Lin, Y-D., Lai, Y-C., Huang, J-X y Chien, H-T. (2018). Three-Tier Capacity and Traffic Allocation for Core, Edges, and Devices for Mobile Edge Computing. *IEEE Transactions on Network and Service Management*, 15 (3), pp. 923-933.

DOI: 10.1109/TNSM.2018.2852643

Three-Tier Capacity and Traffic Allocation for Core, Edges, and Devices for Mobile Edge Computing

Ying-Dar Lin, *Fellow, IEEE*, Yuan-Cheng Lai, Jian-Xun Huang, and Hsu-Tung Chien

***Abstract*—In order to satisfy the 5G requirements of ultra-low latency, mobile edge computing (MEC)-based architecture, composed of three-tier nodes, core, edges, and devices, is proposed. In MEC-based architecture, previous studies focused on the control-plane issue, i.e., how to allocate traffic to be processed at different nodes to meet this ultra-low latency requirement. Also important is how to allocate the capacity to different nodes in the management plane so as to establish a minimal-capacity network. The objectives of this paper is to solve two problems: 1) to allocate the capacity of all nodes in MEC-based architecture so as to provide a minimal-capacity network and 2) to allocate the traffic to satisfy the latency percentage constraint, i.e., at least a percentage of traffic satisfying the latency constraint. In order to achieve these objectives, a two-phase iterative optimization (TPIO) method is proposed to try to optimize capacity and traffic allocation in MEC-based architecture. TPIO iteratively uses two phases to adjust capacity and traffic allocation respectively because they are tightly coupled. In the first phase, using queuing theory calculates the optimal traffic allocation under fixed allocated capacity, while in the second phase, allocated capacity is further reduced under fixed traffic allocation to satisfy the latency percentage constraint. Simulation results show that MEC-based architecture can save about 20.7% of capacity of two-tier architecture. Further, an extra 12.2% capacity must be forfeited when the percentage of satisfying latency is 90%, compared to 50%.**

***Index Terms*—Iterative optimization, mobile edge computing (MEC), three-tier architecture, capacity allocation.**

I. Introduction

As THE volume of data grows rapidly, 5G wireless communication has been proposed [1]. The aim of 5G is to provide very high data rates and ultra-low latency in the order of ms [2], [3]. The performance requirements of latency in 5G networks are a 1 ms limit for data-plane latency, a 10 ms limit for control-plane latency [4], and a 1000 times faster data rates than 4G networks [5]. Thus, one of the most important challenges of 5G is to address a contradiction between the increasing complexity of mobile applications and its limited latency [6].

Traditional wireless architecture is two-tiered and consists of a core and devices. These provide specific functions for the services required by a user, such as a user playing a multimedia game using a mobile device. In this case, the device is responsible for the encoding/decoding functions, while the core is responsible for the calculating the game's operations. However, there is usually a long distance between core and devices, resulting in a long propagation delay which cannot meet the latency limitation of 5G networks.

Thus, MEC-based architecture, based on the concept of fog computing, has been proposed [7]. In this architecture, some equipment, termed "edge", located as an intermediate tier, is deployed between the core and devices: it is a three-tiered network architecture with core, edges, and devices. The edges carry out functions of both core and devices. Thus, in MEC-based architecture, for functions required by traffic, the computation of functions in devices and core can be offloaded to that in edges. Computation offloading from core to edges can avoid the propagation delay between core and edge. Also, since edges are closer to users' devices than the core, when services are served in the nearer edge, the propagation delay will be significantly reduced [4]. Thus MEC-based architecture is thus very suitable for 5G networks because the latency limitation is more likely to be satisfied. On the other hand, computation offloading from devices can reduce devices' loading, so that the cost of devices and their power consumption can be reduced [5].

Most previous research on offloading was designed for two-tier architecture which consists only of a core and devices [8]–[16], where offloading was essentially one way, from devices to core. The research [17] designed for MEC-based architecture mainly handled two-way offloading. However, offloading, covered in [8]–[17], addresses the issue of traffic allocation in the control plane. To the best of our knowledge, no current research considers capacity allocation in the management plane. In fact, capacity allocation and traffic allocation are mutually affected: if for example more capacity is allocated to edges, more traffic will be served in them. Thus, capacity allocation and traffic allocation should be considered simultaneously. The objective of this paper is to address two problems: (1) to allocate the capacity of all nodes in MEC-based architecture to create a minimal-capacity network; and (2) to allocate the traffic to satisfy

This work was supported in part by H2020 collabora-tive Europe/Taiwan research project 5G-CORAL (grant number 761586), and Ministry of Science and Technology, Taiwan for financially supporting this research under Contract No. MOST 106-2218-E-009-018. The associate edi-tor coordinating the review of this paper and approving it for publication was G. Bianchi. *(Corresponding author: Yuan-Cheng Lai.)*

Y.-D. Lin and H.-T. Chien are with the Department of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: ydlin@cs.nctu.edu.tw; hsutung@cs.nctu.edu.tw).

Y.-C. Lai is with the Department of Information Management, National Taiwan University of Science and Technology, Taipei 106, Taiwan (e-mail: laiyc@cs.ntust.edu.tw).

J.-X. Huang was with the Department of Information Management, National Taiwan University of Science and Technology, Taipei 106, Taiwan (e-mail: hibari180505@gmail.com).

the latency percentage constraint, i.e., at least a percentage of traffic satisfying latency constraint. Note that this work uses a latency percentage constraint, rather than a latency constraint, because the former is more suitable to users' Quality of Experiences (QoE).

In order to achieve these objectives, we proposed a "two-phase iterative optimization" (TPIO) algorithm to try to optimize capacity and traffic allocation: one phase for adjusting capacity allocation and the other for adjusting traffic allocation. The amount of inputted traffic is proportional to computation workload because all packets are assumed to be homogenous, i.e., they require the same computation workload. The first phase fixes the allocated capacity and adjusts the traffic allocation to satisfy the latency percentage constraint. The second phase fixes the allocated traffic and adjusts the capacity allocation to minimize the total capacity of all nodes. The two phases take turns until the total capacity has been minimized and the latency percentage constraint has been satisfied.

Note that the problem we address is an optimization problem, that is, constructing a minimal-capacity network under satisfying the latency percentage constraint. However, to avoid too high complexity, our proposed algorithm, TPIO, is actually a heuristic solution to achieve near-optimal results.

To the best of our knowledge, our research is the first to overcome capacity allocation and traffic allocation in MEC-based architecture. Our contributions are as follows: (1) we derive end-to-end latency distribution and the percentage of the traffic satisfying latency constraint; (2) we design a new system model where the offloading can only happen from devices to edges and from core to edges, but not from devices to core; (3) we propose the TPIO method, which allocates the capacity of all nodes in MEC-based architecture, so as to provide a minimal-capacity network and allocate the traffic to satisfy the latency percentage constraint; and (4) we conduct extensive simulations to demonstrate the benefits of MEC and investigate the effects of different parameters in MEC-based architecture.

The rest of this work is organized as follows: Section II covers some background and reviews related work; Section III defines the problem statement and Section IV describes the details of the TPIO algorithm; Section V evaluates TPIO by some simulations and Section VI concludes the work.

II. Background

A. MEC-Based Architecture

The ETSI (European Telecommunication Standard Institute) has proposed MEC-based architecture within a radio access network which is in close proximity to mobile users [7]. MEC-based architecture introduces several advantages, including:

- QoE improvement: MEC can improve the QoE of users by pushing data-intensive tasks towards an edge and locally processing data in proximity to the users instead of at a remote core.
- Rapidly deployment: because MEC-based architecture is an extension of 4G networks, Mobile Network Operators (MNOs) can rapidly deploy new services to consumer and enterprise business segments which can help them differentiate their service portfolios [7].
- Traffic bottleneck reduction: MNOs can reduce traffic bottlenecks at the core and backhaul networks, while assisting in the offloading of heavy computational tasks from power-constrained user devices to an edge.

B. Related Work

Most previous work investigated computation offloading in two-tier architecture. Some of this research discussed the case where only one service is provided [8]–[12]. Tang *et al.* [8] designed a socially-aware mobile network and modeled the offloading problem as a socially-aware computation offloading game. Chen [10] considered the multi-user computation offloading problem in a single-channel wireless setting, so that each user has a binary decision variable (i.e., to offload or not). The same authors in another paper [9] considered the same multi-user computation offloading problem, but extended it from single to multiple channels (i.e., to offload which wireless channel or not).

Other researchers discussed the case where multiple services exist [14]–[16]. Since the complexity of cases for multiple services is very high, these studies made different additional assumptions so as to reduce time complexity. For example, [14] assumed that only one device appears; [16] assumed that different data-center operators offer different specific services, and each user is allowed to subscribe a specific service from a certain data-center operator.

For computation offloading problems, there may be different objectives: latency minimization, energy saving, overhead minimization. For latency minimization [11], [12], [17] attempted to minimize the latency under the constraint of energy consumption, while for energy saving [8], [13]–[15] attempted to reduce energy consumption under the constraint of latency or execution time.

Recent research discussed the offloading problem in MEC-based architecture [17], where the offloading can take place from the core to edges and from the devices to edges. However, there are some major differences between this and our research. First, it only considered traffic allocation as a problem in the control plane. However, we consider not only traffic allocation in the control plane, but also capacity allocation in the management plane. Second, it only considered one service in the network while we include two services, one in core or edges, and the other in devices or edges. Third, it assumed that the traffic arrives simultaneously from devices. In our study, on the other hand, the arrival of packets follows a Poisson process, which is closer to the real environment. Finally, it only considered the nodes' workload to carry out the traffic allocation, but our study considers the latency percentage constraint.

Table I shows the comparisons between earlier studies and our work.

III. System Model and Problem Formulation

In this section, we introduce our system model, the notations used and the problem statement. Table II lists the notations and their meanings.

TABLE I
SUMMARY OF THE RESEARCH ON OFFLOADING

Paper	# of services	Target Network	Offloading direction	Problem in which plane	Objective	Constraints	Approach
[8]	1	Two-tier (Device/Cloud)	↑	Control	Min(E)	L	Game + Social Network
[9]		Two-tier (Device/Cloud)	↑	Control	Min(computation overhead)	T and E	Game theory
[10]		Two-tier (Device/Cloud)	↑	Control	Min(computation overhead)	T and E	Game theory
[11]		Two-tier (Device/Cloud)	↑	Control	Min(L)	E	Game theory (Stackelberg)
[12]		Two-tier (Device/Cloud)	↑	Control	Min(L)	E	Game theory
[13]		Two-tier (Device/Cloud)	↑	Control	L/E	L and E	Cost threshold
[14]	*m*	Two-tier (Device/Cloud)	↑	Control	Min(E)	T	Control Theory + Lyapunov
[15]		Two-tier (Device/Edge)	↑	Control	Min(E)	L	Successive Convex Approximation
[16]	*n*	Two-tier (Fog/Cloud)	↓	Control	Min(E) and Min(transmission cost)	L	Game theory (Stackelberg)
[17]	1	Three-tier (Device/Edge/Cloud)	↑ and ↓	Control	Min(L)	C	Branch and bound
TPIO	2	Three-tier (Device/Edge/Core)	↑ and ↓	Management & Control	Min(C)	L	Two-phase iterative optimization

L: Latency, T: Execution time, E: Energy consumption, C: capacity

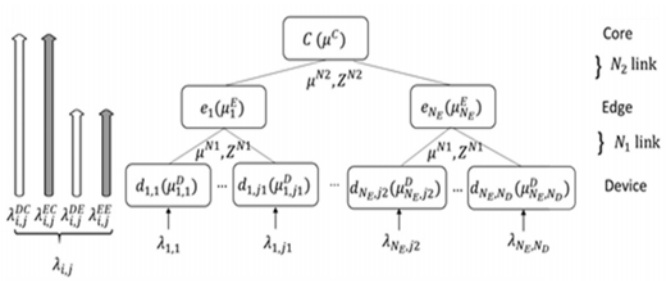


Fig. 1. Three-tier hierarchical MEC-based architecture.

A. System Model

We consider a three-tier hierarchical MEC-based architecture composed of a core, N_E edges and N_D devices, as shown in Fig. 1. C denotes the core, a single network device. E denotes edges and $e_i \in E$ denotes the i-th edge. D denotes the set of the set of devices and $d_{i,j} \in D$ denotes the j-th device connecting to the i-th edge. Each node has its own capacity, denoted by the symbol $\mu : \mu^C$, μ_i^E, and $\mu_{i,j}^D$ represent the capacity of the core, of edge e_i, and of device $d_{i,j}$, respectively. Let $N1$ denote the link between the device and the edge, and $N2$ denote the link between the edge and the core. μ^{N1} and μ^{N2} represent the link bandwidth of links $N1$ and $N2$, respectively. The propagation delays of links $N1$ and $N2$ are denoted as Z^{N1} and Z^{N2}, respectively. The traffic (in units of packets) arrival rate from each device follows a Poisson process. The packet arrival at device $d_{i,j}$ has rate $\lambda_{i.j}$. Each packet requires two services, one provided by the core and the other by the devices. According to MEC-based architecture, core and devices can offload their computation to edges, i.e., the edges deliver services provided by both the core and devices. Therefore, depending where the packet obtains these two services, traffic coming from device $d_{i,j}$, denoted as $g_{i,j}$, can be classified as four types. First, the traffic is served by devices and core, that is, no offloading. We denote the traffic as $g_{i,j}^{DC}$, with rate $\lambda_{i,j}^{DC}$. Second, the traffic obtains services from devices and edges, denoted as $g_{i,j}^{DE}$, with rate $\lambda_{i,j}^{DE}$. Third, the traffic obtains services from edges and core, denoted as $g_{i,j}^{EC}$, with rate $\lambda_{i,j}^{EC}$. Last, traffic obtains both services from edges, $g_{i,j}^{EE}$, with rate $\lambda_{i,j}^{EE}$. For simplicity, according to the classification above, these various traffics are termed DC-type traffic, EC-type traffic, DE-type traffic, and EE-type traffic.

The constrained latency is L. However, previous studies have always focused on achieving the goal where the average latency is less than this constraint. In fact, if such a goal is achieved, it represents that about a half traffic satisfies this constraint, and that about a half traffic exceeds the constraint. In this study, we use a more appropriate metric, termed the latency percentage constraint, i.e., the percentage of the latency constraint satisfied must exceed a threshold, ThL. For example, $ThL = 80\%$ means that at least 80% traffic has satisfied the latency constraint.

TABLE II
USED NOTATIONS

Notations	Meanings
C	Core
$E=\{e_i \mid 1 \le i \le N_E\}$	Edges
$D=\left\{d_{i,j} \middle\vert \begin{matrix}1 \le i \le N_E,\\ 1 \le j \le N_D\end{matrix}\right\}$	Devices
N_E	The number of edges, i.e., $N_E = \|E\|$
N_D	The number of devices, i.e., $N_D = \|D\|$
μ^C	Capacity of the core
μ_i^E	Capacity of the edge e_i
$\mu_{i,j}^D$	Capacity of the device $d_{i,j}$
μ^{N1}	Link bandwidth between device and edge
μ^{N2}	Link bandwidth between edge and core
Z^{N1}	The propagation delay between device to edge
Z^{N2}	The propagation delay between edge and core
$\lambda_{i.j}$	The total traffic arrival rate in $d_{i,j}$
$g_{i,j}^{XY}, X\epsilon\{D,E\}, Y\epsilon\{E,C\}$	The traffic served by X and Y in $d_{i,j}$
$\lambda_{i,j}^{XY}, X\epsilon\{D,E\}, Y\epsilon\{E,C\}$	The arrival rate for the traffic $g_{i,j}^{XY}$
$t_{i,j}^{XY}, X\epsilon\{D,E\}, Y\epsilon\{E,C\}$	The latency for the traffic $g_{i,j}^{XY}$
L	Constrained Latency
ThL	A threshold for the percentage of traffic satisfying latency constraint
TH1	A threshold to limit the iterations of phase 1
TH2	A threshold to limit the iterations of phase 2
ε	A threshold to determine the converge of TPIO algorithm

B. Problem Statement

Our objective for the system model considered in this paper is to minimize total capacity and to satisfy latency percentage constraint. To achieve this, we have to allocate capacity and traffic appropriately. The problem statement is then defined as follows.

Input: The topology composed of a core, N_E edges, N_D devices, the propagation delay of links (Z^{N1}, Z^{N2}), link bandwidth (μ^{N1}, μ^{N2}), traffic arrival rate $\lambda_{i.j}$, latency constraint L, and the threshold *ThL*.

Output: The allocated traffic vector $[\lambda_{i,j}^{DC}, \lambda_{i,j}^{DE}, \lambda_{i,j}^{EC}, \lambda_{i,j}^{EE}]$ and the allocated capacity $[\mu^C, \mu_i^E, \mu_{i,j}^D]$.

Objective: Minimize $\mu^{total} = \mu^C + \sum_i \mu_i^E + \sum_i \sum_j \mu_{i,j}^D$

Constraint: The percentage of traffic required to satisfy the latency constraint L must exceed the threshold *ThL*, that is, $P(t<L) \ge$ *ThL*, where t is the traffic latency.

As given in the problem statement, the allocated capacity and allocated traffic vector of four types are outputs. Thus, the problem is solved by an algorithm in the management plane, and our algorithm runs in a management computer, rather than devices, edges, or core. After determining the allocated traffic vector of four types, the values will be adopted by the devices, edges, and core, which execute suitable algorithms to route these packets in the control plane and schedule them in the data plane. However, these algorithms are beyond the scope of this paper.

IV. TWO-PHASE ITERATIVE OPTIMIZATION

TPIO has two interleaved phases for adjusting the capacity and traffic allocation in order to minimize the total capacity under satisfying the latency percentage constraint. In order to achieve this, we must derive an appropriate formula for latency. First, we derive the equations for latency distribution, then describe the TPIO concept, and finally provide a flow chart to show how the system operates.

A. Latency Distribution

When considering one server which may be a node or a link, it is assumed that packet arrivals follow a Poisson process with rate λ. Packet service time is an exponential distribution with a mean $1/\mu$. Under an M/M/1 queuing model, the probability density function (PDF) of the latency, t, and its cumulative distribution function (CDF) can be represented as

$$f(t) = (\mu - \lambda) e^{-(\mu-\lambda)\times t}, \tag{1}$$

$$F(L) = P(t \le L) = \left\{ \begin{matrix} 1 - e^{-(\mu-\lambda)\times L}, & L \ge 0 \\ 0, & L < 0 \end{matrix} \right\}. \tag{2}$$

However, the traffic in MEC-based architecture requires many servers and different types of traffic need to be served by different servers. Below we calculate the latency percentage constraint for each type of traffic.

1) DC-Type Traffic: First, we consider DC-type traffic which is served by a device and the core. DC traffic must travel to $N1$ link and $N2$ link, and is regarded as being served by four servers: device, $N1$ link, $N2$ link, and core. The propagation delay in $N1$ link is fixed as Z^{N1} and the propagation delay in $N2$ link is fixed as Z^{N2}. Since constrained latency is L, the latency spent in these four servers must be less than $L' = L - Z^{N1} - Z^{N2}$, where L' is constrained latency L without the propagation delay of any links. The CDF of latency for DC-type traffic arriving in $d_{i,j}$ can be expressed by

$$\begin{aligned} F_{i,j}^{DC}(L) = P(t \le L) = & \int_0^{L'} \int_0^{L'-t_1} \int_0^{L'-t_1-t_2} f_{i,j}^D(t_1) \times f^C(t_2) \\ & \times f^{N2}(t_3) \\ & \times \left(1 - e^{-\left(\mu^{N1} - \lambda_i^{N1}\right)\times (L'-t_1-t_2-t_3)}\right) dt_3\, dt_2\, dt_1, \end{aligned}$$

$$\text{where } f_{i,j}^D = \left(\mu_{i,j}^D - \lambda_{i,j}^D\right) \times e^{-\left(\mu_{i,j}^D - \lambda_{i,j}^D\right)\times t},$$

$$f^C(t) = \left(\mu^C - \lambda^C\right) \times e^{-\left(\mu^C - \lambda^C\right)\times t},$$

$$f^{N2}(t) = \left(\mu^{N2} - \lambda^{N2}\right) \times e^{-\left(\mu^{N2} - \lambda^{N2}\right)\times t}. \tag{3}$$

$f_{i,j}^D(t)$, $f^C(t)$, $f^{N2}(t)$ are the PDF of the latency for device $d_{i,j}$, core, and the $N2$ link, respectively. In Eq. (3) the latency spent in the device $d_{i,j}$ is t_1, the latency spent in the core is t_2, the latency spent in the $N2$ link is t_3, and the residual latency spent in the $N1$ link. To obtain the PDF of each server, we must calculate the arrival rate of this server, which is the sum of its arrival rates, i.e., $\lambda_{i,j}^D = \lambda_{i,j}^{DC} + \lambda_{i,j}^{DE}$, $\lambda^C = \sum_i \sum_j (\lambda_{i,j}^{DC} + \lambda_{i,j}^{EC})$. The arrival rate of the link is the total traffic rate going through this link, which is, $\lambda_i^{N1} = \sum_j \lambda_{i,j}$ and $\lambda^{N2} = \sum_i \sum_j (\lambda_{i,j}^{DC} + \lambda_{i,j}^{EC})$. Note that

TABLE III
ALLOCATING TRAFFIC GUIDELINE

Not satisfying constraint	Max(latency)	Shift traffic to
$g_{i,j}^{DC}$	Device	$g_{i,j}^{EC}$
	Edge	no shift
	Core	$g_{i,j}^{DE}$
$g_{i,j}^{DE}$	Device	$g_{i,j}^{EE}$
	Edge	$g_{i,j}^{DC}$
	Core	no shift
$g_{i,j}^{EC}$	Device	no shift
	Edge	$g_{i,j}^{DC}$
	Core	$g_{i,j}^{EE}$
$g_{i,j}^{EE}$	Device	$g_{i,j}^{EC}$
	Edge	$g_{i,j}^{DC}$
	Core	$g_{i,j}^{DE}$

only the propagation delay of links is assumed as a fixed value. However, we actually consider the packet arrival rate and link bandwidth to calculate the transmission delay. As the network latency in the link includes transmission delay and propagation delay, the network latency is actually dependent of the link bandwidth and the packet arrival rate.

2) EC-Type Traffic: EC-type traffic is served by an edge and the core, so that it must travel to links $N1$ and $N2$. Similar to DC-type traffic, the CDF of latency for EC-type traffic arriving in $d_{i,j}$ can be expressed by

$$F_{i,j}^{EC}(L) = P(t \le L) = \int_0^{L'} \int_0^{L'-t_1} \int_0^{L'-t_1-t_2} f_i^E(t_1) \times f^C(t_2) \times f^{N2}(t_3) \times \left(1 - e^{-(\mu^{N1} - \lambda_i^{N1}) \times (L' - t_1 - t_2 - t_3)}\right) dt_3\, dt_2\, dt_1,$$
$$\text{where } f_i^E(t) = \left(\mu_i^E - \lambda_i^E\right) \times e^{-(\mu_i^E - \lambda_i^E) \times t}. \quad (4)$$

$f_i^E(t)$ is the PDF of the latency for edge e_i. Its arrival rate can be calculated as $\lambda_i^E = \sum_j (\lambda_{i,j}^{DE} + \lambda_{i,j}^{EC} + 2\lambda_{i,j}^{EE})$. $f^C(t)$ and $f^{N2}(t)$ are the same as in Eq. (3).

3) DE-Type Traffic: DE-type traffic only travels to $N1$ link, but not to $N2$ link. It is thus regarded as being served by three servers: device, link $N1$, and edge. Since constrained latency is L, the latency spent in these three servers must be less than $L' = L - Z^{N1}$, where L' is constrained latency L without the propagation delay of $N1$ link. The CDF of latency for DE-type traffic arriving in $d_{i,j}$ can be expressed by

$$F_{i,j}^{DE}(L) = P(t \le L) = \int_0^{L'} \int_0^{L'-t_1} f_{i,j}^D(t_1) \times f_i^E(t_2) \times \left(1 - e^{-(\mu^{N1} - \lambda_i^{N1}) \times (L' - t_1 - t_2)}\right) dt_2\, dt_1. \quad (5)$$

4) EE-Type Traffic: Similar to DE-type, EE traffic only travels to link $N1$, but not to link $N2$. Thus, the CDF of latency for EE-type traffic arriving in $d_{i,j}$ can be expressed by

$$F_{i,j}^{EE}(L) = P(t \le L) = \int_0^{L'} \int_0^{L'-t_1} f_i^E(t_1) \times f_i^E(t_2) \times \left(1 - e^{-(\mu^{N1} - \lambda_i^{N1}) \times (L' - t_1 - t_2)}\right) dt_2\, dt_1. \quad (6)$$

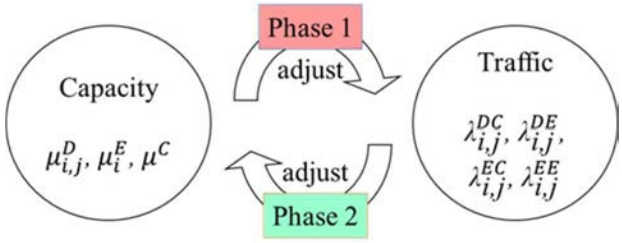


Fig. 2. The TPIO concept.

From Eqs. (3)∼(6), we can derive the CDF of latency for traffic arriving in $d_{i,j}$ by

$$F_{i,j}(L) = \frac{F_{i,j}^{DC}(L)\lambda_{i,j}^{DC} + F_{i,j}^{EC}(L)\lambda_{i,j}^{EC} + F_{i,j}^{DE}(L)\lambda_{i,j}^{DE} + F_{i,j}^{EE}(L)\lambda_{i,j}^{EE}}{\lambda_{i,j}}. \quad (7)$$

Also we can derive the CDF of latency for all traffic as

$$F(L) = \frac{\sum_i \sum_j F_{i,j}(L)\lambda_{i,j}}{\lambda}. \quad (8)$$

After obtaining $F(L)$, the objective is to satisfy $F(L) \ge ThL$.

B. The TPIO Concept

TPIO has two iterative phases for adjusting capacity and traffic allocation. As shown in Fig. 2, in Phase 1 we adjust traffic allocation based on current allocated capacity to satisfy the latency percentage constraint. In Phase 2, we adjust capacity allocation based on current allocated traffic to minimize total capacity.

1) Phase 1 (Low-Latency Traffic Allocation): Using Eqs. (3)∼(6), we determine the traffic types which do not satisfy the latency constraint in each device. For such traffic types, latency should be lowered. Thus, TPIO determines to shift the traffic types which do not satisfy latency percentage constraint to other types. The decision is based on the node with the maximum latency. Table III shows this in outline. For example, the traffic $g_{i,j}^{DC}$ does not satisfy the latency percentage constraint, meaning that we should shift some of $g_{i,j}^{DC}$ to another type. If the core now experiences maximum latency, TPIO will shift some $g_{i,j}^{DC}$ to $g_{i,j}^{DE}$ to prevent over-congestion in the core. Thus, $\lambda_{i,j}^{DC}$ will decrease and be more likely to satisfy its latency percentage constraint after the shift.

When it is known which traffic should be shifted, TPIO determines how much traffic is to be shifted. In order to achieve rapid converge in phase 1, the amount of traffic being shifted is determined according to the appropriate equation of Eqs. (3)∼(6). Thus, we can then determine the amount of traffic to be shifted to satisfy the latency percentage constraint.

For example, if we want to shift some $g_{i,j}^{DC}$ to $g_{i,j}^{DE}$, we select the corresponding Eq. (3) and set $F_{i,j}^{DC}(L) = ThL$ to determine the new $\lambda_{i,j}^{DC}$ and $\lambda_{i,j}^{DE}$.

2) Phase 2 (Low-Capacity Allocation): Traffic allocation has thus been determined after phase 1. However, under current traffic allocation, the capacity may be further reduced to reduce overall capacity. To do so we select the node which

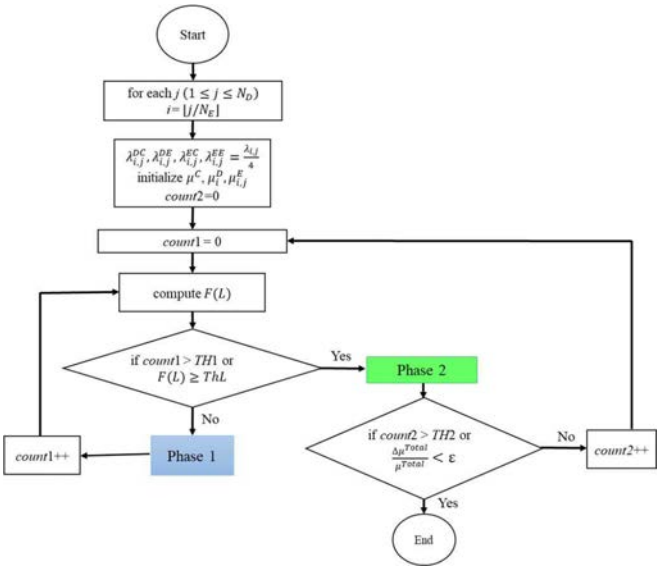


Fig. 3. Flow chart of TPIO.

has the highest capacity reduction to reduce total capacity. For example, if reducing device capacity can generate the largest capacity reduction while satisfying the latency percentage constraint, TPIO will adjust the devices' capacity to rapidly reach the objective of minimum capacity.

C. The TPIO Algorithm

The TPIO algorithm is shown in Fig. 3. Initially, depending on the number of devices and edges, TPIO constructs the hierarchical MEC-based architecture by almost uniformly mapping the devices to edges. Each edge is thus responsible for $\lfloor N_D/N_E \rfloor$ devices except the last one. The device j has the parent edge $i = \lfloor j/\lfloor N_D/N_E \rfloor \rfloor$. For clarity, we use the index (i, j), rather than ($\lfloor j/\lfloor N_D/N_E \rfloor \rfloor$, j) for used notations.

First, the percentages of four types are initially set to be the same, i.e., $\lambda^{DC}_{i,j} = \lambda^{DE}_{i,j} = \lambda^{EC}_{i,j} = \lambda^{EE}_{i,j} = \frac{1}{4}\lambda_{i,j}$. Thus we can determine the arrival rate of each node as

$$\begin{aligned} \lambda^D_{i,j} &= \lambda^{DC}_{i,j} + \lambda^{DE}_{i,j}, \\ \lambda^E_i &= \sum_j \left(\lambda^{DE}_{i,j} + \lambda^{EC}_{i,j} + 2\lambda^{EE}_{i,j}\right), \\ \lambda^C &= \sum_i \sum_j \left(\lambda^{DC}_{i,j} + \lambda^{EC}_{i,j}\right). \end{aligned} \tag{9}$$

TPIO then determines the initial capacity of each node, $\mu^C, \mu^E_i, \mu^D_{i,j}$. It assumes the traffic takes the same time in each node and ignores the latency spent in links to let the allocated capacity be enough. That is, the latency in each node is $L/2$. Thus, according to Eq. (2), the initial allocated capacity for the core, edges, and devices is

$$\begin{aligned} \mu^C &= \lambda^C - \left(\frac{log_e(1-ThL)}{L/2}\right), \\ \mu^E_i &= \lambda^E_i - \left(\frac{log_e(1-ThL)}{L/2}\right), \\ \mu^D_{i,j} &= \lambda^D_{i,j} - \left(\frac{log_e(1-ThL)}{L/2}\right). \end{aligned} \tag{10}$$

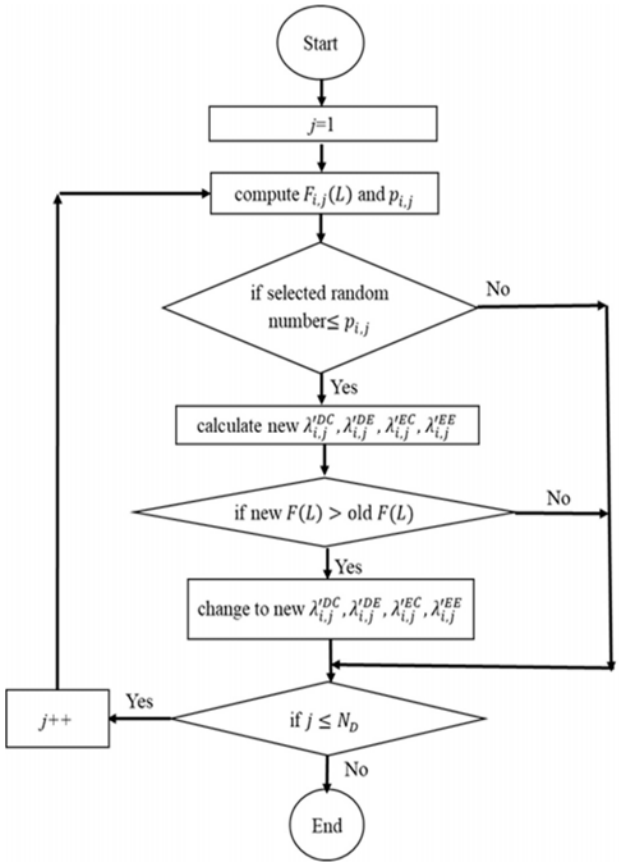


Fig. 4. Flow chart of phase 1.

Next, a loop for iteratively calculating traffic and capacity allocation is repeated until a satisfactory result is obtained. In Fig. 3, phase 1 adjusts traffic allocation and phase 2 adjusts capacity allocation. We compute $F_{i,j}(L)$, the percentage of traffic satisfying the latency constraint in device $d_{i,j}$ according to Eq. (7). Then we calculate $F(L)$ to determine the percentage of all traffic satisfying L. Comparing $F(L)$ with ThL to determine whether current traffic allocation can satisfy the latency percentage constraint. If $F(L) \ge ThL$, i.e., the traffic allaocation satisfies the latency percentage constraint, TPIO will enter phase 2 to reduce the allocated capacity. However, phase 1 might not get the proper traffic allocation when the allocated capacity is too small. To solve this problem, we count the number of iterations in phase 1, i.e., $count1$. If this number is larger than a threshold, $TH1$, it means that current allocated capacity cannot satisfy the latency percentage constraint, and so TPIO enters phase 2.

The stop criteria is when $\Delta\mu^{total}/\mu^{total}$ is less than a tiny value, ϵ, where $\Delta\mu^{total}$ is the capacity variation and μ^{total} is the total capacity. In this case, the capacity reduction ratio is less than a tiny value. Also a threshold, $TH2$, is used to limit the number of iterations of phase 2, $count2$.

1) Phase 1 (Low-Latency Traffic Allocation): The detailed flow chart for phase 1 is shown in Fig. 4. We first compute $F_{i,j}(L)$, the percentage of traffic that satisfies the latency constraint in device $d_{i,j}$ according to Eq. (7). As currently $F(L)$ is smaller than ThL, there are some devices with $F_{i,j}(L) < ThL$.

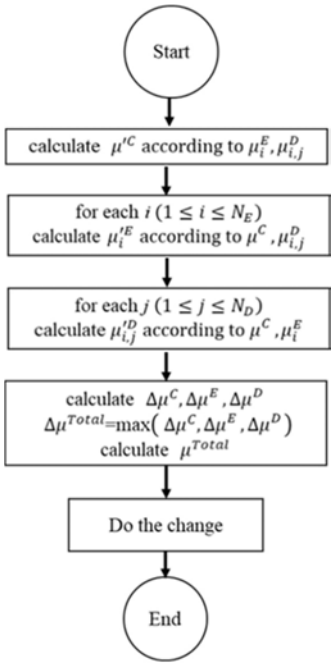


Fig. 5. Flow chart of phase 2.

Which we select as the victims, and then calculate the probability that these victims will adjust their traffic allocation. The probability of changing traffic allocation in device $d_{i,j}$, denoted as $p_{i,j}$, is calculated as

$$p_{i,j} = \max\left(1 - \frac{F_{i,j}(L)}{ThL}, 0\right). \tag{11}$$

If $F_{i,j}(L)$ is much less than *ThL*, it is more likely to shift the traffic in device $d_{i,j}$. According to the probability $p_{i,j}$, if a victim determines it necessary to adjust traffic allocation, TPIO adjusts the traffic according to the guidelines in Table III, and computes the amount of shifted traffic by the corresponding equation among Eqs. (3)∼(6). For example, if the shifted traffic is DC-type traffic, TPIO uses $F_{i,j}^{DC}(L) = ThL$ to reduce this and thus obtains a changed traffic allocation $(\lambda_{i,j}'^{DC}, \lambda_{i,j}'^{DE}, \lambda_{i,j}'^{EC}, \lambda_{i,j}'^{EE})$. After adjusting the traffic for each victim, we compute a new $F(L)$ and determine whether it is better than the old $F(L)$. If so, TPIO replaces $(\lambda_{i,j}^{DC}, \lambda_{i,j}^{DE}, \lambda_{i,j}^{EC}, \lambda_{i,j}^{EE})$ with $(\lambda_{i,j}'^{DC}, \lambda_{i,j}'^{DE}, \lambda_{i,j}'^{EC}, \lambda_{i,j}'^{EE})$. On the other hand, if the new $F(L)$ is worse than the old one, TPIO retains $(\lambda_{i,j}^{DC}, \lambda_{i,j}^{DE}, \lambda_{i,j}^{EC}, \lambda_{i,j}^{EE})$ without making any changes.

2) Phase 2 (Low Capacity Allocation): The detailed flow chart for phase 2 is shown in Fig. 5. Here we reduce the node which has the maximum capacity reduction to reduce the total capacity. First, TPIO computes the new capacity μ'^C with current μ_i^E and $\mu_{i,j}^D$ after satisfying the latency percentage constraint according to Eq. (8). Similarly, TPIO computes the new $\mu_i'^E$ with current μ^C and $\mu_{i,j}^D$, and computes the new capacity $\mu_{i,j}'^D$ with current μ^C and μ_i^E. Comparing the capacity reduction, where $\Delta\mu^C = (\mu^C - \mu'^C)$, $\Delta\mu^E = \sum_i(\mu_i^E - \mu_i'^E)$, and $\Delta\mu^D = \sum_i\sum_j(\mu_{i,j}^D - \mu_{i,j}'^D)$, phase 2 determines which capacity should be reduced. TPIO selects the maximum capacity reduction to carry out the adjustment. For example, if the core's capacity can be reduced the most, TPIO will reduce its capacity, so that capacity allocation is changed to $(\mu'^C, \mu_i^E, \mu_{i,j}^D)$. To determine the terminiation conditon, phase 2 also calculates $\Delta\mu^{total} = \max(\Delta\mu^C, \Delta\mu^E, \Delta\mu^D)$ and $\mu^{total} = \mu^C + \sum_i \mu_i^E + \sum_i\sum_j \mu_{i,j}^D$.

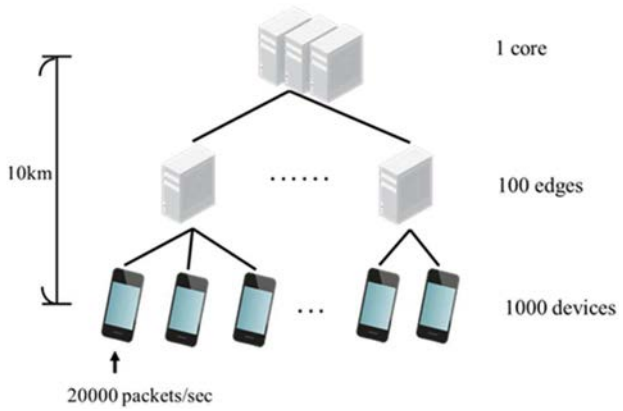


Fig. 6. Simulation scenario.

D. The Time Complexity of the TPIO Algorithm

The TPIO is a two-phase algorithm. As can be seen in Fig. 4, as the algorithm of phase 1 calculates the traffic allocation once for each device, its complexity is O(N_D). Figure 3 shows that TPIO repeats phase 1 many times per phase 2. Since we use a good heuristic to shift the traffic, the average count (*count*1) in phase 1 is not too large, as shown in Section IV-B. Also a threshould *TH*1 is set to limit the times of its iterations. Thus for each phase 2, the time complexity of phase 1 in the worst case is O($N_D \times$ *TH*1).

From Fig. 5 can be seen that the complexity of phase 2 is O(1+N_E+N_D) because it calcutes the capacity reduction for each node, including core, edges and devices. However, the difficulty is to determine how many times of phase 2, but we could not obtain an exact number. However, as we use Eq. (8) to calculate how much capacity can be reduced under current traffic allocation, the capacity reduction might be large at first, but not subsequently. Therefore, the iteration times of phase 2 (*count*2) is not large and TPIO can rapidly converge, as shown in Section IV-B. Also, a threshould *TH*2 is set to limit the number of iterations. Thus, the complexity of TPIO in the worst case can be estimated as O($TH2 \times (N_D \times TH1 + (1 + N_E + N_D))) =$ O($TH2 \times TH1 \times N_D$), because N_D is always larger than N_E.

V. Evaluation

In this section we deal with some simulations to investigate the performance of TPIO. First, we describe the simulation scenario and default parameters, and then investigate the effects of some important parameters. Finally, we give some lessons learned.

A. Scenarios and Parameters

The simulation scenario is a three-tier hierarchical network as shown in Fig. 6, comprising a core, 100 edges, and

TABLE IV
DEFAULT VALUES FOR SIMULATION

Parameters	Default Value
Number of core	1
Number of edges	100
Number of devices	1000
Distance (devices to core)	10 km
Distance (devices to edges)	1 km (when N_E=100)
Packet size	1 Kb
Traffic (per device)	20000 packets/sec
Latency constraint	1 ms
ThL	80%
Link bandwidth (edges to core)	10 Gbps
Link bandwidth (devices to edges)	1 Gbps
*TH*1	100
*TH*2	100
ε	1%

1000 devices. Each edge covers 10 devices and each device has traffic arriving at a rate of 20,000 packets/sec. The distance between the core and the devices is 10 km, so the area of the simulation scenario is $10^2\pi$ km^2. All devices are assumed to be uniformly distributed. Thus, the distance between edges and devices is determined by the number of devices that each edge covers. If the number of edges is varied, the number of devices that each edge covers varies and the distance between edges and devices also varies. Thus, when the distance between the devices and the core is X^{DC}, the distance between the devices and the edges, X^{DE}, can be calcauted as

$$\frac{X^{DE} \times X^{DE}}{\left\lfloor \frac{N_D}{N_E} \right\rfloor} = \frac{X^{DC} \times X^{DC}}{N_D},$$

where $\lfloor \frac{N_D}{N_E} \rfloor$ is the number of devices that each edge covers except the last edge.

The latency constraint follows 5G network guidelines and is set at 1 ms [3]. The link between core and edges is considered as a fiber network, so that the propagation speed is 3×10^8 m/s and the link bandwidth is 10 Gbps. The link between edges and devices is considered as a 5G network, so that bandwidth is set as 1 Gbps. Table IV summarizes the parameters and default values used in this experiment.

B. Results

1) The Performance of TPIO: Figure 7 shows the comparison between the capacity of three-tier and two-tier architectures with different numbers of edges. The unit of Y-axis is Mpackets/sec, i.e., 10^6 packets/sec. The middle tier has edges in three-tier architecture, while the base station in two-tier architecture has no computing capacity. The capacity in three-tier architecture is obviously lower than that of two-tier architecture with the same parameters. Three-tier architecture reduced the capacity by 20.7% ($\frac{50.7-40.2}{50.7}$) at most when $N_E = 50$, which confirmed the necessity of edges.

With only one edge, its location is the same as that of the core. Intuitively, three-tier architecture will be reduced to

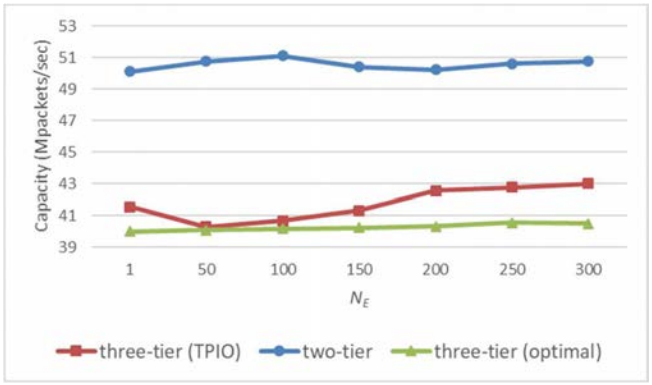


Fig. 7. The performance comparison between three-tier(TPIO), three-tier(optimal) and two-tier.

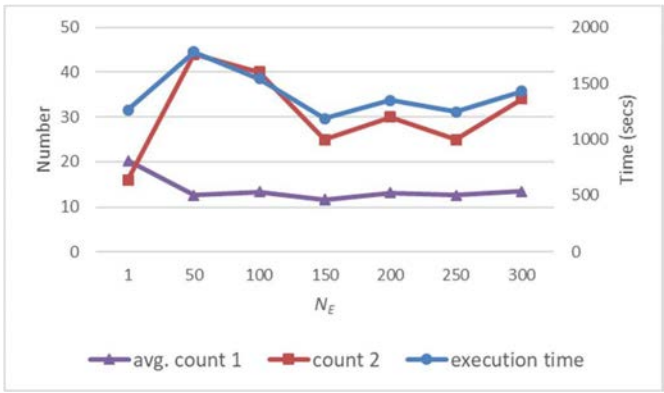


Fig. 8. The time complexity of TPIO.

two-tier architecture and they should have the same allocated capacity. However, even in this case, three-tier architecture still has lower capacity than two-tier architecture. This is because the computation offloading from devices to edges is possible in three-tier architecture, but offloading from devices to core is impossible in two-tier architecture. Offloading computation from devices to edges can significantly decrease the amount of allocated capacity.

Figure 7 also compares the performance of TPIO and the optimal solution, which is solved by a brute-force approach and a time-consuming procedure, in three-tier architecture. We observed that the optimal solutions under different numbers of edges were almost the same. With fewer edges, the capacity of an edge became more concentrated and thus reduced required capacity. On the other hand, in this case the distance between the devices and edges increased, so the capacity of this architecture grew to compensate for this increase in propagation delay. Thus, the positive and negative effects resulted in similar optimal solutions. TPIO also achieved very close to the optimal solution when $N_E = 50$ and a slightly higher capacity by 6.2% ($\frac{43.0-40.5}{40.5}$) at most when $N_E = 300$.

Figure 8 shows the efficiency of TPIO by giving the average number of iterations of phase 1 per phase 2 (average *count*1), the number of iterations of phase 2 (*count*2), and execution time. The average *count*1 was always less than 20 and *count*2 ranged between 15 and 45. Thus we set both *TH*1 and *TH*2 as 100. The specifications of the computer running

the TPIO algorithm are Intel i5-3230M CPU, 4GB RAM, with Windows 10 operating system. The execution time of 1700 seconds at most verifies the acceptable complexity of TPIO.

2) The Effects of the Number of Edges: From Fig. 9(a): when fewer edges existed, TPIO generated less capacity because the capacity of an edge became more concentrated and thus reduced capacity. When the number of edges was too small, for example, $N_E = 1$, the distance between edges and devices was far, resulting in the increase of the required capacity. Thus, the number of edges should be properly selected. We also observed that most capacity was allocated to edges, which confirmed the necessity of edges.

From Fig. 9(b): most traffic was $\lambda_{i,j}^{EE}$, showing that most traffic obtained two functions in edges, which is why μ_i^E was large. When N_E was small, as the distance between edges and devices was large, there was some traffic belonging to λ^{DC}. However, when N_E increased, as edges were closer to devices, λ^{DC} was shifted to λ^{DE} and λ^{EE}, resulting in the decrease of λ^{DC} and the increases of λ^{DE} and λ^{EE}.

3) The Effects of Latency Constraint: The latency constraint in 5G is of the order of ms and will affect the performance of MEC-based architecture. Figure 10(a) shows the capacity allocation under different latency constraints. It is reasonable that a higher latency constraint will result in higher capacity. Note too that the latency constraint is more likely to be satisfied when traffic is served by the devices. Therefore, when latency constraint becomes loose, the capacity of devices decreases to reduce the capacity. Furthermore, the capacity of edges occupies a significantly major portion of total capacity. The capacity ratio of edges was always more than 70%, irrespective of the latency constraint.

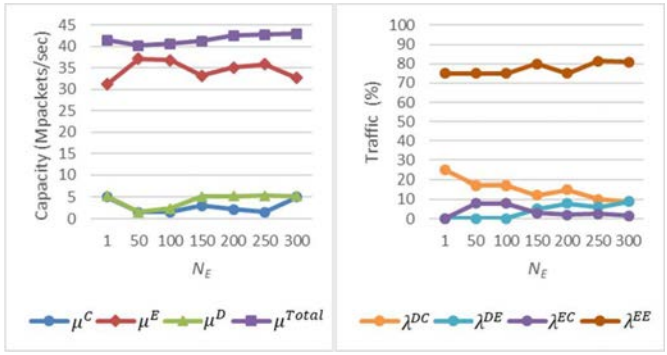
As we know, offloading computation to a higher tier can reduce total capacity, but it may cause longer latency. Thus, traffic should preferentially be served by a higher than by a lower tier when latency constraint is loose, to reduce the capacity. As shown in Fig. 10(b), λ^{DC} was shifted to λ^{EC} by about 25% when latency constraint increased from 0.5ms to 2.5ms.

4) The Effects of ThL: Figure 11(a) shows the capacity allocation under different *ThL*s. The total capacity increased by 12.2% as *ThL* increased from 50% to 90%. It is reasonable that a larger capacity is accompanied by a higher QoE. This figure also shows the trade-off between capacity and QoE.

As we know, offloading computation to a higher tier can reduce total capacity, but it may also cause longer latency. Furthermore, when a high *ThL* is set, the traffic must be served in a lower tier to allow more traffic to satisfy the latency percentage constraint. As shown in Fig. 11(b), the traffic type λ^{DC} and λ^{EC} were reduced with increasing *ThL*. λ^{DC} and λ^{EC} were shifted to λ^{EE} when *ThL* increased from 50% to 90%.

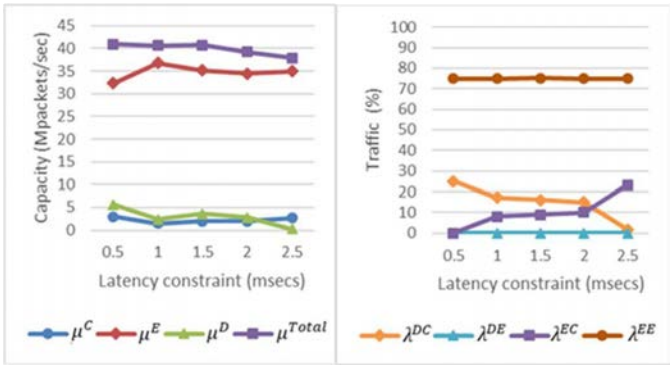
An interesting phenomenon can be observed from Fig. 11(a): the increase in the total capacity was mainly due to the edge capacity when *ThL* changed. From Fig. 11(b) can be seen that λ^{DC} and λ^{EC} were shifted to λ^{EE} when *ThL* changed from 50% to 90%. As λ^{EE} was increased, more capacity in edges had to be allocated.

5) The Effects of the Distance Between Devices and Core: Figure 12(a) shows the capacity under different distances between devices and the core, i.e., X^{DC}. The total capacity increased slightly as X^{DC} increased from 10km to 50km because the extra propagation delay of 0.13ms was needed. We can see that the capacity of core decreased and the capacity of edges increased to avoid this propagation delay. As shown in Fig. 12(b), when X^{DC} became longer, the traffic type λ^{DC} was shifted to λ^{DE} or λ^{EE}. Also the traffic type λ^{EC} was also shifted to λ^{DE} or λ^{EE}. That is, when the distance between



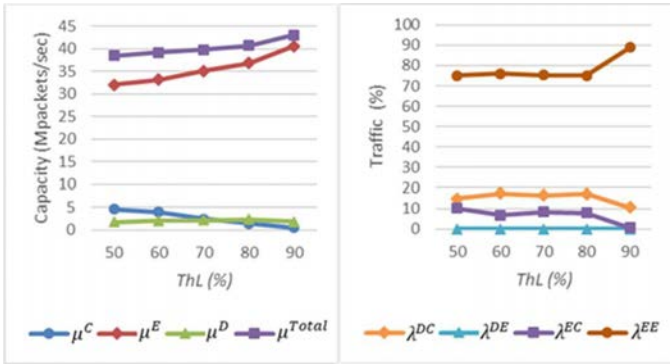
(a) capacity (b) traffic

Fig. 9. The effects of the number of edges.



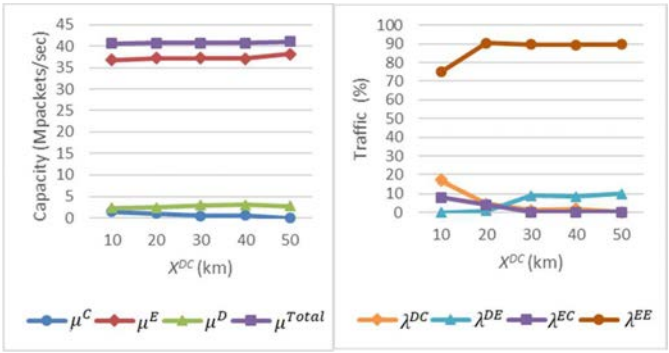
(a) capacity (b) traffic

Fig. 10. The effects of latency constraint.



(a) capacity (b) traffic

Fig. 11. The effects of *ThL*.



(a) capacity (b) traffic

Fig. 12. The effects of X^{DC}.

devices and the core was far, no traffic got its function in the core.

C. Lessons Learned

Some interesting lessons can be learned from these results.

(1) The edges are essential for satisfying the latency percentage constraint and reducing the allocated capacity. If using two-tier architecture, more capacity must be allocated.

(2) The number of edges should be properly selected. Basically, too many edges cause more capacity, while too few edges make the distance between devices and edges too far, also resulting in more capacity.

(3) Most capacity, always more than 70%, is allocated to the edges.

(4) When the latency percentage constraint becomes stricter, however, the increase of capacity is not as large. Thus, *ThL* can be set as a reasonable value, for example, 90%.

(5) When the distance between the core and devices becomes longer, more capacity should be allocated to devices and edges to avoid propagation delays between edges and the core. In this case, no traffic gets its function in the core. Thus, it is meaningless to place the core in a location too far away from devices.

VI. Conclusion

This paper proposes TPIO to optimize capacity and traffic allocation under the constraint of ultra-low latency in a three-tier MEC-based network. TPIO iteratively uses two phases to adjust capacity and traffic allocation because they are tightly-coupled. Given fixed capacity allocation, we use queuing theory to calculate optimal traffic allocation. On the other hand, given fixed traffic allocation, allocated capacity is further reduced under satisfying latency percentage constraint. TPIO exploits queuing theory to calculate latency and its distribution to obtain the percentage of traffic that satisfies latency constraint and decides whether or not it is below *ThL*.

A simulation exercise shows that MEC-based architecture with TPIO can save up to 20.7% of capacity compared to two-tier architecture. This result shows the necessity of a tier of edges. The superiority of TPIO can be observed because its performance is close to the optimal solution and it can significantly reduce time complexity. The role of edges is necessary because they can offload traffic originally served by the core, devices, or both. Another important consequence is the trade-off between the latency percentage constraint and capacity. Additional capacity of 12.2% will be incurred when *ThL* is 90%, compared to that when *ThL* is 50%. Furthermore, λ^{DC} and λ^{EC} are shifted to λ^{EE} when *ThL* is changed from 50% to 90%. Also, the distance between the core and devices is an important factor to affect the capacity and traffic allocation.

Despite being promising, there is much work to be done in the future. First, instead of using two services, a scenario with more diverse services should be considered. Second, the wireless link bandwidth between devices and edges should be extended to a dynamic value instead of a fixed one used in this paper. Finally, the issue about service chain deployment in MEC-based architecture could be further investigated.

References

[1] *5G Related Aspects in ITU-R Working Party 5D (Responsible Group for Terrestrial IMT in ITU-R)*. Accessed: Oct. 14, 2016. [Online]. Available: https://www.itu.int/en/membership/Documents/missions/GVA-mission-briefing-5G-28Sept2016.pdf

[2] N. A. Johansson, Y.-P. E. Wang, E. Eriksson, and M. Hessler, "Radio access for ultra-reliable and low-latency 5G communications," in *Proc. IEEE ICC Workshop 5G Beyond Enabling Technol. Appl.*, Jun. 2015, pp. 1184–1189.

[3] G. P. Fettweis, "The tactile Internet: Applications and challenges," *IEEE Veh. Technol. Mag.*, vol. 9, no. 1, pp. 64–70, Mar. 2014.

[4] J. Zhang, W. Xie, F. Yang, and Q. Bi, "Mobile edge computing and field trial results for 5G low latency scenario," *China Commun.*, vol. 13, no. 2, pp. 174–182, Feb. 2016.

[5] T. O. Olwal, K. Djouani, and A. M. Kurien, "A survey of resource management toward 5G radio access networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1656–1686, 3rd Quart., 2016.

[6] Y. Yu, "Mobile edge computing towards 5G: Vision, recent progress, and open challenges," *China Commun.*, vol. 13, no. 2, pp. 89–99, Feb. 2016.

[7] (Sep. 18, 2014). *Mobile-Edge Computing Introductory Technical White Paper*. [Online]. Available: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing-introductory_technical_white_paper_v1%2018-09-14.pdf

[8] L. Tang, X. Chen, and S. He, "When social network meets mobile cloud: A social group utility approach for optimizing computation offloading in cloudlet," *IEEE Access*, vol. 4, pp. 5868–5879, 2016.

[9] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2015.

[10] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[11] Y. Li, P. Wang, D. Niyato, and Z. Han, "A hierarchical cooperation formation model for downlink data transmission in mobile infostation networks," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 144–152, Jun. 2013.

[12] O. Muñoz, A. P. Iserte, J. Vidal, and M. Molina, "Energy-latency trade-off for multiuser wireless computation offloading," in *Proc. IEEE Workshops Wireless Commun. Netw. (WCNCW)*, 2014, pp. 29–33.

[13] H. Wu, Q. Wang, and K. Wolter, "Tradeoff between performance improvement and energy saving in mobile cloud offloading systems," in *Proc. IEEE Conf. Commun. (ICC)*, Jun. 2013, pp. 728–732.

[14] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.

[15] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[16] H. Zhang *et al.*, "Fog computing in multi-tier data center networks: A hierarchical game approach," in *Proc. IEEE Conf. Commun. (ICC)*, 2016, pp. 1–6.

[17] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jun. 2016, pp. 1–9.