enabled commodity servers are identical (e.g., same specifications) and incur equal operational cost. Therefore, the operational cost minimization objective is equivalent to the minimization of number of active commodity servers (MEC node) [3].

We use ILP to formulate the problem with two binary decision variables, Y_v^t and $X_{p_{bv}}^t$, which represent respectively the location of MEC service (i.e., $Y_v^t = 1$ if at time t, v is chosen as the location of a MEC service and $Y_v^t = 0$ otherwise) and the path between b and v (i.e., $X_{p_{bv}}^t = 1$ if p_{bv} is chosen; $X_{p_{bv}}^t = 0$ otherwise). The objective function of the ILP is to minimize the number of selected MEC nodes, that is, the sum of $Y_v, v \in V$ at every discrete time instance[4], $t \in T$.

To satisfy the service latency requirement, we first decompose the request response time into the following:

1) *Network access time* – represents the time a MEC service request spent during network transmissions, which highly depends on the selection of network path, $X_{p_{bv}}^t$, between an AP and the selected MEC node. To model such delay, we assume that as long as the capacities of the constituent links in the selected network path are not violated by MEC flows, we can represent access delay as a function of network hops. Hence, in order to achieve a required network access time, both link capacity and the number of network hops that the request traverses need to be constrained.
2) *Service processing time* – refers to the time a VM uses to serve a request. We assume that as long as there is an available resource unit, and the request rate is lower than service rate, the processing delay is bounded and can be represented by a mean expected value that depends on the actual VM technology. To satisfy the processing time, we constrain the aggregated resource demands from APs that are served by MEC node at time t to be no more than its physical capacity limit. This ensures a fixed service time at all time by allocating a dedicated resource unit for each request.

The ILP problem is formulated as below:

$$\min \sum_{v \in V} Y_v^t, \forall t \in T, \tag{1}$$

Subject to

$$\sum_{b \in B} \sum_{v \in V} \sum_{p_{bv}(e) \in P_{bv}(e)} w_b^t X_{p_{bv}(e)}^t \le BW_e, \forall e \in E, \forall t \in T, \tag{2}$$

$$\sum_{p_{bv} \in P_{bv}} \sum_{v \in N_b} X_{p_{bv}}^t = 1, \forall b \in B, \forall t \in T, \tag{3}$$

$$\sum_{p_{bv} \in P_{bv}} \sum_{b \in B} A_b^t X_{p_{bv}}^t - k_v Y_v^t \le 0, \forall v \in V, \forall t \in T, \tag{4}$$

$$Y_v^t \in \{0, 1\}, \forall v \in V, \forall t \in T, \tag{5}$$

$$X_{p_{bv}}^t \in \{0, 1\}, \forall p_{bv} \in P_{bv}, \forall t \in T, \tag{6}$$

Constraint (2) guarantees that for all edges, the aggregated bandwidth consumption is less than the link capacity, BW_e, at every time instance, where $P_{bv}(e)$ denotes all paths between b and v that traverse edge, e; Constraint (3) guarantees that flows from the same b are assigned to the same MEC node v

[4]Note that by fixing $T = \{t_0\}$, the problem is reduced to a static placement problem mentioned in Section I.

where v is selected from the set of network locations $N_b = \{v | d_{bv} \leq D\}$ that are within the network latency constraint denoted as D; Constraint (4) guarantees that the aggregated demands from APs at time t, $\sum_{p_{bv} \in P_{bv}} \sum_{b \in B} A_b^t X_{p_{bv}}^t$, served by the selected MEC v is no more than its physical capacity limit k_v and Constraints (5)-(6) limit the decision variables to be either 0 or 1.

Our problem stated above is NP-hard. A relaxed version of our problem (i.e., without the bandwidth capacity constraints (2)) can be obtained from the capacitated set covering problem (CSCP)[5]. Since CSCP problem has been shown to be NP-hard [32], our problem is NP-hard too.

IV. Dynamic Resource Allocation Framework for NFV-enabled MEC

A. Overview

Our problem aims at deriving the optimal MEC locations, amount of resources and network paths to MECs in face of dynamic workloads to satisfy services' low latency requirements while minimizing the overall operational costs incurred within the time period, T. Offline solutions (e.g., overprovisioning) only solve the latency aspect of the problem while ignoring the possible high costs incurred due to inefficient resource utilization. Existing dynamic solutions are either based on local search or global optimization. The former derives the resource allocation in a timely manner by targeting specific network areas suffering from resource exhaustion which however often results in sub-optimal allocations. On the other hand, the latter takes demands across the whole network and is generally able to obtain better results at the cost of running time due to the large scale input from the entire network. Note that such long running time is not tolerable to online MEC as it would affect the performance of low latency services. To overcome the abovementioned issues suffered by most conventional approaches, we propose a novel dynamic optimization framework for NFV-enabled MEC that leverages both the local resource allocation and global re-allocation of resources to achieve a balanced trade-off between resource allocation optimality and algorithm's running time.

Fig. 3 presents the overview of our dynamic resource allocation framework.

1) *Heuristic-based incremental allocation mechanism* (see the right-side of Fig. 3) follows the local search principle and aims at deriving the minimum required resources for MEC in a timely manner in response to temporary workload increase. The idea is to first provision NFV-enabled MEC with the minimum (optimal) number of MECs to satisfy the average user demands. Then, it exploits conventional techniques for coping with (minor) service elasticity (i.e., ALB) to maintain the overall number of MECs at a relatively low level. At the same time, we detect the time point when these mitigation tools will reach their limits (i.e., this implies that the existing MECs have been fully utilized) and cause the MEC system to violate the service response time requirement of the considered service(s). In such event, the allocation of a new MEC node (e.g., within the network latency constraints of APs that overloaded one of the existing MECs) will be chosen from the neighbouring network nodes of the overloaded MEC nodes (e.g., not searching the entire network), and activated in time before service quality degrades. By limiting the search scope to within the overloaded network area for the new MEC node, we significantly reduce the algorithm's running time and avoid service response time violations due to computation congestion at MECs. However, the heuristic-based incremental allocation solution has a major disadvantage due to the fact that it only incrementally adds MEC nodes to the existing MEC nodes that are previously allocated. As a result, the MEC resource allocation may gradually deviate from the optimum over time due to its lack of consideration for global workload variations.
2) *SCPA global reoptimization* (see the left-side of Fig. 3) aims to overcome the disadvantages of heuristic-based incremental allocation by adjusting the allocated resources at a coarse-grained time granularity to a near-optimal state. SCPA is periodically performed in a less frequent manner. It takes the resulting MEC nodes from the incremental solution and globally adjusts the resource allocation to maintain low MEC operational costs[6] within a bounded resulting operational cost.

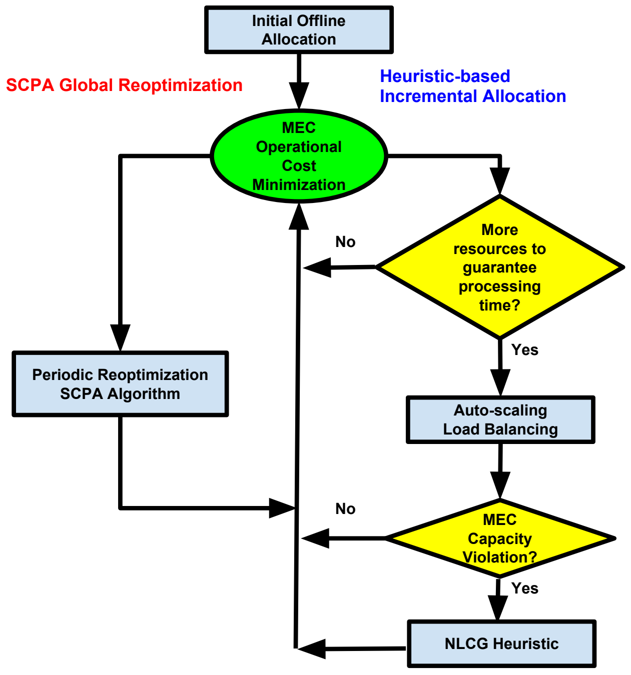

Initial Offline Allocation
SCPA Global Reoptimization
Heuristic-based Incremental Allocation
MEC Operational Cost Minimization
No
More resources to guarantee processing time?
Yes
Periodic Reoptimization SCPA Algorithm
Auto-scaling Load Balancing
No
MEC Capacity Violation?
Yes
NLCG Heuristic

Fig. 3: (Color Online) Dynamic resource allocation framework overview.

[5]In a capacitated set cover instance, we are given a universe X of n elements and a collection $\mathcal{S}$ of m subsets of X with elements having demand $d : X \mapsto \mathbb{R}^+$ and sets having supplies $s : \mathcal{S} \mapsto \mathbb{R}^+$, each subset has an associated cost; the objective is to pick the collection of sets $\mathcal{S}' \in \mathcal{S}$ of least total cost, such that each element $e \in X$ is contained in at least one set $S \in \mathcal{S}'$ while the supply of each set in $\mathcal{S}'$ is not violated [32].

[6]We do not consider migration costs as applications are stateless.

Next, we elaborate on how these two approaches jointly solve the MEC operational cost minimization problem while always conforming to the latency constraint. Our framework follows the procedure below.

1) We derive the initial optimal static MEC placement (i.e., the number of MECs is minimized) in an offline fashion by solving the static version of the problem[7] at time t_0 using CPLEX [33].
2) We leverage conventional ALB mechanisms to cope with service elasticity based on the initial or most current placement and allocation such that the service processing time is guaranteed (i.e., no computation congestion at MECs) and the overall MEC number (e.g., operational cost) is kept low.
3) When the workload approaches the cloud capacity threshold, the system triggers the CVD mechanism based on the projected workload over a time window $\Delta t = t' - t$ where t' is the prediction time slot. Note that Δt will be selected according to the size of MEC network and the hosted mobile applications in MECs.
4) If it is detected that the ALB's limit will be reached within the coming time horizon, Δt, our NLCG algorithm is invoked to derive the desired new MEC node allocation based on the previous allocation solution. By appropriately deriving the NLCG start time, we minimize the added MECs in face of dynamic workloads.
5) A global reoptimization algorithm is performed periodically to adjust the MEC locations of the *entire* network, allocation of MEC nodes and the corresponding network paths such that given a certain user demands, the MEC operational cost is bounded.

B. Heuristic-based Incremental Allocation Mechanism

In the following, we detail every component of our heuristic-based incremental allocation mechanism.

1) Static Offline Resource Allocation: We first derive the minimum required number of MEC nodes, its network locations, amount of allocated resources and AP-to-MEC network paths with CPLEX to support the low latency requirement given the average / expected user demands. We highlight that the offline resource allocation takes place at the network planning stage which does not impose any optimization execution time constraints. However, when the input size to CPLEX is extremely large (e.g., more than 300 network nodes), a relaxed version [8] of the MEC operational cost minimization problem is solved to get a feasible solution within polynomial time.

2) Auto-Scaling and Load Balancing (ALB): Auto-scaling and load balancing are two current existing cloud computing elastic techniques to accommodate dynamic workload variations. We adopt a reactive auto-scaling solution that is triggered once a specific capacity threshold is reached. However, auto-scaling incurs additional VM reconfiguration delays which could affect service response time. This effect can be mitigated by setting a smaller auto-scaling threshold to invoke the auto-scaling mechanism in advance. Alternatively, proactive auto-scaling [34] can be applied to mitigate such auto-scaling overheads.

Algorithm 1 Capacity Violation Detection (CVD)

```
Input: $G(V,E), B$, predicted workload $A^{t'}, v', k_{v'}$
Output: Future time $t'$ and extra load $L_{v'}$ or no NLCG
 1: if current MEC nodes cannot accommodate $A^{t'}$ then
 2:     Derive new AP-to-MEC assignments and resource
        allocation with $VALB$
 3:     if $VALB$ cannot handle $A^{t'}$ then
 4:         Derive $L_{v'}$ by $A^{t'}$, the new assignments and
 5:         capacities of MEC nodes
 6:         Trigger NLCG algorithm return $t', L_{v'}$
 7:     else
 8:         Perform $ALB$
 9:     end if
10: end if
```

For load balancing, we adopt a proximity-aware solution [10] that considers both the residual capacity in MEC nodes and the topological proximity between MEC nodes and APs. Specifically, a flow from an AP to the overloaded MEC node will only be redirected when the newly chosen MEC node, v, is within the network latency cover, N_b, and the residual capacity is sufficient to accommodate the redirected load. By doing so, the network latency and MEC processing time are always bounded after load balancing.

3) Capacity Violation Detection (CVD) Mechanism: ALB have their limits, after which further increase in the request rate will incur increasing queuing delays at MEC nodes and lead to potential latency violations. The core idea of the CVD mechanism is to identify the time when such limitations will be reached so as to allow the system to pro-actively allocate new MEC node(s). Algorithm 1 presents the pseudocode of the CVD mechanism.

For CVD, we first assume that the workload can be reasonably predicted (e.g., perfect prediction). In practice, prediction algorithms predict workloads based on historical workload data. Algorithms such as generalized autoregressive conditional heteroscedasticity model [9] and various more [34] can be accommodated into CVD. We note that prediction techniques are not the main focus of this work. Given the current MEC node locations, resource utilization level and AP-to-MEC assignment, we predict over the time window Δt the aggregated workload $\sum_{p_{bv'} \in P_{bv'}} \sum_{b \in B} A_b^{t'} X_{p_{bv'}}^t$ at v' (i.e., v' is the MEC node that invokes the detection) and check if the predicted workload results in a capacity violation at v' (Line 1 in Algorithm 1). If the current state is predicted to be insufficient to accommodate the projected workload, we then estimate the future system state by virtually running ALB on the current system state with the projected workload.

The virtual ALB (VALB) aims to fully exploit computational resources provided by MEC nodes located in different network locations before triggering NLCG. It checks if load (e.g., offloading tasks from the same AP) from v' could be redirected to other MEC nodes while still conforming to the response time requirements of these flows. If virtual load balancing fails, virtual auto-scaling will be triggered to check

[7]For large scale problem, we solve the relaxed version of our problem, and derive the lower bound of optimal solution.

[8]We relax the routing decision variable (i.e., from integer to linear programming).

Algorithm 2 Network Latency Constraint Greedy (NLCG)

```
Input: $G(V,E), B$ represents APs, existing MEC nodes $V_s$,
    latency constraint $D$, overloaded MEC node $v'$, excess
    flow $L_{v'}$, predicted workload $A^{t'}$
Output: newly selected MEC node(s) and the corresponding
    routes
 1: New MEC node initialization $v_{bmax} \leftarrow \emptyset$
 2: Find the set of APs, $\mathcal{AP}_{v'}$, located in the distance cover
    of overloaded MEC node $v'$
 3: For each network node $v \in V \backslash V_s$, find the APs, $\mathcal{AP}_{v'v}$ ,
    that are located both in the cover of $v$ and $v'$
 4: for all $b \in \mathcal{AP}_{v'}$ do
 5:     for all $v \in N_b$ and $v$ not in $V_s$ do
 6:         if $v$ can accommodate excess flow $L_{v'}$ and
            $|\mathcal{AP}_{v'v}| \geq |\mathcal{AP}_{v'v_{bmax}}|$ then
 7:             $v_{bmax} \leftarrow v$
 8:         end if
 9:     end for
10: end for
11: if no MEC found $v_{bmax} == \emptyset$ then
12:     $v_{bmax} \leftarrow argmax(|\mathcal{AP}_{v'v}|)$
13:     trigger NLCG again with newly derived excess flow
        $L_{v'} = L_{v'} - k_{v_{bmax}}$
14: end if
15: Find network routes for the newly allocated MEC node(s)
    $X_{t'} \leftarrow MinMaxFaireness(v_{bmax}, \mathcal{AP}_{v'v_{bmax}})$
16: Update $Y^{t'}$ with $V_s \leftarrow V_s \cup v_{bmax}$
17: return MEC node locations $Y^{t'}$ and routings $X^{t'}$
```

if it can accommodate additional workloads by invoking auto-scaling. If this fails again, it means ALB will reach its limit within the projected time horizon and the overloaded MEC needs more computational resources to guarantee the service performance. Then, CVD records the excess load that cannot be served by v' as $L_{v'} = \sum_{p_{bv'} \in P_{bv'}} \sum_{b \in B} A_b^{t'} X_{p_{bv'}}^{t'} - k_{v'} Y_{v'}^t$ and triggers the online NLCG heuristic. It is worth mentioning that VALB is running as a real-time simulation where no actual ALB and any network configurations take place.

4) Network Latency Constraint Greedy Heuristic: The NLCG algorithm simultaneously determines the new placement of MEC node(s), the required resources and the corresponding routes. The idea of NLCG (Algorithm 2) is to search for a new MEC node located within the applications' network latency constraints that can accommodate the excess flow, $L_{v'}$, from the overloaded MEC node v' within the projected time. At the same time, the newly selected MEC node needs to satisfy as many flows (e.g., flows from APs served by other MEC nodes) as possible without violating network access delay to increase potential gain via load balancing to the new MEC node.

Specifically, NLCG first derives, for each network node other than existing MEC node $v \in V_s$, the number of APs covered by both the overloaded MEC node v' and v. To this end, NLCG finds the set of APs, denoted by $\mathcal{AP}_{v'} = \{b | d_{bv'} \leq D, b \in B\}$, within the latency coverage of the overloaded MEC, v' (Line 2 in Algorithm 2). Next, it adds all APs that are located within the distance cover of both v' and v into $\mathcal{AP}_{v'v} = \{b | d_{bv} \leq D, d_{bv'} \leq D, b \in B\}$ (Line 3). Then, for each AP within the distance cover $b \in \mathcal{AP}_{v'}$ of overloaded MEC v', NLCG searches the potential MEC node v from the candidate set $N_b = \{v | d_{bv} \leq D, v \in V\}$, and greedily chooses the node $v_{b_{max}}$ that has the highest $\mathcal{AP}_{v'v}$ and can support excess load $L_{v'}$ (Line 4-10). If no viable $v_{b_{max}}$ can be found, NLCG assigns the v that has the largest $\mathcal{AP}_{v'v}$ as v_{bmax} (Line 11-12). This means that there is no single node location that can host all the excess flows $L_{v'}$ from v'. In this case, NLCG will be triggered again with a reduced $L_{v'} = L_{v'} - k_{v_{b_{max}}}$ to find the next location to add (Line 13). NLCG then directs flows in $\mathcal{AP}_{v'}$ previously served by v' to $v_{b_{max}}$ and solve the routing problem using min-max fairness [35] (Line 15).

Algorithm 3 Set Cover Partition Approximation (SCPA)

```
Input: $G(V,E), B$ represents APs
Output: MEC nodes and the corresponding routes
 1: $V_s \leftarrow \emptyset$ where $V_s$ is the set of MEC nodes
 2: while $V_s$ is not a feasible solution do
 3:     Select $v \in V$ that maximizes the increase of newly
        covered APs in $V_s$
 4:     Store newly covered APs by $v$ into $\mathcal{AP}_v$
 5:     $V_s \leftarrow V_s \cup v$
 6: end while
 7: for all $v \in V_s$ do
 8:     $f_v \leftarrow G.fractionalMaxFlow(v, \mathcal{AP}_v)$
 9:     Construct subgraphs $G_v(V_v, E_v)$ with edges and nodes
        traversed by $f_v$
10:     $G_v.partition(\mathcal{AP}_v)$ [36] finds the unsplittable flows
        between APs in $\mathcal{AP}_v$ and $v$
11: end for
12: Superimpose paths found in each subgraph $G_v$
13: return MEC node locations and routings
```

Upon completion of NLCG, VM instantiation will start at NFV-enabled servers that have been selected to serve as MEC nodes. This instantiation process needs to accomplish before application workload $A^{t'}$ arrives so that application's response time will not be affected by VM instantiation. In other words, the overall time of VM instantiation and NLCG running time needs to be smaller than CVD's detection interval. In our framework, since CVD interval (e.g., on the order of minutes [9]) is not on the same order as VM instantiation time (e.g., on the order of tens of milliseconds [14], [15]), the abovementioned condition can be achieved if NLCG's running time is fast. We will evaluate NLCG's running time and heuristic's resulting application response time in Section V.

C. Set Cover Partition Approximation (SCPA) Global Reoptimization Algorithm

To complement our incremental allocation mechanism, we devise the SCPA reoptimization algorithm (see Algorithm 3) with guaranteed performance bounds where an approximation ratio is derived to indicate how far the obtained solution is from the optimal solution. The SCPA algorithm first finds the locations and resources of MEC nodes by solving a CSCP with each MEC node being assigned a subset of demand nodes (e.g., APs) without considering the capacity constraint of each link in the network. Clearly, this solution does not represent a feasible solution to our original problem, as the network link capacity constraint and AP-to-MEC paths are not incorporated. To obtain a feasible solution, SCPA then applies a graph partition technique to find the routes between each AP and MEC node that are assigned such that the link capacity constraint is satisfied. Specifically, we decompose the original MEC operational cost minimization problem into a CSCP and a set of single-source unsplittable flow problem (SSUFP)[9]. The solution to the CSCP gives MEC node allocation and the corresponding AP assignment, while the solution to each SSUFP derives the specific path between each MEC node and its assigned AP.

MEC node selection: We first show how the MEC node allocation for delay-sensitive applications without bandwidth constraints is transformed into a CSCP problem. To this end, we consider each network node $v \in V$ as a set in the CSCP problem, and its computational capacity represents the supply of the set. An AP b denotes an element in the CSCP problem, and it can be covered by v if the network latency constraint is satisfied with $d_{bv} \leq D$. The number of requests at b denotes the demand of its corresponding element in the CSCP problem. Without loss of generality, we assume that the total demand of all APs can be fulfilled by the total resources available in the network. Then, the MEC node allocation without bandwidth constraints but with latency constraints becomes finding a capacitated set cover for the CSCP problem. Let V_s be such a feasible solution to the CSCP problem, which can be found by utilizing the algorithm due to [37]. Each network node in $v \in V_s$ is selected to serve as a MEC node, and the APs, $\mathcal{AP}_v$, that are within its range in terms of network latency, will be covered by the MEC node allocated at v. The procedures of finding each MEC node $v \in V_s$ is described in Algorithm 3 (Line 2-5), whereby the basic idea is to find a network node at each iteration that covers the most of APs until all APs are assigned to one of the selected network node in V_s .

Network path selection: Next, we proceed to find the paths between each of the selected MEC node $v \in V_s$ and its covered APs, $\mathcal{AP}_v$, where the bandwidth resource constraint of each link in G is taken into account. We first get a fractional maximum flow f_v[10] for each MEC node $v \in V_s$ and its assigned APs in $\mathcal{AP}_v$ (Line 8). Based on f_v, we construct $|V_s|$ subgraphs $G_v(V_v, E_v)$ by including v, its assigned $\mathcal{AP}_v$, all other intermediate network nodes (V_v) that connect v and its $\mathcal{AP}_v$, and the links (E_v) traversed by f_v (Line 9). We then find SSUFP in the constructed subgraph for each selected network node $v \in V_s$, by using the algorithm *PARTITION* described [36] (Line 10). The basic idea of algorithm *PARTITION* is to further partition each subgraph into ε subgraphs by including APs that have demands in the same demand interval and the corresponding fractional paths from f_v. Then, in order to find a feasible unsplittable path for all APs in each new subgraph, *PARTITION* updates edge capacities in each newly obtained subgraph by rounding up APs' demand to the upper bound of its demand interval (i.e., this leads to the increase of edge capacity in subgraphs). Next, *PARTITION* iteratively applies augmenting path algorithm to find a feasible (e.g., conforms to augmented link capacities) unsplittable path for each AP. Finally, we superimpose unsplittable flows' solutions of each subgraph G_v to obtain the complete network paths (Line 12) for all APs. However, *PARTITION* violates at most $(4 + \varepsilon)$ relative edge capacity for any $\varepsilon > 0$, where $n^{\frac{1}{2}^{\xi-1}} \leqslant \varepsilon$ and ξ represents the number of partition intervals in algorithm *PARTITION*.

[9]In a single-source unsplittable flow instance (SSUFP), we are given a network $G = (V, E)$, a source vertex s, a set of k commodities with sinks $t_1, ..., t_k$ and the associated real-valued demands $\rho_1, ..., \rho_k$. The objective is to route the demand ρ_i of each commodity i along a single $s - t_i$ flow path so that the total flow routed across any edge e is bounded by the edge capacity BW_e.

[10]Note that maximum flow is a common problem where many different solutions can be applied (e.g., augmenting path algorithms [38]).

D. SCPA Algorithm Analysis

In this section, we derive the performance bounds of our SCPA global reoptimization algorithm detailed in Section IV-C. For this purpose, we will first re-state the following Theorems 1 and 2 given in [37] and [36] respectively.

Theorem 1. *[37]: Given a CSCP, there exists a greedy algorithm that finds a $\ln(N)$ approximation solution within running time of $O(|V|)$, where N gives the largest number of APs served by a MEC node in V_s.*

Theorem 2. *[36]: Given an UFP, algorithm PARTITION finds a $(4+\varepsilon)$ approximation for relative congestion for any $\varepsilon > 0$. The running time of the algorithm is $O(T_1(|V|, |E|)+|V||E|+|E|\varepsilon)$, where $T_1(|V|, |E|)$ is the time to solve a fractional maximum flow problem.*

Using the above, we can state the following theorem for our global reoptimization algorithm:

Theorem 3. *Given a NFV-enabled network environment, $G(V, E)$, where network node $v \in V$ has virtual computational resources k_v, network edge $e \in E$ has bandwidth BW_e, and APs $b \in B, B \subseteq V$ has user demands A_b, there is a fast approximation algorithm for the delay-guaranteed cost minimization problem that delivers a feasible solution with a cost no more than $\ln(N)$ times of the optimal cost in $O(|V| + |V_s|(T_1(|V|, |E|) + |V||E| + |E|\varepsilon))$ time, where N gives the largest number of APs served by a MEC node in V_s, $|V_s|$ gives the number of resulting MEC nodes and $T_1(|V|, |E|)$ is the time to solve a fractional maximum flow problem.*

Proof. We first show that the approximation ratio of our proposed SCPA algorithm is $\ln(N)$ times the optimal solution. Let C^* and C'^* be the optimal solutions to our problem with and without capacity constraints of network links.

The approximation solution to CSCP (Theorem 1) gives the lower bound of our original problem, i.e., $C'^* \leq C^*$. Specifically, in the aforementioned SCPA algorithm, the first step is to find the MEC node locations and the assignment of APs to the selected MEC nodes, which are given by solving the CSCP problem. Such node locations determine the resulting cost of both CSCP and our original problem defined in Section III-B. However, CSCP does not answer through which paths the APs and MEC nodes are connected and the network bandwidth capacity constraints are ignored, which is a special case of our original problem. Hence, the solution to CSCP is the lower bound to the original problem.

Denote by C' and C the solutions of the first (node selection) and second (path selection) steps of the proposed SCPA algorithm. Clearly, we have $C' = C$, because in the second step no network nodes are included or removed. We thus have

$$\begin{aligned} C &= C' \\ &\leq C'^{*} \cdot \ln(N) \text{ , since Theorem 1} \\ &\leq C^{*} \cdot \ln(N). \end{aligned}$$

This means that the approximation ratio of the proposed algorithm is $\ln(N)$.

We then show that feasible unsplittable paths between MEC nodes and the APs assigned to each MEC node can be found in polynomial time and the resulting edge congestion is no more than $(4+\varepsilon)|V_s|$ times edge capacity.

The idea of showing the bound of edge congestion is by considering the worst-case where the edge that has the maximum flow in a subgraph G_v overlaps with all edges from other subgraphs that also have the maximum flow. This situation could occur as we partition the original graph into $|V_s|$ subgraphs after we solve CSCP, and an edge from the original graph G can be shared by many subgraphs. According to Theorem 2, the relative edge congestion is at most $(4+\varepsilon)$ in a subgraph G_v. Hence, the worst-case relative edge congestion in the original graph is at most $(4+\varepsilon)|V_s|$ since an edge in G_v can overlap with at most $|V_s|$ edges when it is superimposed with other edges.

We have now shown that there is a set of unsplittable flows for each subgraph G_v obtained from the solution to CSCP and each edge has a congestion no more than $(4+\epsilon)|V_s|$. However, the edge congestion could violate the bandwidth constraint (2). This can be solved by setting the subgraph edge capacity by $\frac{BW_e}{(4+\varepsilon)|V_s|}$. Then, the edge capacity at all edges can be satisfied. Thus, the solution of the proposed SCPA algorithm satisfies all constraints and there is a feasible solution of paths for the lower bound (e.g., CSCP) of the original problem. This means that the approximation ratio of CSCP is the approximation ratio of the original problem.

Finally, we derive the running time of the proposed SCPA algorithm based on the running time from [36], where they showed solving a SSUFP requires a running time in $O(T_1(|V|,|E|)+|V||E|+|E|\varepsilon)$. More specifically, since our problem consists of solving a CSCP and $|V_s|$ SSUFP, we derive the running time by adding the running time of solving each subproblem. Therefore, the running time in our problem is $O(|V|+|V_s|(T_1(|V|,|E|)+|V||E|+\varepsilon|E|))$. □

V. Performance Evaluation

In this section, we evaluate the efficiency of our proposed framework in terms of service response time (i.e., round-trip time and processing delay at VMs) and cost efficiency under different MEC settings (e.g., network size, application latency requirement and server capacity). We first show in Section V-A that our dynamic resource allocation framework achieves the low latency requirement of the application while resulting in lower operational costs compared to existing approaches. We then focus on the performance analysis of the SCPA reoptimization algorithm in Section V-B. We compare SCPA's results against optimal and heuristic-based incremental allocation and show how close our SCPA algorithm can drive MEC systems back to the optimal state.

We clarify the schemes that will be compared against as follows:

1) *Overprovisioning* – We first solve the MEC placement and allocation at the peak workload with CPLEX in an offline manner and then, for each chosen location, we overprovision VMs with the maximum possible physical capacity to serve user requests, i.e., ALB and new MEC instantiation are never needed in this case.
2) *ALB* – We implement the initial solution from the static allocation problem at $t = 0$. The network performs ALB on the initial MEC locations (fixed locations) when needed.
3) *Heuristic* – Our proposed heuristic-based incremental allocation including NLCG algorithm, ALB and CVD (see the right-side of Fig. 3).
4) *Heuristic+Reoptimization* – Our proposed dynamic framework in full, combining heuristic-based incremental allocation and periodic SCPA global reoptimization that performs every 30 minutes.

A. Service Latency and Operational Costs

We use packet-level simulations to examine detailed MEC service latencies and operational costs. To this end, we create a realistic online NFV-enabled MEC simulation environment with OMNeT++ [39] complemented with an OpenFlow extension module provided by [40]. We implemented our dynamic resource allocation framework that operates as part of the centralized software-defined networking (SDN) controller. The controller connects to each network node through a dedicated network link (see Fig. 1), and dynamically carries out network configuration during MEC node instantiations.

We create a three-layer metropolitan wireless network shown in Fig. 1, consisting of APs, aggregation nodes and mobile core network nodes. In this network, the APs are deployed over an area of $46km^2$ where the deployment density is 0.65 APs per km^2. We further consider 1,800 mobile users moving following the mobility traces of a fleet of taxis operating in San Francisco [18]. Accordingly, we set up 30 APs, 5 aggregation nodes and 5 core network nodes (e.g, set according to part of Paris' core network model [41]) for the considered number of users and area where each network node is equipped with a cluster of commodity servers. In terms of server size, we follow [27] such that each network node has 21 servers and each server has 2.1GHz CPU of 18 cores. Moreover, we consider an AR application [31] where users upload street views captured by their mobile devices for annotations (e.g., building name, available parking places, etc.) computed by MEC. Such application requires a service response time of 480ms [1] and generates upload frames of size 0.5MB at 0.3FPS [31] which requires 230ms for a VM of 600MHz CPU to process [31]. For simplicity, we assume homogeneous frame size and upload rate for all users. In terms of network latency constraint, we set a maximum of 4 network

TABLE II: Performance comparison with realistic topology.

	Latency Requirement	Maximum Latency	Number of MEC nodes (start)$\longrightarrow$(end)	Cost saving(%)
Overprovision	Succeed	480ms	3$\longrightarrow$3	0%
ALB	Fail	132s	2$\longrightarrow$2	42.6%
Heuristic	Succeed	480ms	2$\longrightarrow$3	33.6%
Heuristic+ Reoptimization	Succeed	480ms	2$\longrightarrow$3	33.6%

hops[11] from AP to MEC node [42].

Given the aforementioned setup, we first derive the initial MEC node locations, resources needed and the corresponding network paths by CPLEX solver in an offline manner. Two MEC nodes are selected among all NFV-enabled nodes (the "Number of MEC nodes, (start)$\longrightarrow$(end)" column in Table II shows this number). Then, we execute our simulations for a duration of 1 hour from the abovementioned initial state, during which we gradually increase the AR application workload from 0.3FPS to the peak workload at 3.0FPS in steps of 0.1FPS every 400s. We set a threshold-based VM autoscaling mechanism for our packet-level simulation. Whenever VM load reaches a threshold of 80%, auto-scaling mechanism is triggered with a VM instantiation time of 100ms, This is set according to a realistic NFV commodity servers' instantiation time following [14]. In addition, we set the workload prediction time window, $\Delta t = 400s$ [9] for the NLCG algorithm, and consider a 100% prediction accuracy. This assumption has been largely adopted in the design of online resource allocation algorithms [3], [17], [25]. On the other hand, an inaccurate workload prediction would result in overprovisioning or underprovisioning of MEC resources in practice, which leads to poor cost efficiency and long processing delay respectively. Many existing work such as [43] have studied the impact of prediction inaccuracy and the compensation techniques (e.g., [43] proposed a method to minimize the impact of prediction inaccuracy, in which they minimized the underprovisioning-caused latency violations less than 2% of all requests). Therefore, our evaluation focuses on the proposed algorithms.

Now, we compare our solution against existing solutions in terms of service latency and operational costs. Table II shows our results with respect to satisfaction of the response time requirement, number of resulting MEC nodes (i.e., operational costs) at the start and end of the simulation and the cost savings over time in comparison to the *Overprovision* scheme. From the table, we can see that only the costly *Overprovision* and our solutions (*Heuristic* and *Heuristic+Reoptimization*), manage to satisfy the delay requirement of the considered AR application. In addition, *ALB* results in the lowest number of MEC nodes at the end of the simulations, but it comes with delay penalties due to computation congestion at the two initial MEC nodes. Our solutions have all increased the resulting number of MEC nodes by 1 in response to the increased workload. When we compare the costs over time against *Overprovision*, *ALB* achieves a saving of 42.6%. In contrast, our *Heuristic* and *Heuristic+Reoptimization* lead to a more modest saving (i.e., 33.6% in both cases), but achieves the latency requirement by increasing the overall computational resources through the new allocation of MEC nodes. Such saving is achieved by minimizing the number of required MEC node instantiations whereby the CVD mechanism derives the time instance when the resources of MECs will be fully utilized and cannot accommodate more workloads. However, due to the packet-level simulator's limitation, only a small topology is evaluated, whereby the performance improvement of *Heuristic+Reoptimization* cannot be revealed (e.g., identical results of cost saving in Table II).

[11] According to [42], when maximum number of network hops are no more than 4, MEC always outperforms DC-based cloud in terms of latency.

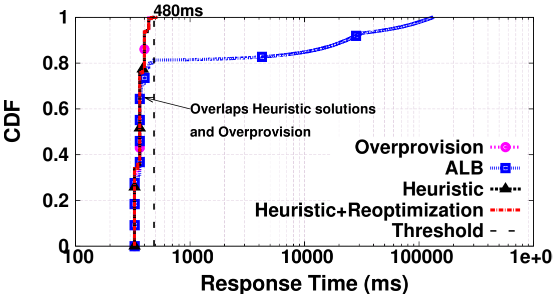


Fig. 4: (Color Online) Response time.

In addition, we observe from the cumulative distribution function (CDF) of response time in Fig. 4 that the resulting response time of our solutions overlap with that of *Overprovision*. This further shows the seamless transition to the new system state, and *Heuristic* approach is fast enough to get VMs ready before workload arrives. On the other hand, *ALB* fails to conform to the latency requirement with 20% (see Fig. 4) of the overall requests exceed the latency threshold (maximum latency at 132s) due to insufficient physical capacity in the fixed limited number of MECs.

Our detailed packet-level simulator allows us to track and examine each and every individual request and response packet in the system. The tradeoff to this is the scalability of the simulator which constrained us to smaller scale simulations. To more comprehensively evaluate our solution, we further evaluate our framework, specifically on the benefits brought by global re-optimization algorithm, SCPA, with larger network topologies in the next section. Also, we thoroughly investigate the impact of different network sizes, network hop constraints and MEC service-hosting servers' sizes to our solution.

B. System Cost Optimality

We proceed to evaluate the improvements provided by SCPA via flow-level simulations with large network topologies, and investigate by how much SCPA can drive the NFV-enabled MEC back to the optimal state. To this end, we compare the resulting MEC operational cost of *Heuristic+Reoptimization* against *Heuristic* and lower bound of optimal solution[12] denoted by OPT_{LB} under different network sizes, latency requirements and physical capacities of NFV-enabled servers. Furthermore, in order to more intuitively present SCPA reoptimization's optimality difference to OPT_{LB} and take into account MECs' resource utilization level, we introduce two metrics: *cost efficiency* and *cost efficiency gap*. The cost efficiency, C_{eff}, quantifies the number of mobile users per MEC node who achieve the required service response time.

$$\text{Cost efficiency, } C_{eff} = \frac{Nb_{users}}{|V_s|} \tag{7}$$

where Nb_{users} is the total number of users who receive their services within the services' latency requirements.

Cost efficiency gap shows how close the resulting cost efficiency of our solutions (i.e., *Heuristic+Reoptimization* and *Heuristic*) is to the OPT_{LB}, that is, the smaller this gap is, the more cost-efficient the solution is. More specifically, this metric is derived as the normalized difference between cost efficiency of our solutions and that of OPT_{LB}.

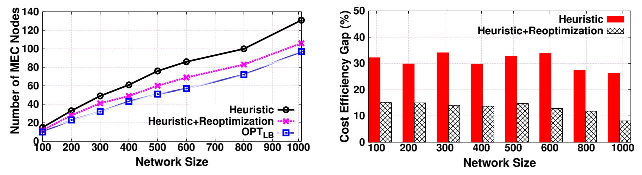
$$\text{Cost efficiency gap, } Gap_{eff} = \left| \frac{C_{eff}^{OPT_{LB}} - C_{eff}}{C_{eff}^{OPT_{LB}}} \right| \tag{8}$$

where $C_{eff}^{OPT_{LB}}$ denotes the cost efficiency of OPT_{LB}.

1) Impact of Network Size: We adopt GT-ITM [44] to generate synthetic network topologies where the probability of having an edge between two nodes is 0.2 with edge capacities uniformly distributed between 300Mbps and 10Gbps. Other setup / parameters related to the application, workload and server capacity remain the same as previously described (see Section V-A). We plot in Fig. 5(a) the average number of MEC nodes in function of different network sizes ranging from 100 nodes to 1000 nodes for *Heuristic*, *Heuristic+Reoptimization* and OPT_{LB}. It must be stressed that the average number of MEC nodes at each network size (e.g., 100 to 1000 nodes) is the average number of MEC nodes of 4 simulations with different service latency requirements (e.g., maximum number of hops from 1 to 4 hops). By doing so, the impact of a specific latency requirement to the MEC node number is reduced, and hence Fig. 5(a) can reflect the impact of network sizes to MEC node number in a more accurate way.

From Fig. 5(a), we see that the *Heuristic+Reoptimization* solution achieves lower operational costs (i.e., lower number of resulting MEC nodes) for all network sizes compared to *Heuristic*. The resulting MEC operational cost of our *Heuristic+Reoptimization* also closely follows that of OPT_{LB}. The relative poorer performance achieved by *Heuristic* is due to its local search nature where the search of a new MEC node is triggered by overloaded existing MEC nodes and carried

[12]Such OPT_{LB} is solved by relaxing both the edge capacity constraint and the routing decision variable X_p (i.e., from integer to linear programming). Note that this is a conservative estimation of the optimal solution, which is smaller than the optimal value. In addition, due to the complexity in deriving the OPT_{LB} solutions for large size networks (e.g., larger than 300 nodes), we stop the CPLEX solver when the optimality gap reaches 5% to avoid long execution time.



(a) Average costs for each network size over different latency requirements.
(b) Cost efficiency gap to OPT_{LB}.

Fig. 5: (Color Online) Impact of network sizes to costs.

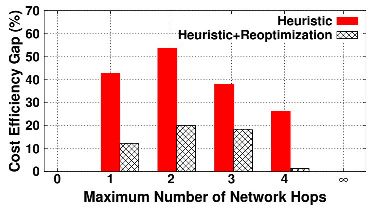


Fig. 6: (Color Online) Cost efficiency gap to OPT_{LB}.

out in the vicinity of these affected nodes. As a result, the optimal MEC location that may benefit the maximum number of users could potentially be omitted during *Heuristic*'s search process, leading to a relatively lower cost efficiency. In contrast, *Heuristic+Reoptimization* utilizes resources more efficiently by searching the optimal MEC locations over the entire network.

We show in Fig. 5(b) the cost efficiency gap to OPT_{LB} for *Heuristic+Reoptimization* and *Heuristic*. We see that *Heuristic*'s cost efficiency gap to OPT_{LB} is always above 25%, whereas *Heuristic+Reoptimization* can improve nearly 20% of *Heuristic*'s cost efficiency due to the global search. In addition, *Heuristic+Reoptimization* consistently achieves an efficiency gap below 15% for any network sizes (see Fig. 5(b)). In particular, we observe that *Heuristic+Reoptimization*'s efficiency gap does not increase with network size, which justifies the theoretical performance bound $\ln(N)$, whereby N represents the largest number of APs served by a MEC node, which is independent to the size of network.

2) Impact of Latency Requirements: The latency requirement can be interpreted as the maximum tolerable number of network hops between APs and MEC nodes. It directly affects the number of APs that a NFV network node can cover (i.e., serving the APs without violating latency requirements). This, in turn, affects the required MEC nodes to cover all APs in the proposed algorithms. To show the impact of this factor, we vary the maximum tolerable number of network hops from 1 to 4, which reflects latency requirements of different nature such as extremely strict network latencies (e.g., 10ms network delay) to loose latencies (e.g., 150ms network latency). We show, with Fig. 6, both *Heuristic* and *Heuristic+Reoptimization*'s cost efficiency gap ratio for each of considered latency requirement. Note that the cost efficiency gap at each latency requirement in Fig. 6 is the average of that of all network sizes (e.g., 100 to 1000).

We see from Fig. 6 that *Heurisitc+Reoptimization* still outperforms *Heuristic* for each latency requirement, and it always achieves an efficiency gap below 20%. In particular, when the maximum network hop is set to zero, both *Heuristic* and *Heurisitc+Reoptimization* achieve an optimal operational cost where the efficiency gap equals to zero. This is due to the fact that the extreme low latency constraint (e.g., 0 hop) restricts all APs to be served as MEC nodes, which makes the resulting number of MEC nodes identical for any MEC allocation algorithms that conforms to the network latency constraint. Similarly, when we look at the other extreme case where the latency constraint is extremely loose (see ∞ in Fig. 6) and the physical NFV servers have infinite capacity, only one MEC node is required in *Heuristic*, *Heurisitc+Reoptimization* and OPT_{LB} (e.g., this leads to 0% cost efficiency gap). From the above two cases, we observe that the selection of MEC resource allocation algorithm does not play a critical role in the resulting MEC operational cost when latency is either extremely loose or strict. However, when the latency requirement is between the two extremes cases, it significantly affects the cost efficiency. For instance, when the latency requirement is set to 1, 2, 3 and 4 network hops, we observe from Fig. 6 that the cost efficiency gap of both *Heuristic* and *Heurisitc+Reoptimization* first increases and then decreases as the maximum tolerable network hops increase. The increase of cost efficiency gap at network hop 1 and 2 compared to 0 hop is due to the enlarged search space for MEC nodes in *Heuristic* and *Heurisitc+Reoptimization*. Such search space enlargement increases the chance of selecting less optimal MEC nodes where MECs' resource utilization is poorer compared to MECs derived by OPT_{LB}. On the other hand, the decrease of efficiency gap at 3 and 4 network hops is the consequence of improved MEC utilization compared to cases with 1 and 2 network hops. Specifically, due to the relaxed latency constraint, a MEC node can serve a larger number of users without violating the network latency requirement, and hence achieve a better resource utilization compared to strict latency requirements. In particular, the relaxed latency constraint at 4 network hops results in a situation where the number of served users in each MEC reaches servers' physical capacity limits, that is, the resource utilization at each derived MEC is almost 100%. Knowing that an optimal MEC resource allocation achieves the least number of MEC also by fully utilizing MECs' resources. Therefore, the fully utilized MEC nodes at 4 network hops achieve a very close cost to OPT_{LB}. Similarly, when the latency becomes even less strict (e.g., ∞), the allocated resources at MECs will reach the servers' physical capacity limits and result in the same operational cost as OPT_{LB} (see ∞ in Fig. 6). Clearly, there is an inter-correlation between applications' latency requirement and the server capacity, which we elaborate in the next subsection.

3) Impact of Physical Capacities: Next, we evaluate the impact of servers' physical capacities to *Heurisitc+Reoptimization*'s MEC costs. To this end, we consider three NFV-enabled servers sizes, namely, *FULL* (i.e., the considered server size (see Section V-A)), *HALF* (i.e., half of *FULL* size), *DOUBLE* (i.e., two times the *FULL* size) [17], [27]. Furthermore, servers of different sizes result in different energy consumption, which can be estimated

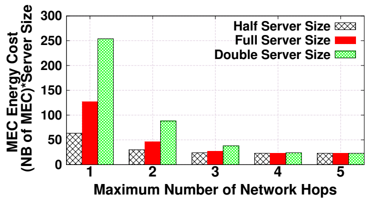


Fig. 7: (Color Online) *Heuristic+Reoptimization*'s average cost for each latency constraint over different network sizes.

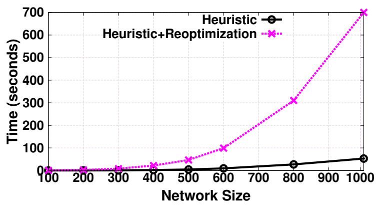


Fig. 8: (Color Online) Algorithm running time comparison.

based on the server size and resource utilization [3]. We take a simplistic assumption in our evaluation whereby the energy consumption is proportional to the server size. That is, we consider *HALF* size servers consume half of *FULL* size servers' energy and correspondingly, *DOUBLE* size servers consume double the amount of energy of *FULL* size servers. We plot in Fig. 7 the average energy cost incurred by *Heurisitc+Reoptimization* under different network latency requirements for each of the abovementioned server sizes. We observe that simulations with *DOUBLE* server size result in higher costs than *HALF* and *FULL* server when network latency requirement is extremely low (e.g., 1 network hop). This is due to the inter-correlation between the two impact factors: latency requirement and server size. More specifically, when network latency requirement is extremely low, the latency requirement impact factor dominates the MEC node searching process leading to almost the same number and placement of resulting MEC nodes for *HALF*, *FULL* and *DOUBLE* size servers. However, the per MEC energy consumption of *DOUBLE* size server is significantly more than that of *HALF* and *FULL* size which results in the overall higher costs (see Fig. 7). In contrast, when the latency requirement becomes less strict (e.g., network hops 3 and 4), *DOUBLE* size servers' energy cost decreases drastically as a consequence of decreased number of required servers and better resource utilization compared to that of strict latency requirements (i.e., each server supports a larger number of users within its network latency constraint). At the same time, we see from Fig. 7 that the resulting cost of the 3 server sizes converges to the same level after 3 network hops. For each server size, more users are served per MEC after the relaxation of latency requirements, and hence all MECs are almost fully utilized. As a consequence, the overall number of MEC nodes with full-size servers is half of that of half-size case and double of double-size case. Given the simplified energy cost assumption, our dynamic resource allocation framework results in the same level of energy consumption for each server size when latency requirement is loose.

Given the above observations, we see that the performance of dynamic resource allocation framework is independent of the network size. In particular, *Heurisitc+Reoptimization* can always improve *Heuristic*'s resulting operational cost except when the latency requirement is extremely low (e.g., 0 hops) or extremely high (e.g., ∞ hops). Also, the observations from the impact factor analysis of latency and server capacity provide insights on the server size selection in the NFV-enable MEC cost minimization problem. We conclude that for extreme low latency applications (e.g., under 10ms), deployment of smaller servers are more desirable in order to achieve lower MEC cost through dynamic resource allocation. However, when the latency requirement is loose, the server size does not have strong influence on the MEC operational costs.

4) Algorithm Running Time: Last, we show in Fig. 8 the average running time of NLCG heuristic and SCPA reoptimization for each network size whereby the average running time is derived over different latency requirements. As Fig. 8 shows, the SCPA takes more time to execute than NLCG heuristic, but achieves a cost efficiency within 20% of OPT_{LB}'s cost efficiency (see Fig. 6). In addition, we observe that when network size is larger than 500 nodes, SCPA running time increases drastically due to the increased complexity in finding unsplittable flows. However, it must be stressed that conventional metropolitan-level wireless networks have network size smaller than 700 nodes [17], [24], and even the maximum execution time (e.g., 200s) for 700 nodes does not affect the desired latency requirements in the considered online NFV-enabled MEC (e.g., SCPA is performed less frequently than incremental MEC allocation in dynamic resource allocation framework). On the other hand, the NLCG's running time is below 50s in the worst case (e.g., network size 1000), which does not affect the latency requirements (i.e., the sum of VM instantiation time and NLCG's running time is always smaller than CVD detection interval).

VI. Summary and Conclusions

We address the challenge of designing dynamic mobile edge-cloud (MEC) resource allocation for delay sensitive mobile applications in a Network Function Virtualization (NFV)-enhanced MEC environment. Specifically, we consider new flexibility afforded by NFV in dynamic MEC instantiations rather than the existing fixed-location MEC allocation practices. For this, we formulate an optimization problem for allocating MEC services at any resource-constrained NFV-enabled nodes so that resources are optimally allocated to satisfy the applications' latency requirements, while incurring minimum operational costs to ISPs. Since the problem is NP-hard, we designed a novel dynamic resource allocation framework consisting of an online heuristic-based incremental allocation solution (i.e., using combination of NLGC algorithm with CVD and ALB mechanisms) and a reoptimization solution (i.e., SCPA) with a guaranteed approximation ratio. In particular, our online heuristic-based incremental allocation mechanism aims to efficiently allocate resources to tackle local MEC computation congestion due to (sudden) increase of workload in a timely manner, such that the low latency requirements are always achieved. The reoptimization solution readjusts the sub-optimal MEC resource allocation resulted by the incremental solution, and drives MEC systems back towards the optimal state. We demonstrate the effectiveness of our dynamic resource allocation framework in NFV-enabled MEC through both packet-level and flow-level simulations. Our results show that only our proposal always ensures that MEC services respond to user requests on time, while achieving up to 33% operational cost reduction in comparison to the current fixed-location MEC practices. Meanwhile, our proposal achieves a near-optimal MEC operational cost whereby the cost efficiency is no more than 20% of that incurred by optimal MEC resource allocation. In addition, our impact factor analysis indicates that MEC applications with extreme low latency requirements (e.g., 10ms) are more in favour of small size servers for cost efficiency purposes.

For the future work, we aim to further investigate dynamic resource allocation for stateful low latency applications, whereby changing the user-to-MEC assignment incurs migration costs. For such scenarios, the average migration costs need to be minimized over a time period. Moreover, we aim to expand the current resource allocation and optimization framework to support multiple services of different performance requirements while still being fast and efficient.

Acknowledgment

This work was partially funded by the CHIST-ERA CONCERT/EPSRC, project number I1402 and the EU H2020 UMOBILE, project number 645124.

References

[1] P. Jain, J. Manweiler, and R. Roy Choudhury, "Overlay: Practical mobile augmented reality," in *ACM MobiSys*, 2015, pp. 331–344.

[2] K. Ha *et al.*, "The impact of mobile multimedia applications on data center consolidation," in *IEEE Conf. on IC2E*, 2013, pp. 166–176.

[3] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[4] M. Satyanarayanan *et al.*, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[5] D. Zeng *et al.*, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Trans. on Computers*, vol. 65, pp. 3702–3712, 2016.

[6] J. Soares *et al.*, "Toward a telco cloud environment for service functions," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 98–106, 2015.

[7] T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, 2013.

[8] S. Wang *et al.*, "Dynamic service migration in mobile edge-clouds," in *IFIP Networking*, 2015.

[9] D. Niu *et al.*, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *IEEE INFOCOM*, 2012, pp. 460–468.

[10] Y. Zhu and Y. Hu, "Efficient, proximity-aware load balancing for dht-based p2p systems," *IEEE Trans. on Parallel and Distributed Systems*, vol. 16, no. 4, pp. 349–361, 2005.

[11] R. Mijumbi *et al.*, "Management and orchestration challenges in network functions virtualization," *IEEE Commun. Mag.*, vol. 54, no. 1, pp. 98–105, 2016.

[12] N. Bouten *et al.*, "Towards nfv-based multimedia delivery," in *Symp. on IEEE IM*, 2015, pp. 738–741.

[13] H. Yin *et al.*, "Edge provisioning with flexible server placement," *IEEE Trans. on Parallel and Distributed Systems, DOI:10.1109/TPDS.2016.2604803.*

[14] A. Madhavapeddy *et al.*, "Jitsu: Just-in-time summoning of unikernels," in *USENIX Symp. on NSDI*, 2015, pp. 559–573.

[15] J. Martins *et al.*, "Clickos and the art of network function virtualization," in *USENIX Conf. on Networked Systems Design and Implementation*, 2014, pp. 459–473.

[16] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *IEEE INFOCOM*, 2016, pp. 1–9.

[17] A. Ceselli, M. Premoli, and S. Secci, "Cloudlet network design optimization," in *IFIP Networking*, 2015.

[18] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "A parsimonious model of mobile partitioned networks with clustering," in *IEEE Communication Systems and Networks and Workshops*, 2009, pp. 1–10.

[19] B. Yang *et al.*, "Seamless support of low latency mobile applications with nfv-enabled mobile edge-cloud," in *IEEE CloudNet*, 2016, pp. 136–141.

[20] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *IEEE INFOCOM*, 2011, pp. 71–75.

[21] B. Yang *et al.*, "Cost-efficient low latency communication infrastructure for synchrophasor applications in smart grids," *IEEE Syst. J., DOI:10.1109/JSYST.2016.2556420.*

[22] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[23] T. Taleb and A. Ksentini, "An analytical model for follow me cloud," in *IEEE GLOBECOM*, 2013, pp. 1291–1296.

[24] Z. Xu *et al.*, "Capacitated cloudlet placements in wireless metropolitan area networks," in *IEEE LCN*, 2015, pp. 570–578.

[25] Z. Xu *et al.*, "Efficient algorithms for capacitated cloudlet placements," *IEEE Trans. on Parallel and Distributed Systems*, vol. 27, pp. 2866–2880, 2016.

[26] M. Jia *et al.*, "Cloudlet load balancing in wireless metropolitan area networks," in *IEEE INFOCOM*, 2016, pp. 1–9.

[27] Q. Xia, W. Liang, and W. Xu, "Throughput maximization for online request admissions in mobile cloudlets," in *IEEE LCN*, 2013, pp. 589–596.

[28] H. Huang and S. Guo, "Service provisioning update scheme for mobile application users in a cloudlet network," in *IEEE ICC*, 2017, pp. 1–6.

[29] R. Landa *et al.*, "Self-tuning service provisioning for decentralised cloud applications," *IEEE Trans. on Network and Service Management*, vol. 13, pp. 197–211, 2016.

[30] Y. Cao and N. Wang, "Toward efficient electric-vehicle charging using vanet-based information dissemination," *IEEE Trans. on Vehicular Technology*, vol. 66, no. 4, pp. 2886–2901, 2017.

[31] R. LiKamWa and L. Zhong, "Starfish: Efficient concurrency support for computer vision applications," in *ACM MobiSys*, 2015, pp. 213–226.

[32] J. R. Current and J. E. Storbeck, "Capacitated covering models," *Environment and planning B: planning and Design*, vol. 15, no. 2, pp. 153–163, 1988.

[33] I. I. CPLEX, "V12. 1: User manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[34] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *IEEE CLOUD*, 2011, pp. 500–507.

[35] B. Radunović and J.-Y. L. Boudec, "A unified framework for max-min and min-max fairness with applications," *IEEE/ACM Trans. on Networking*, vol. 15, no. 5, pp. 1073–1083, 2007.

[36] S. G. Kolliopoulos and C. Stein, "Approximation algorithms for single-source unsplittable flow," *SIAM Journal on Computing*, vol. 31, no. 3, pp. 919–946, 2001.

[37] J. Chuzhoy and J. Naor, "Covering problems with hard capacities," *SIAM Journal on Computing*, vol. 36, no. 2, pp. 498–515, 2006.

[38] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network flows: theory, algorithms, and applications," 1993.

[39] A. Varga, *OMNeT++ Simulator Home Page*, http://www.omnetpp.org.

[40] D. Klein and M. Jarschel, "An openflow extension for the omnet++ inet framework," in *ICST Conf. on Simulation Tools and Techniques*, 2013.

[41] S. Knight *et al.*, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

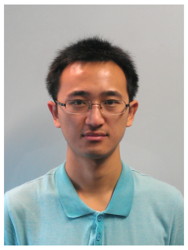
[42] D. Fesehaye *et al.*, "Impact of cloudlets on interactive mobile cloud applications," in *IEEE Conf. on Enterprise Distributed Object Computing*, 2012, pp. 123–132.

[43] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in *IEEE CNSM*, 2010, pp. 9–16.

[44] K. L. Calvert, M. B. Doar, and E. W. Zegura, "Modeling internet topology," *IEEE Commun. Mag.*, vol. 35, no. 6, pp. 160–163, 1997.

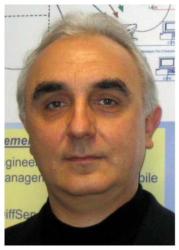
Binxu Yang received the B.S. degree in telecommunications from Xidian University, China; the French Engineering degree from Telecom Bretagne, France; and the M.Res. degree (Distinction) from University College London (UCL), U.K, in 2010, 2013, and 2014, respectively. He is currently a final-year Ph.D. from the Department of Electronic and Electrical Engineering, UCL. His current research interests include mobile edge computing and resource allocation in network function virtualization.

Wei Koong Chai received the B.Eng. degree in electrical engineering from the Universiti Teknologi Malaysia, Johor Bahru, Malaysia, in 2000, and both the M.Sc. (Distinction) and the Ph.D. degrees from the University of Surrey, Surrey, U.K., in 2002 and 2008, respectively. He is currently a Senior Lecturer in the Department of Computing and Informatics, Bournemouth University, Dorset, U.K. as well as a Visiting Academic - Honorary Senior Research Associate - at University College London (UCL). Prior to this, he was with the Department of Electronic and Electrical Engineering, UCL, as Senior Research Fellow. His current research interests include information-centric networking, network science and resource management (e.g., for mobile cloud networks, satellite networks, wireless mesh networks).



Zichuan Xu received his Ph.D. degree from the Australian National University in 2016, ME degree and B.Sc. degree from Dalian University of Technology in China in 2011 and 2008, all in Computer Science. He is currently an Associate Professor at the School of Software, Dalian University of Technology in China. He was a Research Associate at Department of Electronic and Electrical Engineering, University College London, UK. His research interests include cloud computing, software-defined networking, network function virtualization, wireless sensor networks, routing protocol design for wireless networks, algorithmic game theory, and optimization problems.

Konstantinos V. Katsaros received his B.Sc. in informatics (2003), and his M.Sc. (Honours, 2005) and Ph.D. (2010) degrees in computer science from A.U.E.B, Greece. He has worked in the areas of cloud networking, smart grid communications, mobile grid computing, and multicast/broadcast service provision over cellular networks. His research interests focus on NFV/SDN technologies for 5G networks and information-centric networking. Currently, he is a senior research engineer at Intracom Telecom, Greece. Prior to this, he was a research associate at the Department of Electronic and Electrical Engineering, University College London, United Kingdom.



George Pavlou is Professor of Communication Networks in the Department of Electronic and Electrical Engineering, University College London, UK where he co-ordinates research activities in networking and network management. He received a Diploma in Engineering from the National Technical University of Athens, Greece and M.S. and Ph.D. degrees in Computer Science from University College London, UK. His research interests include aspects such as resource management, traffic engineering, quality of service, autonomic networking, network programmability and content-based networking. He has been instrumental in a number of European and UK research projects that produced significant results with real-world uptake and has contributed to standardisation activities in ISO, ITU-T and IETF. He has been the technical program chair of several key conferences in the area and in 2011 he received the Daniel Stokesbury award for distinguished technical contribution to the growth of the network management field.