This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2937342, IEEE Transactions on Network and Service Management

Workload and Capacity Optimization for Cloud-Edge Computing Systems with Vertical and Horizontal Offloading

Minh-Tuan Thai, Ying-Dar Lin, *Fellow, IEEE,* Yuan-Cheng Lai, and Hsu-Tung Chien

***Abstract*—A collaborative integration between cloud and edge computing is proposed to be able to exploit the advantages of both technologies. However, most of the existing studies have only considered two-tier cloud-edge computing systems which merely support vertical offloading between local edge nodes and remote cloud servers. This paper thus proposes a generic architecture of cloud-edge computing with the aim of providing both vertical and horizontal offloading between service nodes. To investigate the effectiveness of the design for different operational scenarios, we formulate it as a workload and capacity optimization problem with the objective of minimizing the system computation and communication costs. Because such a mixed-integer nonlinear programming (MINLP) problem is NP-hard, we further develop an approximation algorithm which applies a branch-and-bound method to obtain optimal solutions iteratively. Experimental results show that such a cloud-edge computing architecture can significantly reduce total system costs by about 34%, compared to traditional designs which only support vertical offloading. Our results also indicate that, to accommodate the same number of input workloads, a heterogeneous service allocation scenario requires about a 23% higher system costs than a homogeneous scenario.**

***Index Terms*—capacity optimization, edge computing, fog computing, optimization, workload offloading.**

I. Introduction

Cloud service providers traditionally rely on massive data centers to provide virtualized computational resources to users with the advantages of high availability and rapid elasticity [1]. Unfortunately, such an approach is not capable of accommodating real-time and delay-sensitive services, such as high-quality video streaming or instant emergency alarms, which require immediate responses. In fact, the transmission of such service workloads to and from remote data centers results in long network latency which might significantly downgrade their performance. Furthermore, the data centers, which provide computational resources in a centralized manner, cannot handle the requirements of mobility and geo-distribution of mobile and Internet of things (IoT) services efficiently.

To overcome such issues of cloud computing, an edge computing paradigm [2]–[6] aims to process service workloads locally, near their sources. In other words, the concept allows deploying virtualized resources of computing and communication on mobile/IoT devices, network edges, and core servers by leveraging virtualization technologies. By doing so, the end-to-end delay for accessing the resources is reduced since the network distance between them and end-users is significantly shortened. Hence, edge computing is likely more suitable for real-time or delay-sensitive services than cloud computing. Furthermore, edge computing enables virtualized resources to be geographically distributed at network edges which can potentially address the requirements of mobility and geo-distribution of mobile and IoT services.

While cloud computing provides high availability, reliability, and good resource utilization, and has almost capacity limit, edge computing offers mobility and enhances performance. As a result, integration of cloud with edge computing has been proposed to exploit the advantages of both these technologies [7]. Conceptually, the approach allocates computation and communication virtualized resources into th hierarchy, such as to end device, network edge, central office, and data center tiers, to handle incoming service workloads. By doing so, cloud-edge computing can efficiently accommodate different types of services whose characteristics are diverse while maintaining availability and reliability. In fact, although end devices and network edges are suitable for real-time and delay-sensitive services, they may not be able to handle services which demand a significant amount of computing capacity. The workloads of such services must thus be redirected to central offices or data centers for executions.

In practice, a cloud-edge computing system must serve a vast, diverse, and ever-increasing volume of service workloads. It is thus crucial and challenging to manage its capital expenditure (CAPEX) and operating expenses (OPEX) efficiently, while guaranteeing service quality (e.g., delay constraints). Research papers such as [7]–[14] have attempted to address these requirements of cloud-edge computing. However, these studies only considered simple two-tier systems, i.e., local network edge nodes and remote cloud servers, or neglected the horizontal offloading between service nodes in the same tier, such as device-to-device and edge-to-edge offloading. To this end, we first derive generic architecture of cloud-edge computing, taking into consideration both vertical and horizontal offloading between service nodes. We then model a workload and capacity optimization problem with the objective of minimizing system computation and communication costs. This would then allow us to study the architecture perfor-

Minh-Tuan Thai is with the Department of Information Technology, Can Tho University, Can Tho, Vietnam.

Ying-Dar Lin and Hsu-Tung Chien are with the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan.

Yuan-Cheng Lai is with the Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan.

Minh-Tuan Thai is the corresponding author (e-mail:tmtuan@cit.ctu.edu.vn).

Manuscript received

mance with different input workloads, service allocations, and capacity price scenarios. Since the problem is a mixed-integer nonlinear programming (MINLP) problem, which is very difficult to address, we develop a branch-and-bound algorithm to obtain optimal solutions iteratively. Experimental results show that our cloud-edge computing design, which supports both vertical and horizontal offloading, can significantly reduce system costs for various operational scenarios by about 34%, compared to a traditional approach which only provides vertical offloading.

Compared to related work, the contributions of this paper are across three aspects: (i) this paper is the first study, to the best of our knowledge, which considers both vertical and horizontal offloading in a cloud-edge computing system and formally models its architecture into an optimization problem; (ii) we derive a branch-and-bound algorithm to reduce the complexity and obtain optimal solutions for the problem; and (iii) various experimental simulations were conducted to investigate important operating scenarios along with significant observations.

Although the proposed algorithm can solve our optimization problem efficiently, its long-running time, especially in large-scale cloud-edge systems which consist of a large number of service nodes and network connections, would be a limitation of this work. Also, our architecture assumes that service nodes fairly treat all submitted service requests. Thus, it is not suitable for a situation in which some service nodes with special capabilities can process particular services more quickly than the others. Besides, the implementation of horizontal offloading capability on end devices and network edges is an extremely complex task due to their greatly varied characteristics [15], [16].

The remainder of this paper is organized as follows. In Section 2, we review existing papers on workload and capacity optimizations for cloud-edge computing, which are closely related to our work. We then develop our architecture of cloud-edge computing and present the problem formulation in Section 3. In Section 4, we elaborate a branch-and-bound algorithm to address the problem. Evaluation study and experimental results are presented in Section 5. Finally, Section 6 concludes this paper.

II. Related work

A variety of research papers [7]–[14] has dealt with the collaborative integration between cloud and edge computing to exploit the advantages of both these technologies. Some of these papers have investigated simple two-tier systems which considered only vertical offloading between local edge nodes and remote cloud servers [7]–[11]. The authors of [12], [13] attempted to provide four-tier architectures of cloud-edge computing, wherein each tier has different capacity, vicinity, and reachability for end-users. Unfortunately, the horizontal offloading between the service nodes in the same tier was not addressed in these architectures.

The reason for these limitations of existing studies, from our understanding, is to reduce the complexity of their optimization models. Besides, the papers seemed to encounter some difficulties, such as the horizontal offload capacity modeling and the loop situations between service nodes, while formulating both vertical and horizontal offloading between service nodes into their optimization models. To address the issues, this paper defines the definitions of parent and sibling nodes of a service node, to which it can do vertical and horizontal offloading, respectively. Also, the node cannot receive the workloads of a service from its siblings if it already horizontally offloads this type of workload in order to prevent loop situations. The consideration of both vertical and horizontal offloading obviously increases the computational complexity of our model, compared to those of existing studies. Thus, we develop an approximation algorithm applying a branch-and-bound approach to obtain solutions for our optimization problem.

Table I summarizes those papers which address workload and cost optimization for cloud-edge computing. It must be noted that in this paper, we use the terms *edge computing* and *fog computing* interchangeably. Generally, the objectives of these cited papers are to minimize system costs or the end-to-end delay of services offered. For example, the paper [8] presented an integration approach to minimize service delay, which utilizes both VM migration and transmission power control. Deng et al. [9] formulated a trade-off between power consumption and delay in a cloud-edge computing system as an MINLP problem. The authors of [10] considered four components, the energy cost of cloud servers, cost of network bandwidth, propagation delay, and the compensation paid for end devices, while formulating a total cost minimization problem for a cloud-edge computing system. Lin et al. [11] introduced a two-phase iterative optimization algorithm which aims to minimize the capacity cost of a three-tier cloud-edge system. Resource allocation problems with the objective of service delay minimization were studied by Souza et al. in [12], [13]. The authors of [14] considered both vertical offloading between edge and cloud nodes and horizontal offloading between neighboring edge nodes in a study on online workload optimization. Unfortunately, the authors only investigated a simple case of single service in this work.

To sum up, our optimization model shares the same objective with related work [9]–[11] which is to minimize computation and communication costs, e.g., C^S and C^N, of the system, while satisfying the end-to-end delay constraints of service requests. Note that some related work [7], [8], [12]–[14] has reversed objective and constraints with our model. In other words, the studies aim to minimize the end-to-end delay D of service requests while ensuring all service allocations.

III. Generic architecture of collaborative cloud-edge computing and optimization model

A. Generic architecture of collaborative cloud-edge computing

Our proposed design of collaborative cloud-edge computing is described in Figure 1. The novel aspect of the design is that it deploys virtualized communication and computation services to four different hierarchical tiers. The first tier of the hierarchy is composed of end devices, such as smartphones,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2937342, IEEE Transactions on Network and Service Management

TABLE I: Comparison of studies on workload and capacity optimization for cloud-edge computing

	# Tiers	# Services	Horizontal offloading	Variables	Objective (Minimize)	Optimization problem	Solution
[8]	2 (Device/Edge)	1	No	RA	D	NLP	Integrated VM migration and TPC
[9]	2 (Device/Cloud)	1	No	WO and CA	C^S	MINLP	Decomposition algorithm
[7]	2 (Edge/Cloud)	Multiple	No	WO	D	MIP	Simulated annealing algorithm
[10]	2 (Device/Cloud)	Multiple	No	WO	$C^S + C^N$	MINLP	Distributed Jacobian algorithm
[11]	3 (Device/Edge/Cloud)	2	No	WO and CA	C^S	NLP	Two-phase iterative optimization
[12]	4 (Device/Edge 1/Edge 2/Cloud)	Multiple	No	RA	D	ILP	Solved by optimization tools
[13]	4 (Device/Edge 1/Edge 2/Cloud)	Multiple	No	Multiple RA	D	Knapsack	Solved by optimization tools
[14]	2 (Device/Cloud)	1	Yes	WO	D	MINLP	Online k-secretary algorithm
Ours	4 (Device/Edge/Office/Cloud)	Multiple	Yes	WO and CA	$C^S + C^N$	MINLP	Branch-and-bound algorithm

(Abbreviation - WO: workload offloading, CA: capacity allocation, RA: resource allocation, D: system delay, C^S: computation cost, C^N: communication cost, MINLP: mixed-integer nonlinear programming, MIP: mixed-integer programming, ILP: integer linear programming, NLP: nonlinear programming, TPC: transmission power control)

IP cameras and IoT sensors, which directly receive service workloads from their sources. In our design, a device can by itself locally process (i.e., carry out local offloading) a fraction of the input workloads or horizontally offload some of the other workloads to neighboring devices, using various short-range wireless transmission techniques such as LTE D2D, Wi-Fi Direct, ZigBee, and Bluetooth. For the remaining workloads, the device needs to vertically offload to network edge nodes (e.g., edge servers, switches, routers, base stations) in the second tier using access network technologies such as Ethernet, Wi-Fi, and 4G/5G. The edge nodes, in turn, can also process a part of received workloads. Further, horizontal and vertical offloading can be carried out by the nodes to dispatch their workloads to nearby edge nodes, and to central offices in the third tier, by leveraging core network technologies. Similar to devices and edge nodes, a central office, which is re-structured as a small-size data center, can also carry out local processing, and horizontal offloading to neighboring central offices and vertical offloading to a remote federated data center in the fourth tier. In our design, the central offices are connected to each other and to the data center using backbone network technologies. The data center, which is located in the top-most tier of the cloud-edge computing hierarchy, acts as the backstop to the whole system. In other words, it is responsible for processing the remaining workloads which cannot be handled by lower tiers.

Our aim is to develop a generic architecture which can be used to deploy different types of services. Taking a real-time vehicle congestion avoidance service in smart cities as an example, IP cameras, which are used for traffic monitoring, can provide instant alarms of emergency events such as car accidents by detecting abnormal traffic behavior. Note that only a fraction of the volume of the captured data is dispatched to an edge server for detailed analysis. The server then processes the data to obtain more refined information which can be sent to automobile drivers or conveyed to news outlets across the whole city. If there is a lack of computation capacity, the server can send the data to nearby edge servers for execution. In cases of serious situations, some data can also be redirected to central offices or even to a remote data center for running traffic re-routing algorithms which need very high computation capacity. Note that the proposed architecture is generic which includes four different hierarchical tiers, such as end device, network edge, central office, and data center which provide both vertical and horizontal offloading between service nodes. In real deployments, the design can be customized and adjusted by, for example, merging the central office and the data center into a cloud tier or removing the horizontal offloading capability between end devices. By doing so, the design becomes specific architectures, such as Edge server, Coordinate device and Device cloud [16] which can accommodate different types of cloud-edge services and applications.

In the following subsection, we provide an optimization model for the proposed cloud-edge computing architecture. We then formulate a cost minimization problem for the model whose important notations are summarized in Table II.

B. Optimization model

1) Workload model: Let $f \in \mathbb{F}$ denote an offered service of a cloud-edge computing system. Each service f has a computation size Z_f^S which is the number of mega CPU cycles required to process a request for service f. Also, communication size Z_f^N indicates the data size of the request in megabytes. Let $I^\alpha, I^\beta, I^\gamma$, and I^δ be the sets of devices, network edges, central offices and data centers of the system, respectively. A service node $i \in I$ could process a set of services $F_i \subseteq \mathbb{F}$, where I is the set of all service nodes of the system, i.e., $I = I^\alpha \cup I^\beta \cup I^\gamma \cup I^\delta$.

a) Local processing: Let p_i^f denote the workload (in requests per second) of a service f which is locally processed

TABLE II: Summary of important notations

Notation	Meaning	Attribute
Service		
$f, \mathbb{F}$	index and set of system services	Input
Z_f^S, Z_f^N	computation and communication size of service f	Input
Service node - Capacity		
i, I	index and set of all service nodes	Input
F_i	set of services supported by a node i	Input
$I^\alpha, I^\beta, I^\gamma, I^\delta$	sets of devices, network edge nodes, central offices and data centers	Input
H_i, V_i, K_i	sets of sibling, parent, child nodes of a node i	Input
μ_i^S	computation capacity of a node $i \in I^\alpha \cup I^\beta$	Variable
n_i, ν_i^S	number of active servers and computation capacity of a server of a node $i \in I^\gamma \cup I^\delta$	Variable
$\mu_{i,j}^N$	communication capacity of network connection from i to j	Variable
Workload		
λ_i^f	input workload of service f to a device $i \in I^\alpha$	Input
p_i^f	workload of service f locally processed by a node i	Variable
$x_{i,j}^f$	horizontal offloading workload of service f from i to j, where $j \in H_i$	Variable
$u_{j,i}^f$	horizontal offloading workload of service f from j to i, where $j \in H_i$	Variable
$y_{i,j}^f$	vertical offloading workload of service f from i to j, where $j \in V_i$	Variable
$v_{i,j}^f$	vertical offloading workload of service f from j to i, where $j \in K_i$	Variable
Delay		
D_i^S	computation delay of a node i	Output
$D_{i,j}^N$	communication delay of a network connection from i to j	Output
$D_\alpha, D_\beta, D_\gamma, D_\delta$	delay of device tier, edge tier, central office tier, and data center tier	Output
$\overline{D}_\alpha, \overline{D}_\beta, \overline{D}_\gamma, \overline{D}_\delta$	delay thresholds of device tier, edge tier, central office tier, and data center tier	Input
Cost		
$W_\alpha^S, W_\beta^S, W_\gamma^S, W_\delta^S$	computation cost of a device, edge node, central office, and data center	Input
$W_{\alpha,\alpha}^N, W_{\alpha,\beta}^N, W_{\beta,\beta}^N, W_{\beta,\gamma}^N, W_{\gamma,\gamma}^N, W_{\gamma,\delta}^N$	communication cost of a device-to-device, a device-to-edge, edge-to-edge, edge-to-office, office-to-office, and office-to-center connection	Input
$C^S, C^N, \mathbb{C}$	computation cost, communication cost, and total cost of the system	Output

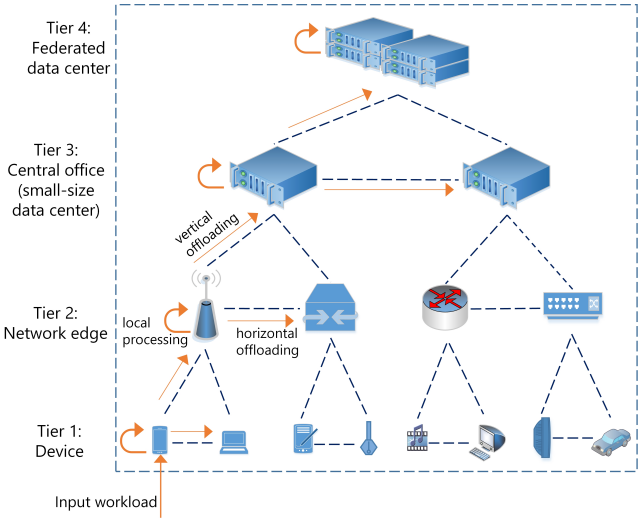


Fig. 1: The generic architecture of collaborative cloud-edge computing

by a node i. We have

$$p_i^f = \begin{cases} \geq 0, & \text{if } f \in F_i, \forall i \in I, \\ = 0, & \text{if } f \notin F_i, \forall i \in I. \end{cases} \quad (1)$$

b) Sibling node and horizontal offloading: The set of siblings H_i of a node $i \in I$ consists of service nodes which are located in the same tier as i, and to which i can horizontally offload its workloads. Also, let $x_{i,j}^f$ be the workload of a service f which is horizontally offloaded from i to a service node $j \in H_i$. Similarly, let $u_{j,i}^f$ be the workload of a service f which is horizontally offloaded from $j \in H_i$ to i. Here, we assume that a service node i can offload the workload of a service f to a sibling node j on condition that j is able to process f, i.e., $f \in F_j$. In addition, to prevent loop situations, a node cannot receive the workloads of a service f from its siblings if it already horizontally offloads this type of workload. Thus, we have

$$x_{i,j}^f = \begin{cases} \geq 0, & \text{if } f \in F_j, \forall j \in H_i, \forall i \in I, \\ = 0, & \text{if } f \notin F_j, \forall j \in H_i, \forall i \in I, \end{cases} \quad (2a)$$

$$u_{j,i}^f = \begin{cases} \geq 0, & \text{if } f \in F_i, \forall j \in H_i, \forall i \in I, \\ = 0, & \text{if } f \notin F_i, \forall j \in H_i, \forall i \in I, \end{cases} \quad (2b)$$

$$x_{i,j}^f * \sum_{j \in H_i} u_{j,i}^f = 0, \ \forall f \in \mathbb{F}, \forall i \in I. \quad (2c)$$

c) Parent/child node and vertical offloading: The set of parents V_i of a service node $i \in I$ consists of the nodes located in the next tier up with i, and to which i can vertically offload its workloads. Let $y_{i,j}^f$ be the workload of a service f which is vertically offloaded from i to a node $j \in V_i$. The set of children K_i of i consists of the nodes which are located in the right lower-tier with i, and from which i receives incoming workloads. Let $v_{j,i}^f$ denote the workload of a service f which is vertically offloaded from $j \in K_i$ to i. Since a device $i \in I^\alpha$ directly receives service workloads from external sources, it

has no child nodes, i.e., $K_i = \emptyset,\ \forall i \in I^\alpha$. Similarly, a data center $i \in I^\delta$ is in the most-top tier of the system, and hence has no parent nodes, i.e., $V_i = \emptyset,\ \forall i \in I^\delta$. Opposed to horizontal offloading, a service node can carry out vertical offloading for all services $f \in F$. In other words, it can dispatch all types of workloads to its parents. Thus, we have

$$y^f_{i,j} \geq 0, \quad \forall f \in \mathbb{F}, \forall j \in V_i, \forall i \in I^\alpha \cup I^\beta \cup I^\gamma, \tag{3a}$$

$$v^f_{j,i} \geq 0, \quad \forall f \in \mathbb{F}, \forall j \in K_i, \forall i \in I^\beta \cup I^\gamma \cup I^\delta. \tag{3b}$$

Let λ^f_i denote the submitted workload of a service f from external sources to a device $i \in I^\alpha$. We have

$$\lambda^f_i \geq 0, \quad \forall f \in \mathbb{F}, \forall i \in I^\alpha. \tag{4}$$

Hence, the workload balanced constraints of the system are defined as

$$\begin{cases} \sum\limits_{j \in H_i} u^f_{j,i} + \lambda^f_i = \sum\limits_{j \in H_i} x^f_{i,j} + \sum\limits_{j \in V_i} y^f_{i,j} + p^f_i & \text{if } i \in I^\alpha, \\ \sum\limits_{j \in H_i} u^f_{j,i} + \sum\limits_{j \in K_i} v^f_{j,i} = \sum\limits_{j \in H_i} x^f_{i,j} + \sum\limits_{j \in V_i} y^f_{i,j} + p^f_i & \text{if } i \in I^\beta \cup I^\gamma \cup I^\delta. \end{cases} \tag{5}$$

2) Computation and Communication delay:

a) Computation delay of device and edge nodes: In the case of a service node i is a device or a network edge, i.e., $i \in I^\alpha \cup I^\beta$, where there is only a single server in the node which is responsible for processing incoming workloads. Hence, the M/M/1 queuing model [17, Ch. 2] is employed to compute the node's computation delay D^S_i as

$$D^S_i = \frac{1}{\mu^S_i - \sum\limits_{f \in F_i} p^f_i * Z^S_f}, \tag{6}$$

$$\sum_{f \in F_i} p^f_i * Z^S_f < \mu^S_i \leq \mu^{S,MAX}_i, \forall i \in I^\alpha \cup I^\beta, \tag{7}$$

where μ^S_i is the computation capacity (in mega CPU cycles per second), i.e., service rate, of the node i; and $\sum\limits_{f \in F_i} p^f_i * Z^S_f$ is the total CPU cycles per second demanded for executing all workloads $p^f_i, \forall f \in F_i$, which have to be locally processed by i. Note that μ^S_i cannot exceed the maximum value $\mu^{S,MAX}_i$.

b) Computation delay of central office and data center nodes: In the case where a service node i is a central office or a data center, i.e., $i \in I^\gamma \cup I^\delta$, there are multiple parallel servers in the node which can process incoming workloads. Therefore, we apply the M/M/n queuing model in [10] to compute the node's computation delay D^S_i as

$$D^S_i = \frac{1}{n_i * \nu^S_i - \sum\limits_{f \in F_i} p^f_i * Z^S_f} + \frac{1}{\nu^S_i}, \tag{8}$$

$$n_i * \nu^S_i > \sum_{f \in F_i} p^f_i * Z^S_f, \forall i \in I^\gamma \cup I^\delta, \tag{9a}$$

$$\nu^S_i \leq \nu^{S,MAX}_i, \forall i \in I^\gamma \cup I^\delta, \tag{9b}$$

$$n_i \in \mathbb{N}, \tag{9c}$$

$$n_i \leq n^{MAX}_i, \forall i \in I^\gamma, \tag{9d}$$

where n_i is the number of active servers of the node i, and which cannot exceed the maximum value $n^{MAX}_i, \forall i \in I^\gamma$; and ν^S_i is the computation capacity of the servers which cannot exceed the maximum value $\nu^{S,MAX}_i$. Note that we assume that the servers are identical and there is not constraint of an upper limit for the number of active servers ν^S_i of a data center $i \in I^\delta$.

c) Communication delay of network connections: Let (i, j) be a network connection from a service node i to another node j, where $j \in H_i \cup V_i$. Each connection (i, j) has a communication capacity $\mu^N_{i,j}$, i.e., network bandwidth, which is measured in megabytes per second. The total amount of data transmitted through (i, j) per second is $\sum\limits_{f \in F_i} x^f_{i,j} * Z^N_f$ or $\sum\limits_{f \in \mathbb{F}} y^f_{i,j} * Z^N_f$ .Thus, its communication delay $D^N_{i,j}$ is calculated by the M/M/1 queuing model as

$$D^N_{i,j} = \begin{cases} \frac{1}{\mu^N_{i,j} - \sum\limits_{f \in F_i} x^f_{i,j} * Z^N_f} + \frac{Ni,j}{LS} & \text{if } j \in H_i, \\ \frac{1}{\mu^N_{i,j} - \sum\limits_{f \in \mathbb{F}} y^f_{i,j} * Z^N_f} + \frac{Ni,j}{LS} & \text{if } j \in V_i, \end{cases} \tag{10}$$

$$\sum_{f \in F_i} x^f_{i,j} * Z^N_f < \mu^N_{i,j}, \forall j \in H_i, \forall i \in I^\alpha \cup I^\beta \cup I^\gamma, \tag{11a}$$

$$\sum_{f \in \mathbb{F}} y^f_{i,j} * Z^N_f < \mu^N_{i,j}, \forall j \in V_i, \forall i \in I^\alpha \cup I^\beta \cup I^\gamma, \tag{11b}$$

where $\frac{N_{i,j}}{LS}$ is the propagation delay of a network connection (i, j); and where $N_{i,j}$ is the distance between i and j, and LS is the speed of light, i.e., approximately $3 * 10^8 m/s$.

d) Computation and communication delay of the cloud-edge computing system: The delay of a tier in our cloud-edge computing architecture which consists of its computation and communication delay is defined as

$$D_\alpha = \frac{1}{|I^\alpha|} \sum_{i \in I^\alpha} D^S_i + \frac{1}{|H_i|} \sum_{i \in I^\alpha} \sum_{j \in H_i} D^N_{i,j}, \tag{12a}$$

$$D_\beta = \frac{1}{|I^\beta|} \sum_{i \in I^\beta} D^S_i + \frac{1}{|H_i|} \sum_{i \in I^\beta} \sum_{j \in H_i} D^N_{i,j} + \frac{1}{|V_i|} \sum_{i \in I^\alpha} \sum_{j \in V_i} D^N_{i,j} + \frac{1}{|H_i|} \sum_{i \in I^\alpha} \sum_{j \in H_i} D^N_{i,j}, \tag{12b}$$

$$D_\gamma = \frac{1}{|I^\gamma|} \sum_{i \in I^\gamma} D^S_i + \frac{1}{|H_i|} \sum_{i \in I^\gamma} \sum_{j \in H_i} D^N_{i,j} + \frac{1}{|V_i|} \sum_{i \in I^\beta} \sum_{j \in V_i} D^N_{i,j} + \frac{1}{|H_i|} \sum_{i \in I^\beta} \sum_{j \in H_i} D^N_{i,j} + \frac{1}{|V_i|} \sum_{i \in I^\alpha} \sum_{j \in V_i} D^N_{i,j} + \frac{1}{|H_i|} \sum_{i \in I^\alpha} \sum_{j \in H_i} D^N_{i,j}, \tag{12c}$$

$$D_\delta = \frac{1}{|I^\delta|} \sum_{i \in I^\delta} D^S_i + D^N_{i,j} + \frac{1}{|V_i|} \sum_{i \in I^\gamma} \sum_{j \in V_i} + \frac{1}{|H_i|} \sum_{i \in I^\gamma} \sum_{j \in H_i} D^N_{i,j} + \frac{1}{|V_i|} \sum_{i \in I^\beta} \sum_{j \in V_i} D^N_{i,j} + \frac{1}{|H_i|} \sum_{i \in I^\beta} \sum_{j \in H_i} D^N_{i,j} + \frac{1}{|V_i|} \sum_{i \in I^\alpha} \sum_{j \in V_i} D^N_{i,j} + \frac{1}{|H_i|} \sum_{i \in I^\alpha} \sum_{j \in H_i} D^N_{i,j}, \tag{12d}$$

where $D_\alpha, D_\beta, D_\gamma$, and D_δ are the delay of device, edge, central office, and data center tier, respectively. Since a cloud-

edge computing system has to satisfy its delay constraints, we then have

$$D_\alpha \le \overline{D}_\alpha, \tag{13a}$$
$$D_\beta \le \overline{D}_\beta, \tag{13b}$$
$$D_\gamma \le \overline{D}_\gamma, \tag{13c}$$
$$D_\delta \le \overline{D}_\delta, \tag{13d}$$

where $\overline{D}_\alpha, \overline{D}_\beta, \overline{D}_\gamma$, and $\overline{D}_\delta$ are the delay thresholds of device, edge, central office, and data center tier, respectively.

3) Total Cost Minimization Problem: Here, we focus on minimizing the total cost of a cloud-edge computing system which consists of the computation cost of service nodes and the communication cost of network connections.

a) Computation cost of service nodes: Let $W^S_\alpha, W^S_\beta, W^S_\gamma$, and W^S_δ be the computation capacity cost (in money units/mega CPU cycles per second) of a device, an edge node, a central office, and a data center, respectively. Thus, the computation cost of the system is

$$C^S = W^S_\alpha * \sum_{i \in I^\alpha} \mu^S_i + W^S_\beta * \sum_{i \in I^\beta} \mu^S_i + W^S_\gamma * \sum_{i \in I^\gamma} n_i * \nu^S_i + W^S_\delta * \sum_{i \in I^\delta} n_i * \nu^S_i. \tag{14}$$

b) Communication cost of network connections: Let $W^N_{\alpha,\alpha}, W^N_{\alpha,\beta}, W^N_{\beta,\beta}, W^N_{\beta,\gamma}, W^N_{\gamma,\gamma}$, and $W^N_{\gamma,\delta}$ denote the communication capacity cost (in money units/MBps) of a device-to-device, a device-to-edge, an edge-to-edge, an edge-to-office, an office-to-office, and an office-to-center connection, respectively. Thus, the communication cost of the system is

$$C^N = C^N_\alpha + C^N_\beta + C^N_\gamma, \tag{15}$$

where

$$C^N_\alpha = \sum_{i \in I^\alpha} \sum_{j \in V_i} W^N_{\alpha,\alpha} * \mu^N_{i,j} + \sum_{i \in I^\alpha} \sum_{j \in H_i} W^N_{\alpha,\beta} * \mu^N_{i,j}, \tag{16a}$$

$$C^N_\beta = \sum_{i \in I^\beta} \sum_{j \in V_i} W^N_{\beta,\beta} * \mu^N_{i,j} + \sum_{i \in I^\beta} \sum_{j \in H_i} W^N_{\beta,\gamma} * \mu^N_{i,j}, \tag{16b}$$

$$C^N_\gamma = \sum_{i \in I^\gamma} \sum_{j \in V_i} W^N_{\gamma,\gamma} * \mu^N_{i,j} + \sum_{i \in I^\gamma} \sum_{j \in H_i} W^N_{\gamma,\delta} * \mu^N_{i,j}. \tag{16c}$$

c) System total cost: The total system cost $\mathbb{C}$ of a cloud-edge computing is defined as

$$\mathbb{C} = C^S + C^N. \tag{17}$$

Since we aim to minimize the total cost of the cloud-edge computing system while guaranteeing its delay constraints, we hence have an optimization problem which is defined as

$$(\mathcal{P}) \quad \min_{p^f_i, x^f_{i,j}, y^f_{i,j}, \mu^S_i, n_i, \nu^S_i, \mu^N_{i,j}} \mathbb{C}, \tag{18a}$$

$$s.t. \quad (1) - (5), (7), (9), (11), (13). \tag{18b}$$

IV. Algorithm design - Branch-and-bound with parallel multi-start search points

As can be seen, the problem $\mathcal{P}$ has integer variables n_i and nonlinear delay constraint functions. Thus, $\mathcal{P}$ is an MINLP problem, which is generally very difficult to solve. We thus apply the *Branch-and-bound algorithm* [18] to address the problem. The general idea is to relax $\mathcal{P}$ to a tree whose nodes are nonlinear programming (NLP) subproblems, which are easier to tackle, by removing the integrality conditions of $\mathcal{P}$, i.e., variables n_i can be continuous. Each subproblem node in the tree is iteratively selected for solving using a depth-first search strategy. When a feasible solution with variables n_i as integers is found, it provides an upper bound solution for the original problem $\mathcal{P}$. Other nodes that exceed this bound are pruned, and the enumeration is conducted until all leaf nodes of the tree are resolved or the search conditions are reached.

Algorithm 1 shows the pseudo-code of our proposed algorithm, in which $\mathbb{C}(\mathcal{N}, \mathcal{O})$ denotes the current optimal solution for $\mathcal{P}$, where $\mathcal{N}$ and $\mathcal{O}$ are the sets of determined integers and continuous variables of the solution, respectively. In the algorithm, we first attempt to find an initial solution by applying a *Feasibility Pump* [19] relaxation heuristic (lines 2-5), before starting the branch-and-bound procedure. If a feasible solution $\mathbb{C}^*(\mathcal{N}^*, \mathcal{O}^*)$ is reached, it becomes the current optimal solution $\mathbb{C}(\mathcal{N}, \mathcal{O})$ (line 4). After that, an NLP sub-problem $\mathcal{SP}$, which is generated by removing the integrality conditions of variables n_i of the problem $\mathcal{P}$, is added to the tree data structure $\mathcal{T}$ (lines 6-7). Next, the branch-and-bound procedure starts to iteratively solve the sub-problem $\mathcal{SP} \in \mathcal{T}$ by the well-known Interior/Direct algorithm [20] with parallel multiple initial searching points. Then we have four possibilities. If a feasible solution $\mathbb{C}^*(\mathcal{N}^*, \mathcal{O}^*)$ which is smaller than the current optimal solution $\mathbb{C}(\mathcal{N}, \mathcal{O})$ is obtained and $\mathcal{N}^*$ are integers, it becomes the current optimal solution and the node $\mathcal{SP}$ is pruned, i.e., we remove $\mathcal{SP}$ and its sub-nodes from $\mathcal{T}$ (line 13-14). If $\mathcal{N}^*$ is not an integer, a branching operation is performed on a variable $n_i \in N^*$. In other words, two new sub-problems $\mathcal{SSP}1$ and $\mathcal{SPP}2$ of $\mathcal{SP}$ are created and added into $\mathcal{T}$ using the *Pseudo-cost branching* method [19] (line 16). In cases where $\mathbb{C}^*(\mathcal{N}^*, \mathcal{O}^*)$ is equal to or greater than $\mathbb{C}(\mathcal{N}, \mathcal{O})$, or there is no a feasible solution, the node $\mathcal{SP}$ is also pruned. The branch-and-bound procedure is repeated until all nodes of $\mathcal{T}$ have been resolved (line 8).

Note that we solve an NLP problem $\mathcal{SP} \in \mathcal{T}$ using the Interior/Direct algorithm which can just yield a local optimal solution $\mathbb{C}^*(\mathcal{N}^*, \mathcal{O}^*)$. To address this issue, we apply the parallel multiple-start point method which attempts to obtain a global optimal solution by running the Interior/Direct algorithm with different initial values of the variables of the problem $\mathcal{SP}$. In this paper, the values are randomly generated, which satisfies the lower and upper bound constraints of the variables.

V. Numerical results

In this section, we present the numerical results which were obtained from simulation experiments carried out to investigate the performance of our proposed cloud-edge computing architecture.

A. Experiment parameter settings

For simplicity, but without the loss of generality, we carried out simulation experiments using a tree topology of a cloud-edge computing system whose number of service nodes and

Algorithm 1: Branch-and-bound with parallel multi-start search points

$\mathbb{C}(\mathcal{N}, \mathcal{O}) \leftarrow \infty$, $\mathcal{T} \leftarrow \emptyset$;
/* Attemp to find an initial solution */
Solve $\mathcal{P}$ by *Feasibility Pump* heuristic;
if *find a feasible solution* $\mathbb{C}^*(\mathcal{N}^*, \mathcal{O}^*)$ **then**
  $\mathbb{C}(\mathcal{N}, \mathcal{O}) \leftarrow \mathbb{C}^*(\mathcal{N}^*, \mathcal{O}^*)$;
end
/* Begin branch-and-bound procedure */
$\mathcal{SP} \leftarrow$ Relax integrality constraints of $\mathcal{P}$;
$Add(\mathcal{T}, \mathcal{SP})$;
while $\exists \mathcal{SP} \in \mathcal{T}$ *has not been reached or pruned* **do**
  Select a subproblem $\mathcal{SP} \in \mathcal{T}$ by depth-first strategy;
  Solve $\mathcal{SP}$ by *Parallel Multi-start Interior/Direct* algorithm;
  if *find a feasible solution* $\mathbb{C}^*(\mathcal{N}^*, \mathcal{O}^*) < \mathbb{C}(\mathcal{N}, \mathcal{O})$ **then**
    if $\mathcal{N}^* \in \mathbb{N}$ **then**
      $\mathbb{C}(\mathcal{N}, \mathcal{O}) \leftarrow \mathbb{C}^*(\mathcal{N}^*, \mathcal{O}^*)$;
      $Prune(\mathcal{T}, \mathcal{SP})$;
    else
      Create 2 subproblem nodes $\mathcal{SSP}1$, $\mathcal{SPP}2$ of $\mathcal{SP}$ by *Pseudo-cost branching* on a $n_i \in \mathcal{N}^*$;
    end
  else
    $Prune(\mathcal{T}, \mathcal{SP})$;
  end
end

their network distances are summarized in Table III. In the topology, a service node has only one parent node, and the nodes with the same parent can carry out horizontal offloading to each other. Note that the topology can be extended to more service nodes with similar results. Table IV shows the important parameters applied in our experiments, refering [9] and [10]. All optimization problems were modeled by AMPL [21], [22], and the algorithm 1 was implemented using Knitro [23] optimization solvers.

In these experiments, we compared our cloud-edge computing architecture design (WH) which supports horizontal offloading to a traditional design (NH) which does not support horizontal offloading between service nodes. The optimization model of NH is similar to the problem $\mathcal{P}$, but its service nodes have no sibling nodes, i.e., $H_i = \emptyset$, $\forall i \in I$. Consequently, the nodes can only carry out local and vertical workload offloading. It is appropriate because NH shares common characteristics with related work [7]–[14]. Two designs WH and NH were evaluated under *light, medium*, and *heavy* workload cases in which the arrival rate λ_i^f was adjusted to generate submitted service workloads whose total demanded computation capacity was about 10%, 50%, and 100%, respectively, of the maximum capacity of all service nodes in device, edge, and central office tiers.

Note that the objective of our optimization model is to minimize the total system cost $\mathbb{C}$ which consists of the computation cost of service nodes and the communication cost

TABLE III: Service nodes and their distances in experiments

Service node	Value	Distance	Value
# devices per edge	4	device-to-device	0.1 Km
# edges per central office	3	device-to-edge	1 Km
# central offices	3	edge-to-edge	1 Km
# data center	1	edge-to-central office	10 Km
		office-to-office	100 Km
		office-to-data center	1000 Km

of network connections. It obviously is the main performance metric of our experiments. We also present the results of other metrics such as computation capacity allocation, workload allocation, and horizontal offloading workloads to explain the phenomenon observed from the experiments.

B. Analysis of results

The simulation results presented in this section evaluate the effectiveness of WH and NH designs under three different operational scenarios. We first investigate their performance in the unbalanced and balanced input workload scenarios. Then, two service allocation strategies, i.e., homogeneous and heterogeneous, are tested. Finally, the impact of different situations of computation capacity costs on WH and NH designs is observed.

1) Unbalanced vs. Balanced workload scenario: We observed the performance of WH and NH designs under *unbalanced* and *balanced* workload scenarios. As can be seen in Figure 2, in an unbalanced scenario, the input workloads λ_i^f of a device $i \in I^\alpha$ were randomly generated according to an exponential distribution with different mean values $\overline{\lambda}_i^f$ calculated using Equations 19 and 20 whereas the workloads were exponentially distributed with the same value of $\overline{\lambda}_i^f$ in a balanced scenario. In the unbalanced scenario, $\overline{\lambda}_0^f$ and Δ_0 were set to 0.1 and 2.0 for all workload cases, and ϕ was set to 0.5, 1.0, and 1.5 for light, medium, and heavy workload cases. In the balanced workload scenario, $\overline{\lambda}_i^f$ was set to 2.5, 10.0 and 25.0 for these cases.

$$\overline{\lambda}_i^f = \begin{cases} \overline{\lambda}_0^f, & \text{if } \ i = 0, \\ \overline{\lambda}_{i-1}^f * \Delta_j, & \text{otherwise}, \forall i \in K_j, \forall j \in I^\beta, \end{cases} \tag{19}$$

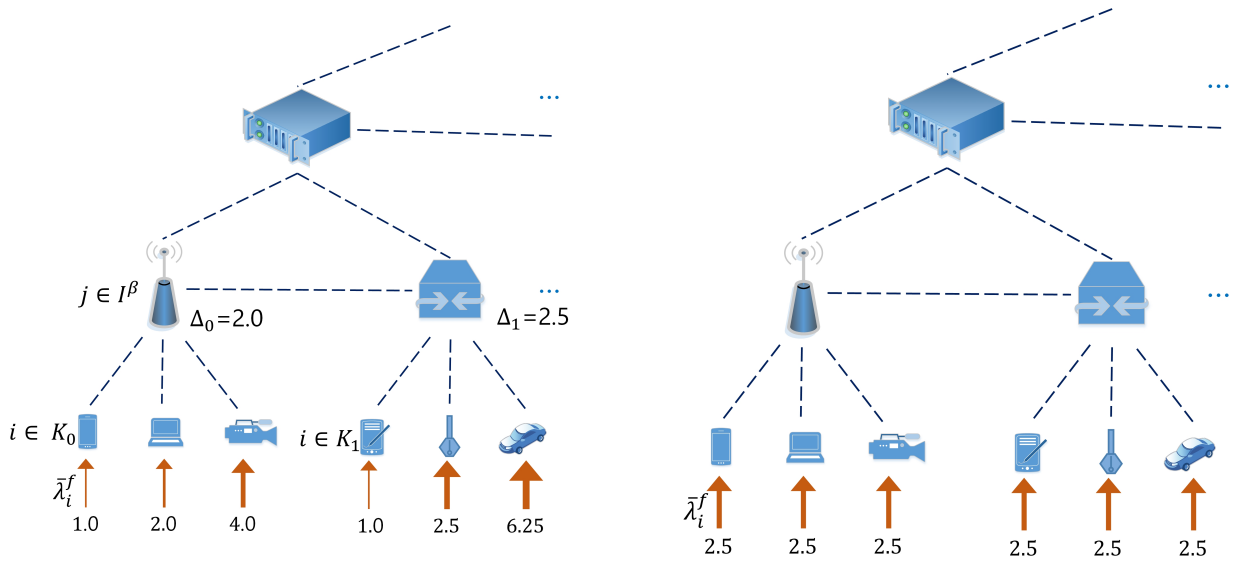
where

$$\Delta_j = \begin{cases} \Delta_0, & \text{if } \ j = 0, \\ \Delta_{j-1} + \phi, & \text{otherwise}, \forall j \in I^\beta. \end{cases} \tag{20}$$

As can be seen in Figure 3a, the design with horizontal offloading WH can significantly reduce the total system cost in an unbalanced workload scenario, compared to the NH design which has no horizontal offloading ability. In the case of a light workload, for example, the cost was decreased approximately 4.4 (i.e., from 15.9 to 11.5) ten thousands of a money unit by the WH design, which means about a 28% decrease. The cost difference was 12.6 and 27.0 ten thousands of money unit, which means about 15% and 10% lower in medium and heavy workload cases. The improvement is explained in Figure 3b, 3c, and 3d. Compared to the NH design, the WH approach can process more workloads in the service nodes of lower tiers instead of vertical offloading to those in higher tiers,

TABLE IV: Experiment parameters

Notation	Meaning	Value	Unit
$\|F\|$	# offered services	4	n/a
Z_f^S	computation size of services	[1 100 1 100]	Mcycles/request
Z_f^N	communication size of services	[1 5 1 5]	MBytes/request
$\mu_i^{S,MAX}, \nu_i^{S,MAX}$	maximum computation capacity	[1000 3000]	Mcycles
n_i^{MAX}	maximum # servers	10	n/a
$W_\alpha^S, W_\beta^S, W_\gamma^S, W_\delta^S$	computation cost	[5.0 10.0 15.0 20.0]	money units/Mcycles
$W_{\alpha,\alpha}^N, W_{\alpha,\beta}^N, W_{\beta,\beta}^N, W_{\beta,\gamma}^N, W_{\gamma,\gamma}^N, W_{\gamma,\delta}^N$	communication cost	[1.0 2.0 3.0 4.0 5.0 6.0]	money units/MBytes
$\overline{D_\alpha}, \overline{D_\beta}, \overline{D_\gamma}, \overline{D_\delta}$	delay constraints	[1.0 1.0 1.0 5.0]	ms
λ_i^f	arrival rate of service workloads	$exponential(\overline{\lambda_i^f})$	requests/ms



(a) Unbalanced scenario ($\overline{\lambda_0^f} = 1.0, \Delta_0 = 2.0, \phi = 0.5$)

(b) Balanced scenario ($\overline{\lambda_i^f} = 2.5$)

Fig. 2: Unbalanced and balanced input workload scenarios

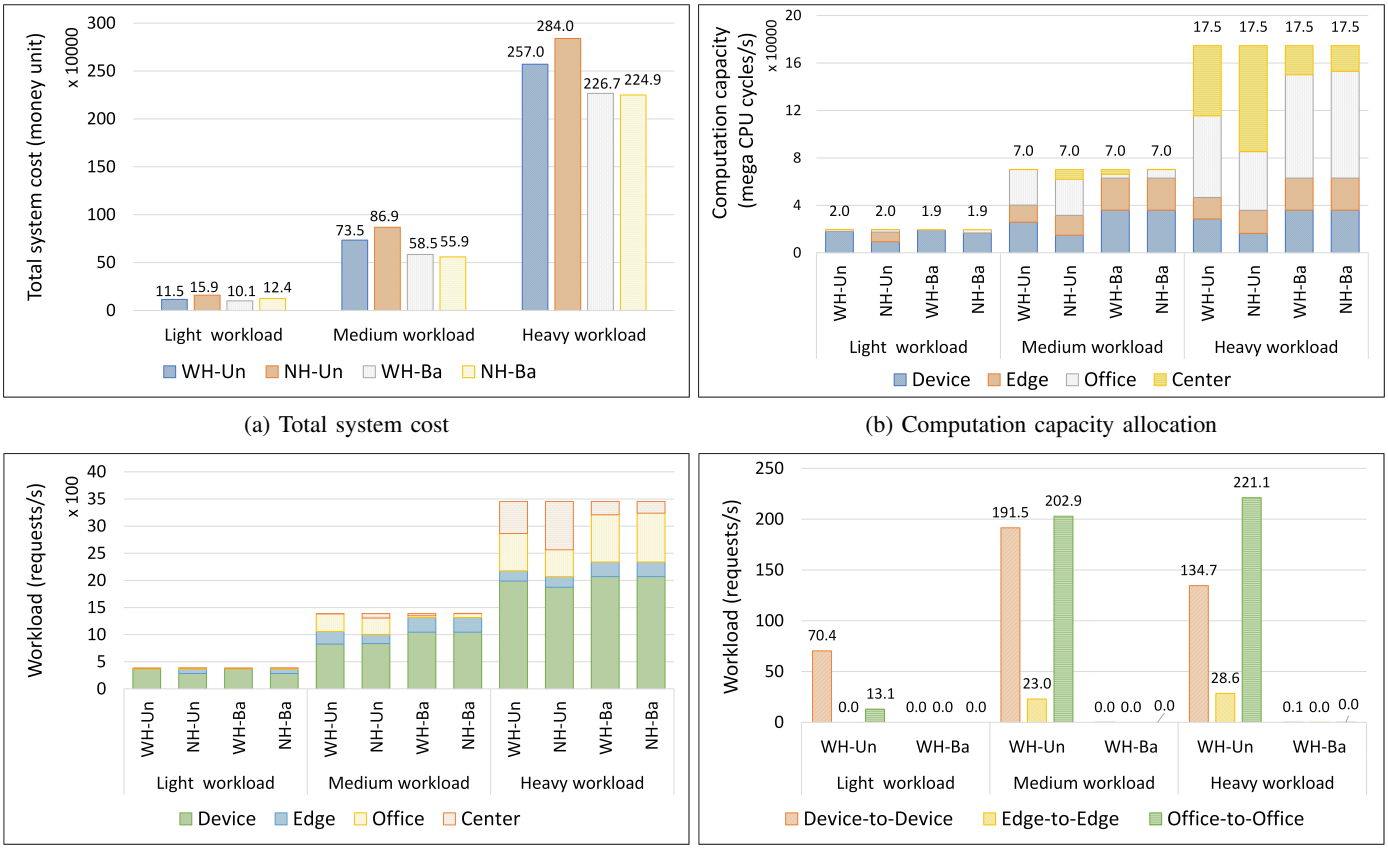
by carrying out horizontal offloading to utilize lower tiers' nodes. In other words, the WH approach can allocate more computation capacity in the service nodes of lower tiers to process the workloads, and since the costs of the computation capacity of lower tiers are lower than those of higher tiers, the total system cost was significantly reduced using this approach.

Figure 3a also clearly shows that in a balanced workload scenario, the total system cost of the WH and NH designs was roughly the same in every workload case. The numerical results in Figures 3b and 3c also illustrate that the computation capacity and workload allocation were approximately equal in these two approaches. The reason why the WH design did not produce much improvement is that its service nodes received the same input workloads. As a result, horizontal offloading did not contribute to a decrease in the total system cost by utilizing the service nodes in lower tiers; it might increase the cost by adding extra communication cost. The nodes either had to process by themselves or vertically offload their received workloads to parents, and had almost no horizontal offloading operations (see Figure 3d). In other words, the WH and NH designs behaved similarly in a balanced workload scenario which led to their performance being roughly the same.

2) Service allocation - Homogeneous vs. Heterogeneous: We conducted a further experiment to investigate the performance of the WH and NH designs in homogeneous and heterogeneous service allocation scenarios. In a homogeneous scenario, all service nodes $i \in I$ have the same capability which can process all services $f \in \mathbb{F}$, i.e., $F_i = \mathbb{F}, \forall i \in I$. On the other hand, in a heterogeneous scenario, their capability F_i can differ from each other. Note that this experiment was conducted with unbalanced input workloads.

As anticipated, Figure 4a shows that a heterogeneous service allocation scenario required higher system cost in both WH and NH designs, compared to a homogeneous scenario. For example, in the case of a light workload, the cost increased by about 12% and 23% (i.e., from 11.5 to 12.9, and from 15.9 to 19.6 of ten thousands of money unit) in WH and NH designs. This phenomenon can be explained by the fact that in a heterogeneous scenario, a service node might receive the workload of a service which it cannot accommodate. The workloads were then horizontally or vertically offloaded which resulted in an increase in total system cost. As can be seen in Figure 4c, in the homogeneous scenario, more workloads were vertically offloaded from lower to higher tiers for processing. Consequently, the two designs had to allocate

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2937342, IEEE Transactions on Network and Service Management



(a) Total system cost

(b) Computation capacity allocation

(c) Workload allocation

(d) Horizontal offloading

Fig. 3: Performance of WH and NH designs in unbalanced and balanced workload scenarios.(WH-Un: With horizontal offloading in unbalanced scenario, NH-Un: Without horizontal offloading in unbalanced scenario, WH-Ba: With horizontal offloading in balanced scenario, NH-Ba: Without horizontal offloading in balanced scenario)

more computation capacity to higher ties this scenario (see Figure 4b).

Figure 4a also shows that in a heterogeneous scenario, compared to the NH design, the WH design decreased the system cost more in light and medium workload cases. The cost was decreased by about 34% (i.e., from 19.6 to 12.9) and 18% (i.e., from 97.2 to 80.1), in these two cases. The corresponding numbers for a homogeneous scenario were 28% (i.e., from 15.9 to 11.5) and 15% (i.e., from 86.9 to 73.5). However, in the case of heavy workload, the cost improvement for the heterogeneous scenario was only about 5%, instead of 10% as for the homogeneous scenario.

3) Impact of computation capacity costs: In this experiment, the performance of the WH and NH designs was investigated in decreased and equal computation capacity cost situations. In other words, the costs of computation capacity of lower tiers were more expensive than those of higher tier in a decreased cost situation. On the other hand, the costs were the same in every tier in an equal-cost situation. The costs $W_\alpha^S, W_\beta^S, W_\gamma^S, W_\delta^S$ were set to [20.0 15.0 10.0 5.0] and [12.5 12.5 12.5 12.5] in money units, in the decreased and equal computation capacity cost situations, respectively. Note that the homogeneous service allocation and unbalanced input workload scenarios were used for this experiment.

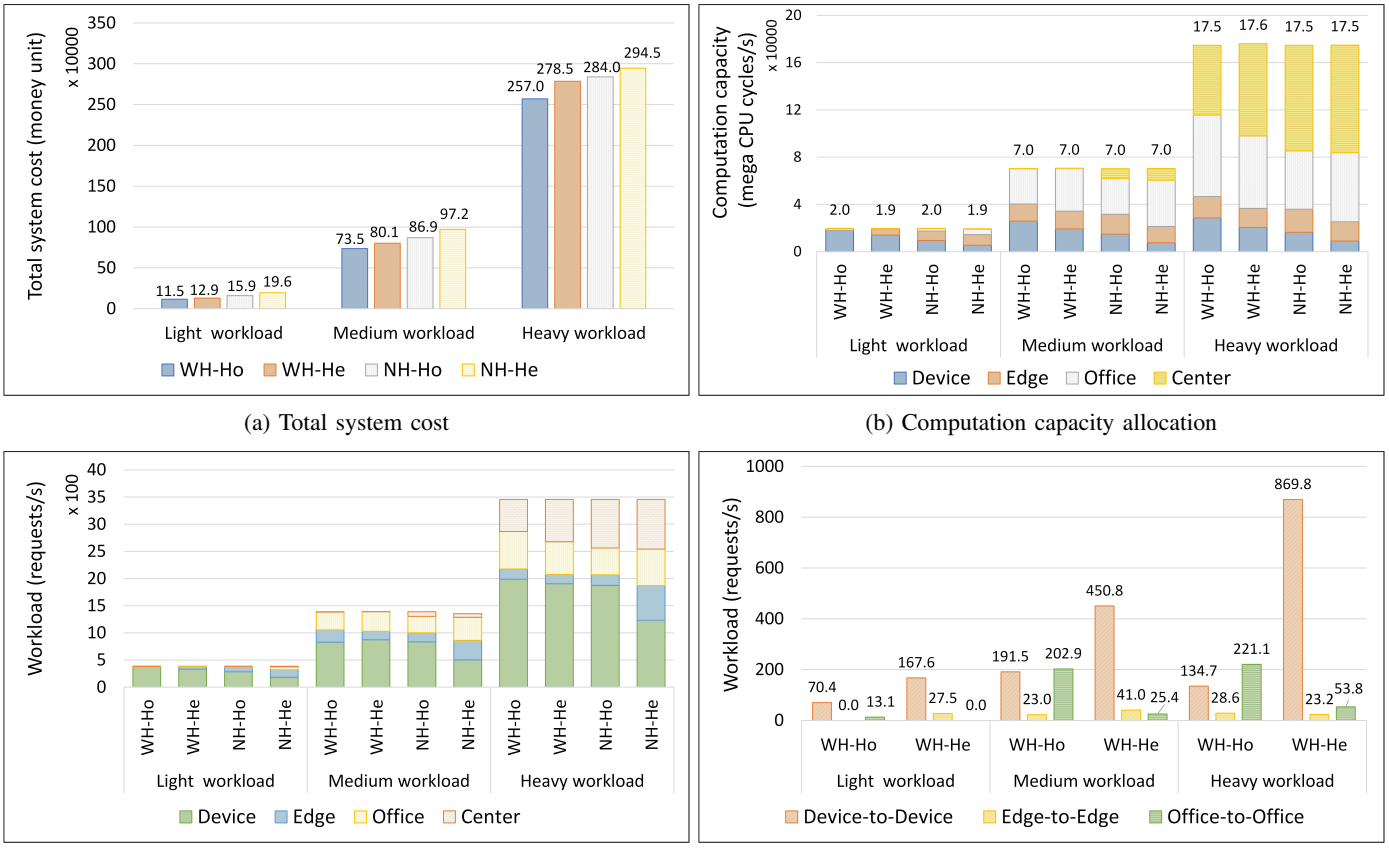
As can be seen in Figure 5a, experimental results showed that the WH design did not produce any noticeable improvement, compared to the NH design. In other words, the system cost of two designs was approximately the same in every workload case in both computation capacity cost situations. Figure 5d illustrates that there was almost no horizontal offloading by the WH design in the decreased cost scenario. Although the design still carried out some horizontal offloading operations in the equal-cost scenario, it did not lead to a significant decrease in the system cost.

Figure 5b clearly shows that most of the computation capacity was allocated to the data center tier in the decreased cost scenario since it had the cheapest computation cost. A significant number of workloads was processed by lower tiers, such as device, edge, and central office (see Figure 5c). It meant that the workloads of small computation size were processed by lower tiers. On the other hand, those of large size were offloaded to the data center tier for processing in this computation cost situation.

VI. Conclusion

In this paper, we have developed a cloud-edge computing architecture which includes vertical and offloading to efficiently and promptly process different virtualized computing services. The workload and capacity optimization model of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2937342, IEEE Transactions on Network and Service Management



(a) Total system cost

(b) Computation capacity allocation

(c) Workload allocation

(d) Horizontal offloading

Fig. 4: Performance of WH and NH designs in homogeneous and heterogeneous service allocation scenarios. (WH-Ho: With horizontal offloading in homogeneous scenario, WH-He: With horizontal offloading in heterogeneous scenario, NH-Ho: Without horizontal offloading in homogeneous scenario, NH-He: Without horizontal offloading in heterogeneous scenario)

the design was then formulated as an MINLP problem to investigate its effectiveness in different operational scenarios. We further derived an approximation algorithm which applied branch-and-bound method to find an optimal solution for the problem iteratively.

Experimental results showed that our proposed cloud-edge computing design could significantly reduce the total system cost by 34% in an unbalanced input workload scenario, compared to conventional designs which only provided vertical offloading. However, the total system cost of these two designs was roughly the same in a balanced input scenario. Furthermore, the results also show that a heterogeneous service allocation required 12% and 23% more system cost than a homogeneous scenario in WH and NH designs, respectively. Another interesting observation was that in contrast, as in an increased computation capacity cost situation, horizontal offloading did not produce a noticeable improvement in system cost in decreased and equal-cost situations.

There are still some important research problems come out of this paper which should be carefully studied in our future work when we will focus on extending our problem formulation to study the impact of service diversity on the performance of cloud-edge computing. Furthermore, a two-tier global and local optimization approach, which carries out a global level optimization on groups of service nodes, and concurrent local level optimizations on service nodes of every group of the system, will also be considered to provide the scalability for large-scale cloud-edge computing systems. Another possible future work is to develop a real testbed which allows us to implement and study different resource management and optimization models for cloud-edge computing systems.

ACKNOWLEDGMENT

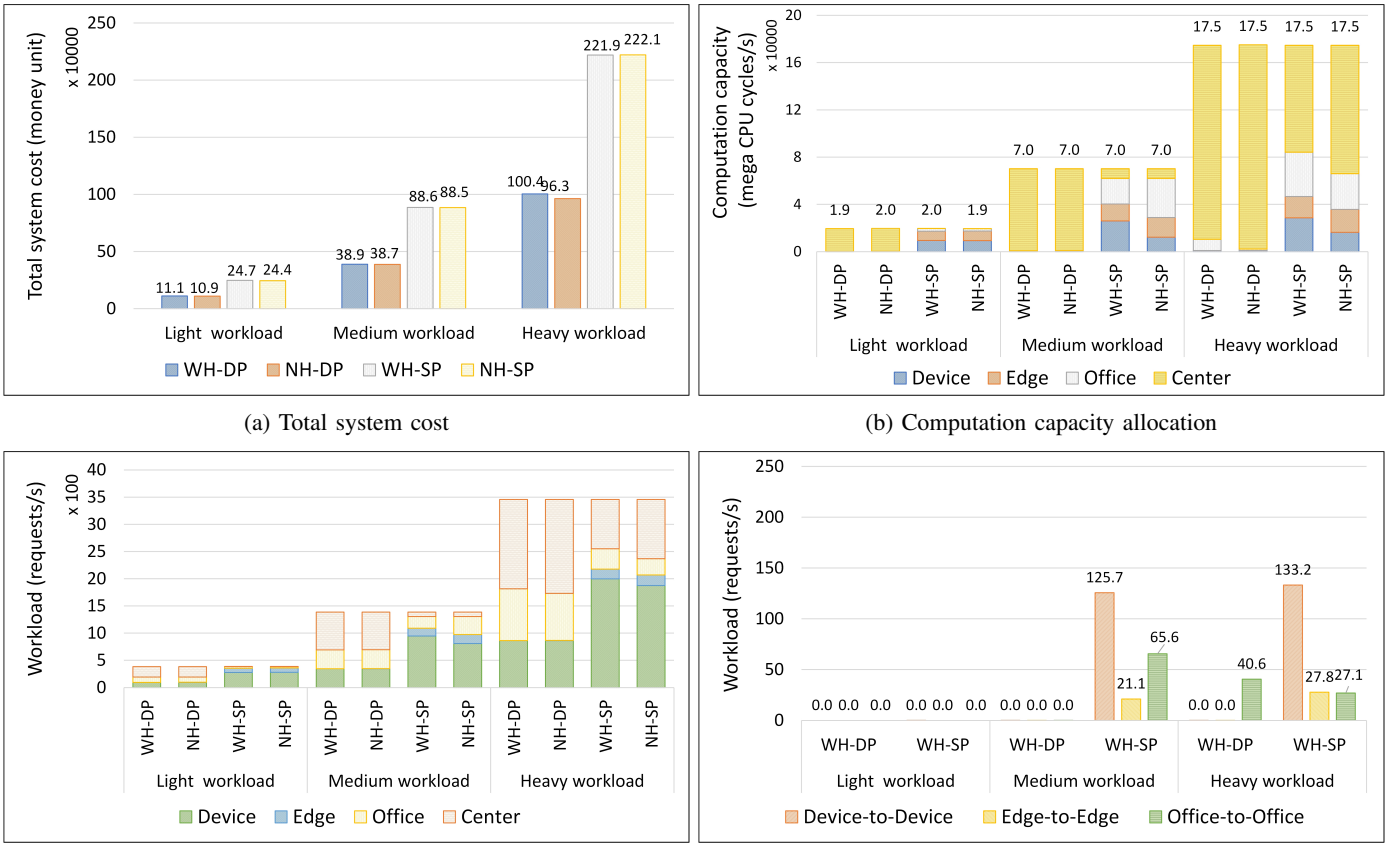
The authors would like to thank the support from the H2020 collaborative Europe/Taiwan research project 5G-CORAL (grant number 761586). This work was done when the first author was with the EECS International Graduate Program, National Chiao Tung University, Hsinchu, Taiwan.

REFERENCES

[1] M. Gharbaoui, B. Martini, and P. Castoldi, "Anycast-based optimizations for inter-data-center interconnections [Invited]," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 4, no. 11, pp. B168–B178, Nov. 2012.

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[3] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017, arXiv: 1702.05309. [Online]. Available: http://arxiv.org/abs/1702.05309



(a) Total system cost

(b) Computation capacity allocation

(c) Workload allocation

(d) Horizontal offloading

Fig. 5: Performance of WH and NH designs in decreased and equal computation capacity cost scenarios. (WH-DP: With horizontal offloading in decreased computation capacity cost scenario, NH-DP: Without horizontal offloading in decreased computation capacity cost scenario, WH-SP: With horizontal offloading in equal computation capacity cost scenario, NH-SP: Without horizontal offloading in equal computation capacity cost scenario)

[4] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, Jan. 2016, pp. 1–8.

[5] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "ENORM: A Framework For Edge NOde Resource Management," *IEEE Transactions on Services Computing*, pp. 1–1, 2018.

[6] A. Munir, P. Kansakar, and S. U. Khan, "IFCIoT: Integrated Fog Cloud IoT Architectural Paradigm for Future Internet of Things," *arXiv:1701.08474 [cs]*, Jan. 2017, arXiv: 1701.08474. [Online]. Available: http://arxiv.org/abs/1701.08474

[7] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, Apr. 2016, pp. 1–9.

[8] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid Method for Minimizing Service Delay in Edge Cloud Computing Through VM Migration and Transmission Power Control," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 810–819, May 2017.

[9] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.

[10] L. Yu, T. Jiang, and Y. Zou, "Fog-Assisted Operational Cost Reduction for Cloud Data Centers," *IEEE Access*, vol. 5, pp. 13 578–13 586, 2017.

[11] Y. Lin, Y. Lai, J. Huang, and H. Chien, "Three-Tier Capacity and Traffic Allocation for Core, Edges, and Devices for Mobile Edge Computing," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 923–933, Sep. 2018.

[12] V. B. C. Souza, W. Ramrez, X. Masip-Bruin, E. Marn-Tordera, G. Ren, and G. Tashakor, "Handling service allocation in combined Fog-cloud scenarios," in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–5.

[13] V. B. Souza, X. Masip-Bruin, E. Marin-Tordera, W. Ramirez, and S. Sanchez, "Towards Distributed Service Allocation in Fog-to-Cloud (F2c) Scenarios," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2016, pp. 1–6.

[14] G. Lee, W. Saad, and M. Bennis, "An Online Secretary Framework for Fog Network Formation with Minimal Latency," *arXiv:1702.05569 [cs, math]*, Feb. 2017, arXiv: 1702.05569. [Online]. Available: http://arxiv.org/abs/1702.05569

[15] K. Wang, H. Yin, W. Quan, and G. Min, "Enabling Collaborative Edge Computing for Software Defined Vehicular Networks," *IEEE Network*, vol. 32, no. 5, pp. 112–117, Sep. 2018.

[16] K. Toczé and S. Nadjm-Tehrani, "A Taxonomy for Management and Optimization of Multiple Resources in Edge Computing," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[17] D. Gross, J. Shortle, J. Thompson, and C. Harris, *Fundamentals of Queueing Theory, 4th Edition*. Wiley, Sep 2011. [Online]. Available: https://www.wiley.com/en-us/Fundamentals+of+Queueing+Theory%2C+4th+Edition-p-9780471791270

[18] I. Quesada and I. E. Grossmann, "An LP/NLP based branch and bound algorithm for convex MINLP optimization problems," *Computers & Chemical Engineering*, vol. 16, no. 10, pp. 937–947, Oct. 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0098135492800288

[19] L. Bertacco, M. Fischetti, and A. Lodi, "A feasibility pump heuristic for general mixed-integer problems," *Discrete Optimization*, vol. 4, no. 1, pp. 63–76, Mar. 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1572528606000855

[20] R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban, "An interior algorithm for nonlinear optimization that combines line search and trust region steps," *Mathematical Programming*, vol.

107, no. 3, pp. 391–408, Jul. 2006. [Online]. Available: https://link.springer.com/article/10.1007/s10107-004-0560-5

[21] R. Fourer, D. M. Gay, and B. W. Kernighan, "A Modeling Language for Mathematical Programming," *Management Science*, vol. 36, no. 5, pp. 519–554, May 1990. [Online]. Available: https://pubsonline.informs.org/doi/abs/10.1287/mnsc.36.5.519

[22] D. M. Gay, "The AMPL Modeling Language: An Aid to Formulating and Solving Optimization Problems," in *Numerical Analysis and Optimization*, ser. Springer Proceedings in Mathematics & Statistics. Springer, Cham, 2015, pp. 95–116. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-17689-5_5

[23] "Artelys knitro - nonlinear optimization solver," URL https://www.artelys.com/en/optimization-tools/knitro, [accessed on: 15 March 2018].

Minh-Tuan Thai joined the faculty of the College of Information and Communication Technology at Can Tho University, Vietnam and has served as a lecturer/assistant professor from 2005. He received the M.S. and Ph.D. degrees in computer science from National Chiao-Tung University (NCTU), Taiwan in 2013 and 2018, respectively. His research interests include software-defined networking, network function virtualization, computer and network security, and 5G mobile edge computing.

Ying-Dar Lin is a Distinguished Professor of computer science at National Chiao Tung University (NCTU), Taiwan. He received his Ph.D. in computer science from the University of California at Los Angeles (UCLA) in 1993. He was a visiting scholar at Cisco Systems in San Jose during 20072008, CEO at Telecom Technology Center, Taiwan, during 2010-2011, and Vice President of National Applied Research Labs (NARLabs), Taiwan, during 2017-2018. Since 2002, he has been the founder and director of Network Benchmarking Lab (NBL, www.nbl.org.tw), which reviews network products with real traffic and automated tools, and has been an approved test lab of the Open Networking Foundation (ONF) since July 2014. He also cofounded L7 Networks Inc. in 2002, later acquired by D-Link Corp, and OPrueba Inc. in 2018. His research interests include network security, wireless communications, and network softwarization. His work on multi-hop cellular was the first along this line, and has been cited over 850 times and standardized into IEEE 802.11s, IEEE 802.15.5, IEEE 802.16j, and 3GPP LTE-Advanced. He is an IEEE Fellow (class of 2013), IEEE Distinguished Lecturer (20142017), ONF Research Associate, and received in 2017 Research Excellence Award and K. T. Li Breakthrough Award. He has served or is serving on the editorial boards of several IEEE journals and magazines, and is the Editor-in-Chief of IEEE Communications Surveys and Tutorials (COMST). He published a textbook, Computer Networks: An Open Source Approach (www.mhhe.com/lin), with Ren-Hung Hwang and Fred Baker (McGraw-Hill, 2011).

Yaun-Cheng Lai received his Ph.D. degree in the Department of Computer and Information Science from National Chiao Tung University in 1997. He joined the faculty of the Department of Information Management at National Taiwan University of Science and Technology in August 2001 and has been a distinguished professor since June 2012. His research interests include performance analysis, software-defined networking, wireless networks, and IoT security.

Hsu-Tung Chien received his B.S. and M.S. degrees in computer science from Tung Hai University, Taiwan, and National Chiao Tung University (NCTU), Taiwan, in 2014 and 2017, respectively. He has been pursuing his Ph.D. in NCTU since 2017. His research interests include wireless networks, mobile networks, and protocol design. Meanwhile, he is part of H2020 projects in 5GPPP, 5G-CORAL, Crosshaul, and Transformer, to design and develop an edge and fog system, front-/back- hauls, and a vertical slicer for 5G networks.