Telecommunication Systems
https://doi.org/10.1007/s11235-023-01061-x

Cost optimization of cloud-edge-fog federated systems with bidirectional offloading: one-hop versus two-hop

Bo-Shiuan Lin[1] · Binayak Kar[1] · Tai-Lin Chin[1] · Ying-Dar Lin[2] · Chung-Yueh Chen[1]

Accepted: 11 September 2023

Abstract

Edge and fog computing technologies are akin to cloud computing but operate in closer proximity to users, offering similar services on a more widely distributed and localized scale. To enhance the computing environment and enable efficient offloading of computing requests, we propose a unified federation of these technologies, forming a federated cloud-edge-fog (CEF) system. Unlike current offloading models limited to single-hop and unidirectional vertical scenarios, our model facilitates two-hop, bidirectional (horizontal and vertical) offloading. The CEF model enables not only fog and edge devices to offload tasks to the cloud but also allows the cloud to offload tasks to the edges and fogs, creating a more dynamic and flexible computing ecosystem. To optimize this system, we formulate an optimization problem focused on minimizing the total cost while adhering to latency constraints. We employ simulated annealing as the solution approach. By adopting the proposed CEF model and optimization strategy, organizations can effectively leverage the strengths of cloud, edge, and fog computing while achieving significant cost reductions and improved task offloading efficiency. The findings from our study indicate that adopting a two-hop offloading approach can result in cost savings of 10–20% compared to the traditional one-hop method. Furthermore, when incorporating horizontal and bidirectional offloading, cost savings of approximately 12% and 20% can be achieved, respectively, in contrast to scenarios without horizontal offloading and only unidirectional vertical offloading. This advancement holds promise for optimizing computing resources and enhancing the overall performance of distributed systems in real-world applications.

Keywords Cloud-edge-fog federated systems · Bidirectional vertical offloading · Horizontal offloading · Two-hop offloading

1 Introduction

Cloud computing technology has become increasingly popular in recent years, exerting a significant impact on the development trajectory of the information industry. This architecture offers users the advantage of not having to invest in storage and maintenance, as it leverages the robust computing power and information technology resources provided by cloud service providers to fulfill user requests [1]. However, the drawback of this architecture lies in the distance between the resources and end users, making it unsuitable for latency-sensitive applications. To address the limitations of cloud computing, researchers have proposed two new paradigms: edge computing [2] and fog computing [3]. Edge computing brings compute and storage capabilities closer to the edge network, while fog computing enables numerous heterogeneous devices to collaborate and perform high computing and storage tasks [4]. Both edge and fog computing solutions offer advantages such as proximity to users and lower costs

✉ Binayak Kar
bkar@mail.ntust.edu.tw

Bo-Shiuan Lin
cc19970623@gmail.com

Tai-Lin Chin
tchin@csie.ntust.edu.tw

Ying-Dar Lin
ydlin@cs.nctu.edu.tw

Chung-Yueh Chen
c2836432@gmail.com

[1] Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Keelung Rd., Sec.4, Taipei 106, Taiwan

[2] Department of Computer Science, National Yang Ming Chiao Tung University, University Road, Hsinchu 300, Taiwan

Published online: 05 October 2023

compared to the cloud [5]. Nevertheless, edge and fog computing also have their own shortcomings. They have limited computing resources, which hinders their ability to process a large number of requests, and their processing power is relatively inferior to that of the cloud [6].

To address the aforementioned challenges, a federation technique has been adopted, which involves interconnecting various computing entities [7]. This allows for dynamic resource allocation to cater to user demands by offloading requests from one service provider to another. Such dynamic offloading has a significant impact on system performance, as it enables service providers to optimize their performance by leveraging resources from others [8]. The effectiveness of this optimization relies on factors such as computing resource availability, latency limitations, and low-cost computation, among others. Historically, most research has focused on two-tier architectures, such as cloud-edge [9, 10], cloud-fog [11], or edge-fog [12] federations. There has been relatively limited exploration of three-tier architectures [13], which involve the federation of cloud, edge, and fog components. Moreover, in current federated systems, the offloading process has been confined to unidirectional offloading, involving the transfer of tasks from one level to another. For instance, tasks are offloaded from the edge to the cloud, from the fog to the edge, or from the edge to the fog, etc.

In this paper, we introduce a generic cloud-edge-fog (CEF) federated system that employs bidirectional offloading. This means that not only can edges offload tasks to the cloud, but the cloud can also offload tasks to the edges. Our architecture comprises three tiers: the cloud is positioned at the top, the edges are in the middle, and the fogs form the bottom tier. Each edge is connected to several fogs, and the cloud is connected to all edges. Additionally, all edges in the middle tier are horizontally interconnected. Further details regarding our proposed architecture can be found in Sect. 3. Compared to existing architectures that are limited to single-hop offloading scenarios, our CEF federated system supports more versatile offloading scenarios. Within our architecture, we also considered two-hop offloading. For instance, the cloud can offload its requests not only to connected edges but also to fogs, and similarly, a fog can offload tasks to both edges and the cloud. This enhanced flexibility in offloading empowers the system to efficiently allocate tasks across different tiers, optimizing the overall performance and resource utilization.

The proposed architecture is motivated by two primary factors: computation cost and computation power. Computation cost gradually decreases if we move requests from the cloud tier to the fog tier [6]. Despite such low computation cost, resources in edges and fogs remain unused as a result of low computation power; computation power gradually increases from fog tier to cloud tier [1]. In terms of computation power, the cloud ranks highest, while the fog possesses the lowest. Correspondingly, the cloud incurs higher computation costs, whereas the fog has the advantage of the lowest computation costs. The edge's computation cost and computation power lie between those of the cloud and fog. Thus, our architecture enables the offloading of tasks from higher tiers to lower tiers to achieve low-cost computation, and vice versa, offloading tasks from lower tiers to higher tiers to handle high computation power requirements [14].

Therefore, in the envisioned generic architecture for Cloud, Edge, and Fog (CEF), bidirectional offloading facilitates mutual support among all components, allowing them to offload tasks to one another. Additionally, horizontal offloading among edge nodes further enhances their collaboration. To address the challenge, we proposed a mixed-integer optimization problem to minimize the cost with latency as a constraint. To obtain a near-optimal solution, we employed the simulated annealing (SA) algorithm. To the best of our knowledge, our research is the first to design a CEF federated system with bidirectional two-hop offloading. The key contributions of our paper can be summarized as follows.

1. We proposed a generic three-tier federated architecture where nodes in the model possess the capability to offload requests to distinct tiers or within the same tier, thereby fulfilling user demands effectively.
2. We formulate a method to minimize the total cost of the federated system while considering latency as a constraint.
3. To attain a near-optimal solution, we utilized the SA algorithm as a solution.
4. Furthermore, we have conducted performance comparisons between one-hop and two-hop offloading scenarios within the proposed architecture.

The remainder of this paper is organized as follows. In Sect. 2, related work is reviewed. We introduce our system model in Sect. 3, and formulate the cost optimization problem in Sect. 4. In Sect. 5, the solution algorithms are presented and the experimental results are given in Sect. 6. Finally, in Sect. 7 concludes this paper.

2 Related works

A variety of research papers has focused on different federated architectures between cloud, edge, and fog computing to exploit the advantages of these technologies [15–17]. Table 1 summarizes these papers and makes a comparison with our work.

Some of this research considered a CEF federated system. Chekired et al. [18] proposed a decentralized multi-tier fog architecture for Industrial Internet of Things (IIoT)

Table 1 Comparison of related works

Paper	Target network	Offloading direction	Multi-hop	Arrival	Number of servers	Objective (Minimize)	Constraints	Solution
[18]	Three-tier (Fog/Edge/Cloud)	↑	No	Only Fog	One	Latency	Task allocation	Simulated annealing algorithm
[19]	Three-tier (Edge/Office/Cloud)	↑, ← and →	No	Only Edge	Multiple	Cost	Latency and service rate	Branch-and-bound algorithm
[20]	Three-tier (Fog/Edge/Cloud)	↑, ↓, ← and →	No	Only Fog	One	Latency	Service rate and migration rate	Optimal Parallel Scheduling
[21]	Three-tier (Device/Fog/Cloud)	↑ and ↓	No	Only Device	Multiple	Latency	Service rate	Genetic algorithm
[22, 23]	Two-tier (Edge/Cloud)	↑, ↓, ← and →	No	All	Multiple	Cost	Latency and service rate	Simulated annealing algorithm
[24]	Two-tier (Edge/Cloud)	↑ and ↓	No	All	Multiple	Cost	Latency and service rate	Service provision for edge federation algorithm
[25]	Two-tier (Edge/Cloud)	↑	No	Only Edge	One	Number of served tasks	Latency	Single cloudlet and coordinated provisioning
[26]	Two-tier (Edge/Cloud)	↑	No	All	One	Energy consumption	Latency	Pruning algorithm and deep reinforcement learning
[27]	Two-tier (Edge/Cloud)	↑	No	Only Edge	One	Latency	Task allocation	Simulated annealing algorithm
[28]	Two-tier (Fog/Cloud)	↑, ↓, ← and →	No	Only Fog	One	Cost	Latency and service rate	Breadth-First Search
[29]	Two-tier (Fog/Cloud)	↑, ← and →	No	Only Fog	One	Latency	Service rate	Scheduling algorithm
[30]	Two-tier (Vehicle/Edge)	← and →	No	All	Multiple	Latency	Reserve price	Pricing-based workload distributor algorithm
[31]	Two-tier (VFC/MEC)	↓	No	Only MEC	Multiple	Cost	Latency and service rate	Decentralized offloading configuration protocol
[32]	Two-tier (Vehicle/Edge)	↑ and ↓	No	Only Vehicle	One	Energy consumption	Offloading survivability and latency	Genetic algorithm
Ours	Three-tier (Cloud/Edge/Fog)	↑, ↓, ← and →	Yes	All	Multiple	Cost	Latency and service rate	Simulated annealing algorithm

applications and developed an IIoT offloading algorithm by solving a mixed nonlinear integer programming problem. Thai et al. [19] developed a cloud-edge computing architecture that includes vertical and horizontal offloading. They derive a branch-and-bound algorithm to reduce the complexity of the problem. Deng et al. [20] divided evacuation into dispatching and scheduling to solve the burst load evacuation problem, using optimal routing and optimal parallel scheduling. Aburukba et al. [21] proposed a hybrid fog-cloud architecture. Using the genetic algorithm to address the scheduling problem to minimize the deadline misses of requests. Although these papers considered a CEF federated system, none of them addressed the issue where each tier of nodes could receive users' requests. A federated architecture with UEs, vehicular-fogs, edges, and cloud was proposed along with a probabilistic offloading strategy [33]. They used a subgradient-based algorithm to estimate the optimal offloading probabilities to minimize the Quality of Service (QoS) violation while satisfying the delay constraint. However, none of these above-discussed papers considered multi-hop offloading, and traffic arrival is limited to only one tier. In this paper, we considered multi-hop offloading, where traffic can arrive at any layer of the architecture.

The following papers are based on cloud-edge federated systems. An omnidirectional offloading in a two-tier federated cloud-edge architecture was proposed to optimize cost, using a modified simulated annealing algorithm to find near-optimal results [22, 23]. Cao et al. [24] designed an edge federation to provide services between edge infrastructure providers and the cloud. The service provision for the edge federation algorithm was also developed by these authors to dynamically resolve the problem of achieving an efficient service provision solution. Ascigil et al. [25] considered Function-as-a-Service in the edge-cloud systems. They use single cloudlet provisioning and coordinated provisioning to predict upcoming requests accurately. Dong et al. [26] discussed the efficient deployment of "cloud-edge" tasks and overall load balancing and proposed joint cloud-edge task deployment based on a pruning algorithm with deep reinforcement learning. Tong et al. [27] proposed a hierarchical edge cloud architecture that uses workload placement algorithms and adjusts the service rate to minimize the average execution delay. All these papers consider one-hop offloading. However, in this paper, we considered two-hop offloading.

The following papers focus on cloud-fog federated systems. Faticanti et al. [28] proposed a multi-domain federated fog ecosystem while guaranteeing resource locality constraints and proposed a new deployment technique based on a breadth-first search. Razaq et al. [29] proposed an integer linear programming model and a dynamic programming algorithm to preserve user privacy while minimizing the average end-to-end (E2E) delay. With experiments, these authors demonstrate that the proposed method can ensure security and effectively reduce the delay time. However, in these above papers, each node is considered as a single server, which is not reasonable. In this paper, we consider multiple servers.

The following papers are based on edge-fog federated systems. Sharmin et al. [30] proposed a micro-level fog federation environment for vehicular networks that handle delay-sensitive applications and derived a pricing-based workload distributor algorithm to balance the workload. Yen et al. [31] designed a two-tier edge and vehicular-fog federated architecture and proposed a decentralized offloading configuration protocol (DOCP) for low-cost offloading. Mourad et al. [32] propose a vehicular edge computing fog-enabled scheme that allows a task to be offloaded to a federated vehicle node. They minimized computation execution time and energy consumption and used the genetic algorithm as a solution. However, in these papers, bidirectional vertical offloading and horizontal offloading have not been considered simultaneously. Using bidirectional vertical offloading and horizontal offloading simultaneously can effectively distribute a request to the appropriate node for better results, and this is the crux of our paper.

3 System model

3.1 CEF federated systems

In this section, we discuss our proposed CEF federated systems. The notations and variables used in this paper are enumerated in Table 2. As illustrated in Fig. 1, the system model comprises three tiers: cloud, edge, and fog. There is a single cloud node in the cloud tier, which is connected to m edge nodes in the edge tier, and each edge node is connected to n_j fog nodes in the fog tier. All edges in the edge tier are connected to each other. For simplification, we have not considered any direct communication between any two fogs. A fog can communicate with its sibling fogs via its parent edge node. The communication between two connected nodes is bidirectional. Each node in each tier has a different number of servers, and each server has different capacities. The cloud node has n'' servers, each with service rate μ''. For the edge tier, each edge e_j has n'_j servers, each with service rate μ'_j. Similarly, for the fog tier, each fog consists of $n_{j,i}$ servers, each with service rate $\mu_{j,i}$. Since each node in each tier has computing resources, each can receive a request (λ'' for cloud tier, λ'_j for edge tier, or $\lambda_{j,i}$ for fog tier) directly from a user. A node can offload part of the request to other nodes to reduce the cost as long as the latency constraint is satisfied. The cost is calculated based on the number of requests transmitted from one node to another and processed

Table 2 Notation table

Notation	Description
C	Cloud node
E, F	Set of edge nodes, fog nodes
m, n_j	Number of edge nodes of the system, number of fog nodes connected to jth edge node
j, i	$1\sim m$, $1\sim n_j$
e_j, $f_{j,i}$	jth edge node in E, ith node in jth fog in F where jth fog is connected to e_j
c	Erlang's c formula
Request	
λ'', λ'_j, $\lambda_{j,i}$	Input request rate from user to C, e_j, and $f_{j,i}$
R'', R'_j, $R_{j,i}$	Total requests rate processed by C, e_j, and $f_{j,i}$
β''_j, β'_j, $\beta^{\star}_{j,j'}$, $\beta'_{j,i}$, $\beta_{j,i}$	Offloading probabilities from C to e_j, e_j to C, e_j to $e_{j'}$, e_j to $f_{j,i}$, and $f_{j,i}$ to e_j
$\hat{\beta}$, $\hat{\beta}_j$, $\hat{\beta}_{j,i}$	Un-offloading probabilities by C, e_j, and $f_{j,i}$
Computing resource	
μ'', μ'_j, $\mu_{j,i}$	Service rate of a server in C, e_j, and $f_{j,i}$
r'_j, $r'_{j,j'}$, $r_{j,i}$	Transmission rate between C and e_j, e_j and $e_{j'}$, and e_j and $f_{j,i}$
n'', n'_j, $n_{j,i}$	Number of servers in C, e_j, and number of devices in $f_{j,i}$
Cost	
S	Total cost
$S''_{C,E}$, $S'_{E,E'}$, $S_{E,F}$	Total communication cost between cloud and edge tier, within the edge tier, and between edge and fog tier
S''_{total}, S'_{total}, S_{total}	Total computation cost of cloud tier, edge tier, and fog tier
[5pt] $s''_{C,E}$, $s'_{E,E'}$, $s_{E,F}$	Unit communication cost between cloud and edge tier, within the edge tier, and between edge and fog tier
s'', s', s	Unit computation cost of cloud tier, edge tier, and fog tier
Latency	
T''_j, T'_j, $T^{\star}_{j,j'}$, $T'_{j,i}$, $T_{j,i}$	Communication latency from C to e_j, e_j to C, e_j to $e_{j'}$, e_j to $f_{j,i}$, and $f_{j,i}$ to e_j
l'', l'_j, $l_{j,i}$	Computation latency of C, e_j, and $f_{j,i}$
$\mathcal{L}_{max}$	Latency limitation

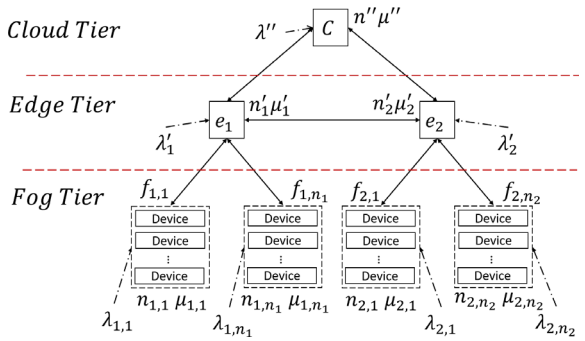


Fig. 1 Proposed CEF federated systems

by the nodes. Our objective is to minimize the total cost, subject to the presumed latency constraint satisfaction.

3.2 One-hop versus two-hop offloading

Our proposed model is not limited to single-hop offloading; we consider both one and two-hop offloading. Table 3 shows all possible offloading scenarios in our architecture, where ONE and TWO represent one-hop and two-hop, respectively. Because of the page limit of the paper, the fog tier is shown as a figure, and the edge tier and cloud tier are described in the text.

3.2.1 One-hop offloading

Because each fog node is only connected to only one edge node in the fog tier, a request may be offloaded to the parent edge node. As shown in Fig. 2, $\beta_{j,i}$ is the offloading probability from $f_{j,i}$ to e_j. $\beta_{j,i}\lambda_{j,i}$ is amount of requests are offloaded to e_j, and $\hat{\beta}_{j,i}\lambda_{j,i}$ is the number of requests that are kept in $f_{j,i}$. For the edge tier, an edge node can offload

Table 3 One-hop and two-hop offloading scenarios

Request received by	Offloaded to cloud	Offloaded to edge	Offloaded to fog
Cloud		ONE (All edges)	TWO (All fogs)
Edge	ONE	ONE (Adjacent edges)	ONE (Child fogs)
			TWO (Children of adjacent edges)
Fog	TWO	ONE (Parent edge)	TWO (Sibling fogs)
		TWO (Adjacent edges of the parent)	

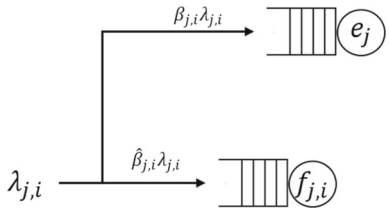


Fig. 2 One-hop offloading for fog tier

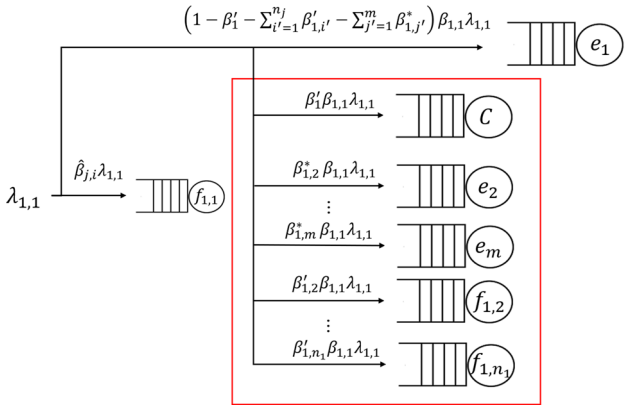


Fig. 3 Two-hop offloading for fog tier

its requests to the cloud, adjacent edges, or child fog nodes. The cloud can offload jobs to all the edge nodes.

3.2.2 Two-hop offloading

Figure 3 shows two-hop offloading for the fog tier. The part enclosed by the red rectangle illustrates the second-hop offloading. In two-hop offloading, a fog node can offload its request via the parent edge to the cloud, via the parent edge to other edges that are adjacent to its parent, and via the parent edge to its sibling fog nodes. β'_j, $\beta^\star_{j,j'}$, and $\beta'_{j,i'}$ represent the offloading probability from e_j to C, $e_{j'}$ and $f_{j,i'}$, respectively. For the edge tier, an edge can offload its request to the fogs that are directly connected to adjacent edges. Similarly, the cloud can offload its request to all fogs with two-hop offloading.

4 Problem formulation

In this section, we analyze the system model of one-hop offloading by calculating latency and cost. The analysis of two-hop offloading is given in Appendix A.

4.1 Latency calculation

The latency calculation is divided into two parts: communication and computation, and the latency is derived depending on different tiers of nodes, assuming that computation workloads are given according to the Poisson process [34].

4.1.1 Communication latency

We use the sojourn time equation of the M/M/1 queuing model to derive communication latency but do not consider the propagation latency from the user to each node. There are five different offloading scenarios for one-hop offloading. Let $T_{j,i}$, $T'_{j,i}$, $T^\star_{j,j'}$, T'_j, and T''_j be the communication latency for transmitting one request from $f_{j,i}$ to e_j, e_j to $f_{j,i}$, e_j to $e_{j'}$, e_j to C, and C to e_j which can be represented as follows.

$$T_{j,i} = \frac{1}{r_{j,i} - \lambda_{j,i}\beta_{j,i}}, \quad r_{j,i} \geq \lambda_{j,i}\beta_{j,i}, \quad \forall j, i, \tag{1}$$

$$T'_{j,i} = \frac{1}{r_{j,i} - \lambda'_j\beta'_{j,i}}, \quad r_{j,i} \geq \lambda'_j\beta'_{j,i}, \quad \forall j, i, \tag{2}$$

$$T^\star_{j,j'} = \frac{1}{r'_{j,j'} - \lambda'_j\beta^\star_{j,j'}}, \quad j \neq j', \qquad r'_{j,j'} \geq \lambda'_j\beta^\star_{j,j'}, \quad \forall j, j', i, \tag{3}$$

$$T'_j = \frac{1}{r'_j - \lambda'_j\beta'_j}, \quad r'_j \geq \lambda'_j\beta'_j, \quad \forall j, \tag{4}$$

$$T''_j = \frac{1}{r'_j - \lambda''\beta''_j}, \quad r'_j \geq \lambda''\beta''_j, \quad \forall j, \tag{5}$$

where $\lambda_{j,i}\beta_{j,i}$ is the request offloaded from $f_{j,i}$ to e_j, $\lambda'_j\beta'_{j,i}$ is the request offloaded from e_j to $f_{j,i}$, $\lambda'_j\beta^\star_{j,j'}$ is the request offloaded from e_j to $e_{j'}$, $\lambda'_j\beta'_j$ is the request offloaded from e_j to C, and $\lambda''\beta''_j$ is the request offloaded from C to e_j.

Please refer to Appendix A.1 for the communication latency of the two-hop offloading.

4.1.2 Computation latency

For computation latency, we consider an M/M/C model as we assume that the cloud node and each edge node have multiple servers and that each device in each fog node is a computation server. In the first step, we calculate the total requests processed by each node and then estimate the computation latency. Let $R_{j,i}$ be the total request processed by $f_{j,i}$, which can be represented as

$$R_{j,i} = \lambda_{j,i} - \lambda_{j,i}\beta_{j,i} + \lambda'_j\beta'_{j,i}, \tag{6}$$

where $\lambda_{j,i}\beta_{j,i}$ is the request offloaded to e_j and $\lambda'_j\beta'_{j,i}$ is the request received from e_j. Let R'_j be the total request processed by e_j, which can be represented as

$$R'_j = \lambda'_j - \lambda'_j\beta'_j + \lambda''\beta''_j - \sum_{i=1}^{n_j}\lambda'_j\beta'_{j,i} + \sum_{i=1}^{n_j}\lambda_{j,i}\beta_{j,i} - \sum_{j'=1,j'\neq j}^{m}\lambda'_j\beta^{\star}_{j,j'} + \sum_{j'=1,j'\neq j}^{m}\lambda'_{j'}\beta^{\star}_{j',j}, \tag{7}$$

where $\lambda'_j\beta'_j$, $\lambda'_j\beta'_{j,i}$, and $\lambda'_j\beta^{\star}_{j,j'}$ are the requests offloaded to C, $f_{j,i}$, and $e_{j'}$. $\lambda''\beta''_j$, $\lambda_{j,i}\beta_{j,i}$, and $\lambda'_{j'}\beta^{\star}_{j',j}$ are the requests received from C, $f_{j,i}$, and $e_{j'}$. Let R'' be the total requests processed by C, which can be represented as

$$R'' = \lambda'' - \sum_{j=1}^{m}\lambda''\beta''_j + \sum_{j=1}^{m}\lambda'_j\beta'_j, \tag{8}$$

where $\lambda''\beta''_j$ are the requests offloaded to e_j and $\lambda'_j\beta'_j$ are the requests received from e_j. Let $l_{j,i}$, l'_j, and l'' be the computation latency for processing one request by $f_{j,i}$, e_j, and C which can be represented as follows:

$$l_{j,i} = \frac{c\left(n_{j,i}, \frac{R_{j,i}}{\mu_{j,i}}\right)}{n_{j,i}\mu_{j,i} - R_{j,i}} + \frac{1}{\mu_{j,i}}, \quad \forall j, i, \tag{9}$$

$$l'_j = \frac{c\left(n'_j, \frac{R'_j}{\mu'_j}\right)}{n'_j\mu'_j - R'_j} + \frac{1}{\mu'_j}, \quad \forall j, \tag{10}$$

$$l'' = \frac{c\left(n'', \frac{R''}{\mu''}\right)}{n''\mu'' - R''} + \frac{1}{\mu''}, \tag{11}$$

where $c()$ is Erlang's c formula [34]. Please refer to Appendix A.2 for the computation latency of the two-hop offloading.

4.2 Cost calculation

4.2.1 Communication cost

To calculate the communication cost, we need to determine the number of offloading requests between tiers and within tiers and then multiply this by the unit communication cost. The communication costs between the cloud tier and edge tier ($S''_{C,E}$), within the edge tier ($S'_{E,E'}$), and between the edge tier and fog tier ($S_{E,F}$) are denoted as follows.

$$S''_{C,E} = s''_{C,E}\left(\sum_{j=1}^{m}\lambda''\beta''_j + \sum_{j=1}^{m}\lambda'_j\beta'_j\right), \tag{12}$$

$$S'_{E,E'} = s'_{E,E'}\left(\sum_{j=1}^{m}\sum_{j'=1,j'\neq j}^{m}\lambda'_j\beta^{\star}_{j,j'}\right), \tag{13}$$

$$S_{E,F} = s_{E,F}\left(\sum_{j=1}^{m}\sum_{i=1}^{n_j}\lambda'_j\beta'_{j,i} + \sum_{j=1}^{m}\sum_{i=1}^{n_j}\lambda_{j,i}\beta_{j,i}\right), \tag{14}$$

where $s''_{C,E}$, $s'_{E,E'}$, and $s_{E,F}$ are the unit communication cost between cloud and edge tier, within the edge tier, and between edge and fog tier, respectively. Please refer to Appendix A.3 for the communication cost of the two-hop offloading.

4.2.2 Computation cost

The computation cost here is calculated based on the number of requests processed in each tier and multiplied by the unit computation cost. Let S'_{total}, S'_{total}, and S_{total} be the computation cost of cloud, edge, and fog tiers, respectively, which can be represented as follows.

$$S''_{total} = s''R'', \tag{15}$$

$$S'_{total} = s'\sum_{j=1}^{m}R'_j, \tag{16}$$

$$S_{total} = s\sum_{j=1}^{m}\sum_{i=1}^{n_j}R_{j,i}, \tag{17}$$

where s'', s', and s are the unit computation cost of cloud, edge, and fog tiers, respectively. Please refer to Appendix A.4 for the computation cost of the two-hop offloading.

4.3 Objective and constraints

The objective function of one-hop offloading is calculated by the sum of the communication and computation cost in Eq. (18). The problem is formulated as follows.

$$\text{Minimize } (S''_{C,E} + S'_{E,E'} + S_{E,F} + S''_{total} + S'_{total} + S_{total}), \tag{18}$$

$s.t.$

$$\hat{\beta} l'' + \sum_{j=1}^{m} \beta_j'' \left(l_j' + T_j'' \right) \leq \mathcal{L}_{max}, \tag{19}$$

$$\hat{\beta}_j l_j' + \beta_j' \left(l'' + T_j' \right) + \sum_{i=1}^{n_j} \beta_{j,i}' \left(l_{j,i} + T_{j,i}' \right) + \sum_{j'=1, j' \neq j}^{m} \beta_{j,j'}^{\star} \left(l_{j'}' + T_{j,j'}^{\star} \right) \leq \mathcal{L}_{max}, \tag{20}$$

$$\hat{\beta}_{j,i} l_{j,i} + \beta_{j,i} \left(l_j' + T_{j,i} \right) \leq \mathcal{L}_{max}, \tag{21}$$

$$\hat{\beta} + \sum_{j=1}^{m} \beta_j'' = 1, \tag{22}$$

$$\hat{\beta}_j + \beta_j' + \sum_{i=1}^{n_j} \beta_{j,i}' + \sum_{j'=1, j' \neq j}^{m} \beta_{j,j'}^{\star} = 1, \quad \forall j, \tag{23}$$

$$\hat{\beta}_{j,i} + \beta_{j,i} = 1, \quad \forall j, i, \tag{24}$$

$$R'' < n'' \mu'', \tag{25}$$

$$R_j' < n_j' \mu_j', \quad \forall j, \tag{26}$$

$$R_{j,i} < n_{j,i} \mu_{j,i}, \quad \forall j, i. \tag{27}$$

The constraints in Eqs. (19), (20), and (21) ensure that the sum of the computation and communication latency does not exceed the threshold for the cloud, edge, and fog tiers, respectively. The constraints in Eqs. (22), (23), and (24) ensures the law of total probability in the cloud, edge, and fog, respectively. The constraints in Eqs. (25), (26), and (27) ensure that the total requests processed by the cloud, edge, and fog are less than the total available service rate. The objective function of two-hop offloading is given in Appendix A.5.

5 Proposed algorithm

In order to solve the cost optimization problem noted above, we adopted the simulated annealing [35] to find the near-optimal solution for the best offloading probabilities between nodes so as to minimize total cost.

Algorithm 1 describes the framework of the simulated annealing. In each iteration, a new solution is generated, and the objective function S_{new} is calculated in Eq. (18). Let Δ_S denote the difference in cost between the current iteration and the previous iteration. If $\Delta_S \leq 0$, the S_{new} is accepted. Otherwise, the solution is accepted by probability $exp\left(-\frac{\Delta_S}{T}\right)$. Finally, check whether it is better than the current saved optimal solution. The initial temperature T_{ini}, the final temperature T_{ter}, and the cooling parameter α $(0 < \alpha < 1)$ control the progress of the algorithm until the temperature is less than the terminal threshold T_{ter}. The run time of the SA algorithm does not have any bounded time, as the cooling parameter determines the time complexity [35].

Algorithm 1 Simulated Annealing Algorithm

Input: $\lambda'', \lambda_j', \lambda_{j,i}, \mu'', \mu_j', \mu_{j,i}, s_{C,E}'', s_{E,E'}', s_{E,F}, s'', s', s, \mathcal{L}_{max}$

Output: $\hat{\beta}, \hat{\beta}_j, \hat{\beta}_{j,i}, \beta_j'', \beta_j', \beta_{j,j'}^{\star}, \beta_{j,i}', \beta_{j,i}$

1: Randomly generate $\hat{\beta}, \hat{\beta}_j, \hat{\beta}_{j,i}, \beta_j'', \beta_j', \beta_{j,j'}^{\star}, \beta_{j,i}', \beta_{j,i}$ and calculate the objective function S_{old} by Eq. (18)
2: $T \leftarrow T_{ini}$
3: **while** $T > T_{ter}$ **do**
4: Generate a new solution by Algorithm 2 and calculate the objective function S_{new}
5: $\Delta_S = S_{new} - S_{old}$
6: **if** $\Delta_S \leq 0$ **then**
7: $S_{old} \leftarrow S_{new}$
8: **else if** $Random\ (0, 1) < exp\left(-\frac{\Delta_S}{T}\right)$ **then**
9: $S_{old} \leftarrow S_{new}$
10: **end if**
11: **if** $S_{old} \leq S_{optimal}$ **then**
12: $S_{optimal} \leftarrow S_{old}$
13: **end if**
14: $T \leftarrow \alpha \cdot T$
15: **end while**

Algorithm 2 Generate A New Solution

Input: $\Delta_S, \hat{\beta}_j, \hat{\beta}_{j,i}, \hat{\beta}_j, \beta_j'', \beta_j', \beta_{j,j'}^{\star}, \beta_{j,i}', \beta_{j,i}$

Output: $\hat{\beta}, \hat{\beta}_j, \hat{\beta}_{j,i}, \beta_j'', \beta_j', \beta_{j,j'}^{\star}, \beta_{j,i}', \beta_{j,i}$

1: **if** $\Delta_S \leq 0$ **then**
2: $\hat{\beta} \leftarrow \hat{\beta} \cdot (1 - \gamma)$
3: $\hat{\beta}_j \leftarrow \hat{\beta}_j \cdot (1 - \gamma)$
4: $\hat{\beta}_{j,i} \leftarrow \hat{\beta}_{j,i} \cdot (1 + \gamma)$
5: **else if** $\Delta_S > 0$ **then**
6: $\hat{\beta} \leftarrow \hat{\beta} \cdot (1 + \gamma')$
7: $\hat{\beta}_j \leftarrow \hat{\beta}_j \cdot (1 + \gamma')$
8: $\hat{\beta}_{j,i} \leftarrow \hat{\beta}_{j,i} \cdot (1 - \gamma')$
9: **end if**
10: **repeat**
11: Randomly generate $\beta_j'', \beta_j', \beta_{j,j'}^{\star}, \beta_{j,i}', \beta_{j,i}$
12: Calculate computation latency and communication latency based on the generated probability
13: **until** the latency constraints are satisfied

Algorithm 2 is used to generate a solution in Algorithm 1, and is executed after every iteration of SA, if $\Delta_S \leq 0$ in Algorithm 1. The $\hat{\beta}$ and $\hat{\beta}_j$ of the cloud and edge will decrease by γ percent and the $\hat{\beta}_{j,i}$ the fog will increase by γ percent. Then, if $\Delta_S > 0$ the $\hat{\beta}$ and $\hat{\beta}_j$ of the cloud and edge will increase by γ' percent and the $\hat{\beta}_{j,i}$ the fog will decrease by γ' percent. Then algorithm process will continue to randomly generate offloading probabilities to other connected nodes and calculate computation and communication latency based on the probabilities obtained until suitable offloading probabilities are found. Finally, the obtained probabilities are returned to Algorithm 1 for the next iteration. The reason we designed it this way is that for both the cloud and each edge, we can obtain better results by offloading the request, while for each fog, the offloading will make the result worse. When $\Delta_S \leq 0$, SA will reduce the number of requests processed by the cloud and each edge and increase the number of requests processed

Table 4 Parameters table

	Cloud	Edge	Fog
Number of server or device	10 servers	10 servers	20 devices
Service rate of one server or one device	200K request/s	50K request/s	5K request/s
Arrival rate	600K request/s	150K request/s	30K request/s
Latency constraint	1 s	1 s	1 s
Unit computation cost	20/request	15/request	10/request

	Between cloud and edge	Between edge and edge	Between edge and fog
Transmission rate	1250K request/s	1250K request/s	1250K request/s
Unit communication cost	4 /request	3 /request	2 /request

by each fog. Conversely, when $\Delta_S > 0$, it will increase the number of requests processed by the cloud and each edge and reduce the number of requests processed by each fog.

6 Numerical results

In this section, we put the proposed generic architecture of a CEF federated system with one-hop and two-hop offloading into practice and subsequently conduct a comparative analysis of various aspects between the two approaches. While previous research has focused on existing three-tier architectures [18–21], predominantly limited to one-hop offloading, our paper goes beyond by exploring two-hop offloading possibilities.

6.1 Experiment scenarios

The CEF federated systems consisted of one cloud node with 10 servers, three edge nodes, each with 10 servers, each edge connecting with six fogs, each consisting of 20 devices for the experiment. To set the parameters for our experiments, we referred to [19, 31]. The service rate of a server was 200K requests/sec for the cloud, 50K requests/sec for each edge, and the service rate of a device was 5K requests/sec for each fog. The latency constraint for each request was set at one second. NORDUnet [24], a research and education network, is utilized to study workloads over time, revealing that the average workload in most areas amounts to one-third of the maximum workload. We thus designed arrival rates at 30% of the service rate of each node. Hence, the arrival rate was 600K requests/sec, 150K requests/sec, and 300K requests/sec for the cloud, edge, and fog, respectively. In subsequent experiments, the arrival rate of this setting was regarded as the standard. The transmission rate between any two nodes was set to 1250k requests per second. We assumed

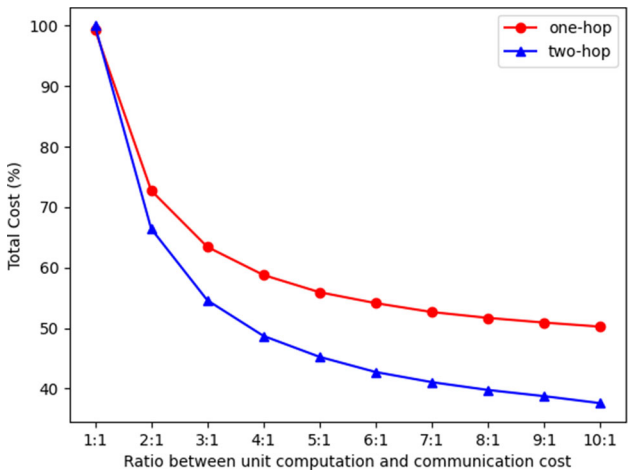


Fig. 4 Change in the ratio between computation and communication cost

there was no change in the transmission rate resulting from the simultaneous transmission of many requests. The initial temperature T_{ini}, cooling parameter α, and final temperature T_{ter} in simulated annealing are set to 300, 0.9, and 10. We explore multiple values ranging from 0 to 1 for γ and γ', and finally set them to 0.3 and 0.1, respectively, following [22]. The detailed parameter settings for our experiment are listed in Table 4.

Figure 4 shows the result of the change in the ratio between computation and communication costs. We fixed unit computation costs at 20/request, 15/request, and 10/request for the cloud, edge, and fog, respectively, and adjusted the communication cost accordingly. In cases where the computation cost and communication cost have a 1:1 ratio, two-hop offloading results in a higher average cost than one-hop offloading due to the increased communication load. For our experiment, we opted for a median value of 5:1 as the standard computation and communication ratio.

6.2 One-hop versus two-hop offloading analysis

6.2.1 Total cost: one-hop versus two-hop

Figure 5 demonstrates the cost variations of one and two-hop offloading concerning changes in arrival rates at different tiers. Specifically, Fig. 5a indicates that two-hop offloading consistently outperforms one-hop offloading, yielding up to 21% cost savings, especially when the arrival rate increases to 1800K. This superiority arises because two-hop offloading allows the cloud to offload requests to a fog, which has a lower unit computation cost than an edge or cloud, leading to better performance. However, when the fog tier is full, the costs increase more rapidly, as edges cannot offload requests to fogs. This scenario is evident in Fig. 5a at an arrival rate of 1200K. Furthermore, Fig. 5b illustrates the changes in arrival rate at the edge tier while keeping the cloud and fog tiers unchanged. In this case, two-hop offloading exhibits approximately 10% better cost performance compared to one-hop offloading. This improvement is attributed to the more efficient offloading of requests from the cloud to fog in the two-hop approach. Based on these findings, we conclude that two-hop offloading offers better cost efficiency compared to one-hop offloading.

6.2.2 Workload: one-hop versus two-hop

Figures 6 and 7 illustrate the percentages of workload offloaded under various latency constraints. We analyzed four different latency constraints: 1 s, 0.5 s, 0.1 s, and 0.05 s, to assess system performance.

Figure 6 presents the results of requests received by the cloud concerning different latency constraints. With one-hop offloading, more requests are offloaded from the cloud to the edge tier. In contrast, with two-hop offloading, more requests are directed to the fog tier, leading to better performance when the latency constraint is 0.5 s or more. As the constraint tightens to 0.1 s, approximately 60% and 30% of requests are offloaded from the cloud to the edge and fog tiers, respectively, as depicted in Fig. 6b. However, as the latency constraint becomes even more stringent at 0.05 s, both one-hop and two-hop offloading strategies predominantly leave most of the requests in the cloud tier for processing. This is because the edge and fog have a lower service rate than the cloud, resulting in increased processing time for requests in these tiers (Fig. 6a, b). Consequently, to meet the stringent latency constraint, the cloud needs to handle the majority of the requests.

Figure 7 shows the results of requests received by an edge under different latency constraints for both one-hop and two-hop offloading. When the latency constraint is 0.5 s or greater, the edge prioritizes offloading requests to the fog tier in both one-hop and two-hop offloading scenarios. In two-hop offloading (Fig. 7b), a larger proportion of requests are offloaded to the fog tier compared to one-hop offloading (Fig. 7a). This is because, in one-hop offloading, offloading is limited from one edge to other edge nodes, whereas in two-hop offloading, an edge can offload to another edge, which can then further offload the request to fog nodes. As the latency constraint reduces to 0.1 s, both one-hop and two-hop offloading still result in approximately 50–55% of requests being offloaded to the fog tier due to the stringent latency constraint, with the remaining 30–35% of requests remaining at the edge tier for processing. Further reducing the latency constraint to 0.05 s (Fig. 7a, b) results in both one-hop and two-hop offloading leaving around 60% of the requests in the edge tier for processing. About 20% of the requests are offloaded to the cloud tier, and the remainder is offloaded to other edges. This adjustment in offloading behavior is necessary to meet the extremely strict latency constraint.

6.2.3 Average sojourn time: one-hop versus two-hop

Figure 8 shows the average sojourn time for one and two-hop offloading scenarios when the cloud, edge, and fog tiers receive requests. The sojourn time estimation is based on the total communication time and computation time. When the cloud receives a request, two-hop offloading exhibits a relatively longer average sojourn time compared to one-hop offloading. This is because, in two-hop offloading, requests are offloaded to the fog tier, which introduces additional communication time, while in one-hop offloading, requests are only offloaded to the edge tier. Additionally, requests processed in fog take longer than those processed in the edge, resulting in longer computation time in two-hop offloading. Similarly, when a request is received by the edge, both one and two-hop offloading strategies prioritize offloading requests to the fog for low-cost processing. However, two-hop offloading leads to longer sojourn time due to the following reasons: In one-hop offloading, offloading requests are directed from an edge to its child fogs, while in two-hop offloading, requests can be offloaded to fogs that are children of adjacent edges. As a result, when fogs become full in one-hop offloading, requests are processed by edges, and the cloud can expedite computation time. In contrast, in two-hop offloading, requests can still be processed by fogs that are children of adjacent edges, thereby slowing computation time. When a fog receives a request, it can process the request internally or offload it to its parent edge in one-hop offloading. In two-hop offloading, a fog can offload requests to sibling fogs via a parent edge, adjacent edges of the parent edge, and the cloud. To prioritize low-cost processing, the fog will prefer to handle requests internally. Hence, two-hop offloading results in a longer sojourn time compared to one-hop offloading.

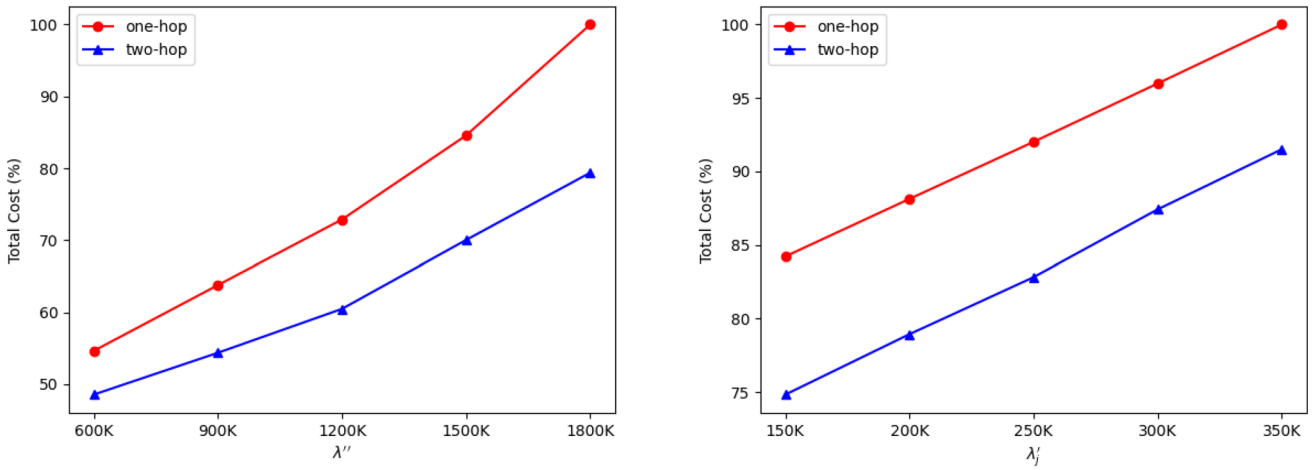


Fig. 5 Total cost with change in arrival rate in **a** cloud, and **b** edge

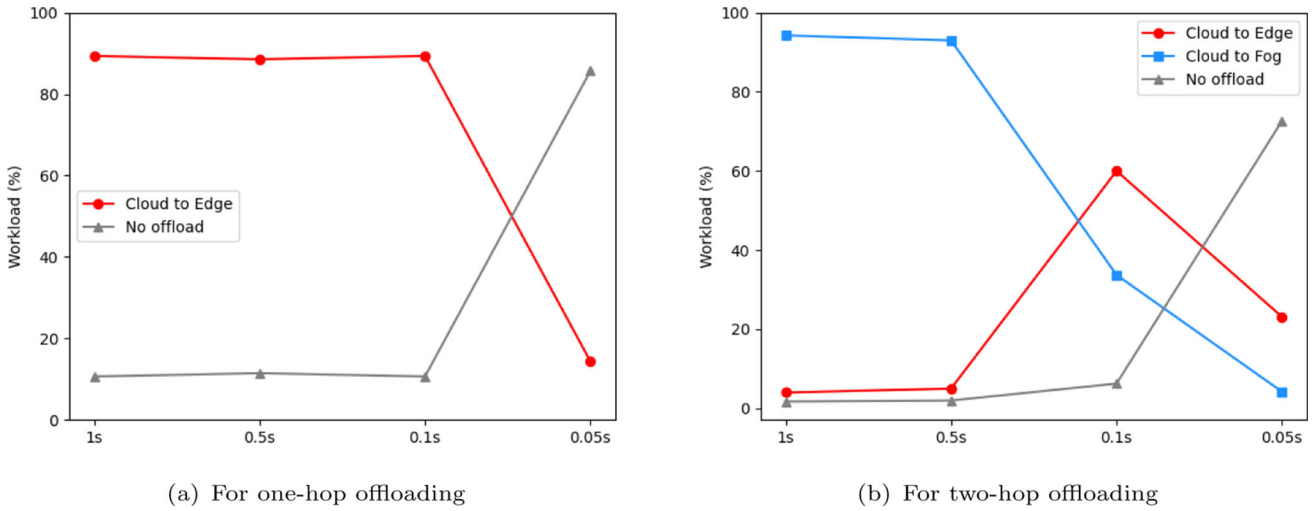


Fig. 6 Request received by Cloud with different latency constraint

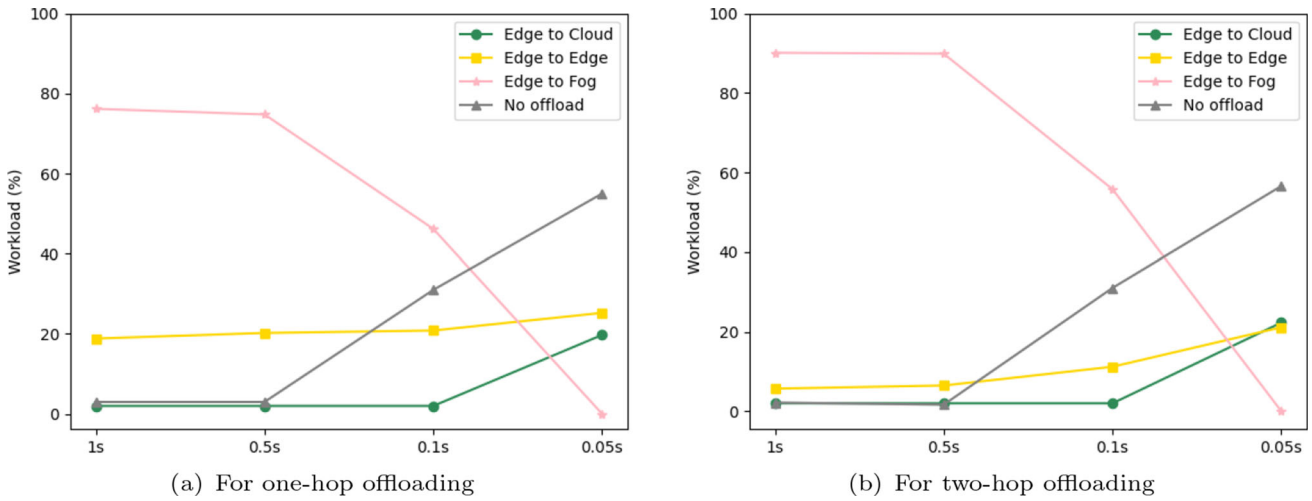


Fig. 7 Request received by Edge with different latency constraint

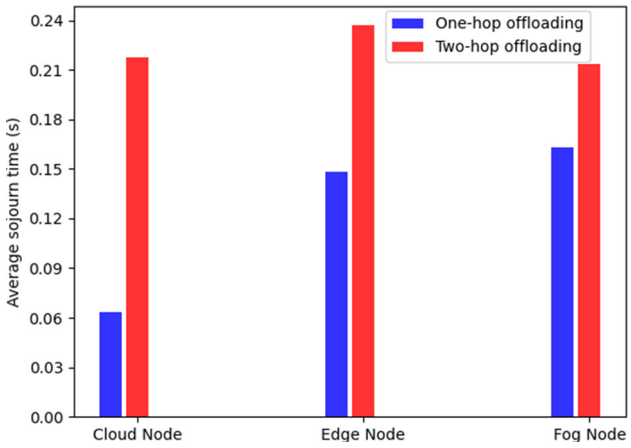


Fig. 8 Average sojourn time of one-hop and two-hop offloading

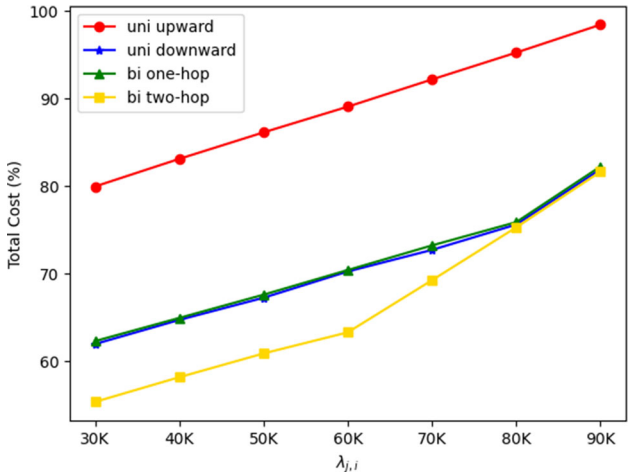


Fig. 9 Total cost with unidirectional and bidirectional offloading when change in the arrival rate of Fog

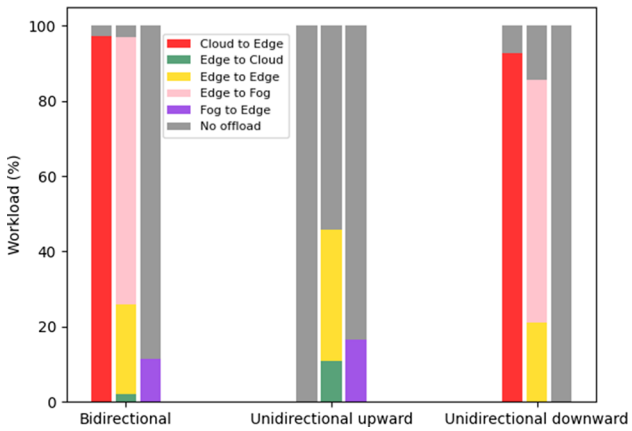


Fig. 10 Offloading result of unidirectional and bidirectional offloading when the requests are received by the cloud, edge, and fog

6.3 Unidirectional versus bidirectional offloading analysis

In this section, we will analyze the performance of unidirectional and bidirectional vertical offloading. Unidirectional vertical offloading encompasses two types: vertically upward and vertically downward offloading. In the former, lower-tier nodes can only offload tasks to nodes in upper tiers, while in the latter, nodes in upper tiers can only offload tasks to nodes in lower tiers. On the other hand, bidirectional offloading allows both upper-tier and lower-tier nodes to offload tasks to each other, creating a more flexible and interconnected offloading mechanism.

6.3.1 Total cost: unidirectional versus bidirectional

Figure 9 illustrates the changes in arrival rate in the fog tier, while the cloud and edge remain unchanged. For arrival rates up to 60K, two-hop offloading results in a 10% lower total cost compared to one-hop offloading. However, as the arrival rate increases from 60K to 80K, the cost difference between the two offloading strategies gradually diminishes. When the arrival rate exceeds 80K, both one and two-hop offloading demonstrate similar costs, as the fog tier's service rate is reached, and it stops receiving requests from upper tiers. Unidirectional and bidirectional offloading results are similar when changing the arrival rate at the fog, edge, and cloud. Unidirectional upward offloading consistently yields the worst results because it can only offload requests to a higher tier, where the computation cost is higher. On the other hand, the cost of unidirectional downward offloading and bidirectional one-hop offloading are mostly similar, as both approaches enable offloading from a higher-tier node to a lower-tier node for low-cost computation. However, bidirectional offloading stands out with nearly 20% lower cost than one-hop offloading, especially during low traffic arrival rates. As the traffic increases, the gap between the two offloading strategies narrows down. Due to space constraints, detailed results are omitted from the paper.

6.3.2 Workload: unidirectional versus bidirectional

Figure 10 illustrates one-hop offloading scenarios for bidirectional and unidirectional upward and downward offloading. The three bars (left, middle, and right) represent results when requests are received by the cloud, edge, and fog, respectively. When a request reaches the cloud, bidirectional and unidirectional downward offloading predominantly offload most requests to the edge tier for low-cost computation, whereas unidirectional upward offloading processes all requests at the cloud. Upon receiving a request at the edge, bidirectional and unidirectional downward offloading mostly direct requests to the fog tier, with a

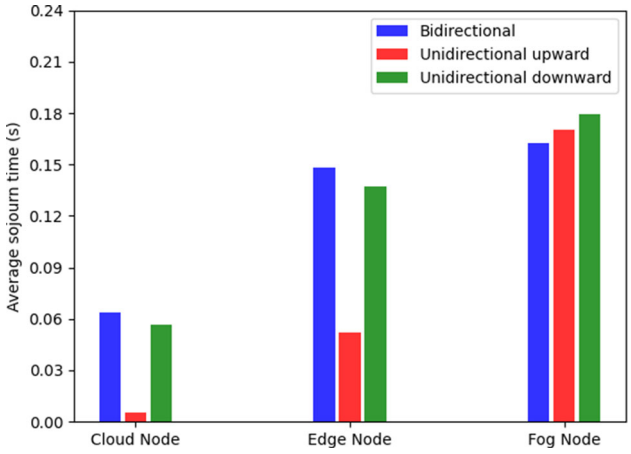


Fig. 11 Average sojourn time of bidirectional and unidirectional offloading

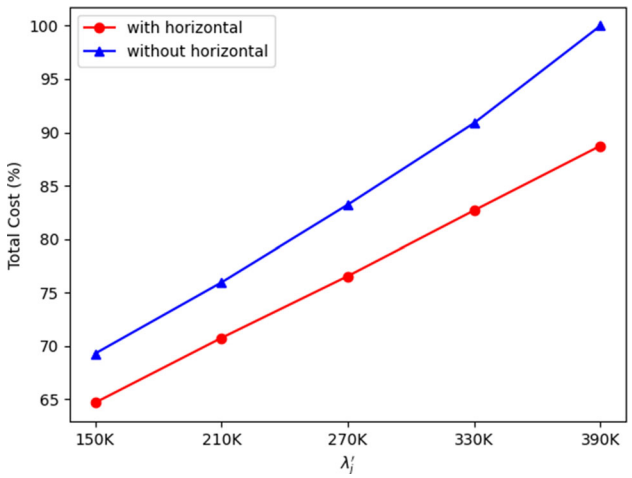


Fig. 12 Total cost with change in the arrival rate of Edge

few being offloaded to adjacent edges or processed by the receiving edge. In contrast, unidirectional upward offloading processes the majority of requests either internally or offloads them to adjacent edges. Both bidirectional and unidirectional upward offloading scenarios exhibit a very low percentage of requests being offloaded from the edge to the cloud. When the fog receives requests for low-cost computation, most or all requests are processed within itself. A small percentage of requests may be offloaded to the edge in bidirectional and unidirectional upward offloading.

6.3.3 Average sojourn time: unidirectional versus bidirectional

Figure 11 presents the average sojourn time for bidirectional and unidirectional offloading. In bidirectional offloading and unidirectional downward offloading, where computation costs in the lower tier are lower than in the higher tier, the average sojourn time remains nearly the same across all cases. Notably, unidirectional upward offloading minimizes the average sojourn time when the cloud receives the request. In this scenario, most requests are processed by the edge itself, with only a small number being offloaded to the cloud. Conversely, in unidirectional downward offloading, requests are processed by the fog, and due to the faster processing speed of an edge compared to a fog, the sojourn time is reduced in unidirectional upward offloading. When a fog receives a request, most requests are processed within the fog for low cost, with only a small number being offloaded to the edge or cloud. Consequently, the sojourn time is nearly similar for all unidirectional and bidirectional offloading scenarios.

6.4 With versus without horizontal offloading analysis

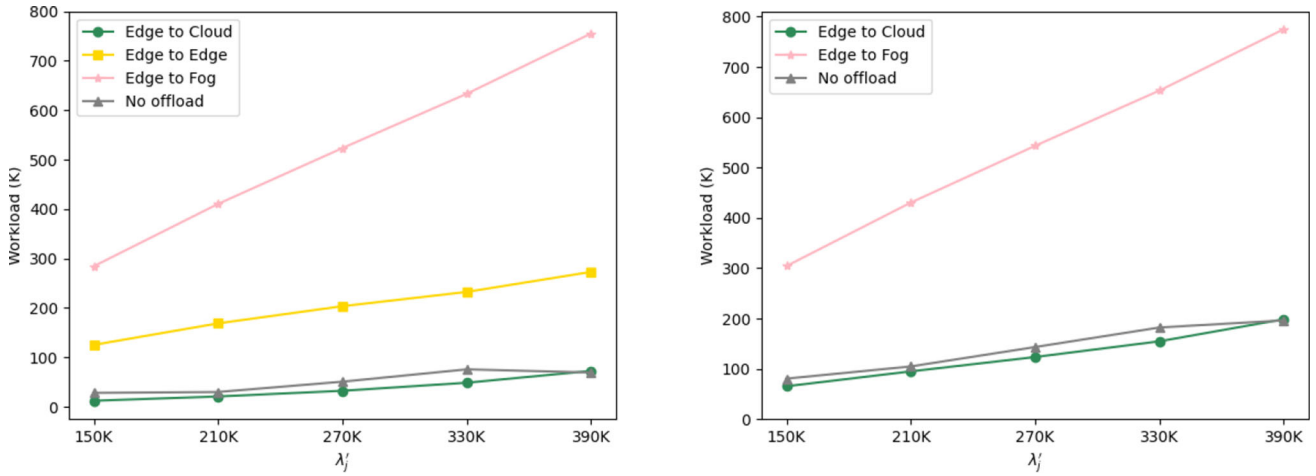
In comparing the performance of architectures with and without horizontal offloading, we restricted our analysis to one-hop offloading.

6.4.1 Total cost: with versus without

Figure 12 shows the result of w/ and w/o horizontal offloading architectures when there is a change in the arrival rate in the edge tier. The results show that in all cases, w/ horizontal offloading architecture will always have a lower average cost than w/o horizontal offloading. At an arrival rate of 150K/sec, the w/ horizontal offloading architecture can reduce the average cost by about 12%. As the arrival rate increases, the difference between the two continues to grow because the edge can offload requests to adjacent edges for processing, benefiting from their lower computation costs compared to the cloud. Additionally, fogs have a minimal service rate, making them less capable of handling large requests effectively.

6.4.2 Workload: with versus without

The offloading results for requests received by the edge of w/ horizontal and w/o horizontal offloading are shown in Fig. 13a, b, respectively. The results in both figs show that both w/ horizontal and w/o horizontal offloading architecture will try to offload a request to the fog tiers for low-cost computation. In Fig. 13a, due to w/ horizontal offloading, the edge will offload more requests to other edges, compared to Fig. 13b, where more requests will offload to the cloud since there is no horizontal offloading.



(a) For w/ horizontal offloading (b) For w/o horizontal offloading

Fig. 13 Offloading result of w/ horizontal offloading and w/o horizontal offloading

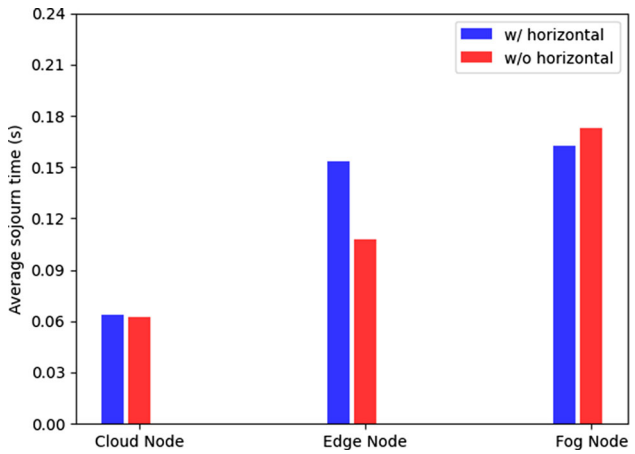


Fig. 14 Average sojourn time of w/ horizontal offloading and w/o horizontal offloading

6.4.3 Average sojourn time: with versus without

Figure 14 shows the average sojourn time of w/ horizontal and w/o horizontal offloading. When requests are received by the cloud, the average sojourn times are nearly identical in both cases because, in both scenarios, requests are offloaded to the edges. Similarly, when the fog receives a request, the average sojourn times are nearly equal in both cases as the fog itself processes the requests. However, when an edge receives a request, the average sojourn time is longer in the w/ horizontal offloading scenario. In this scenario, requests can be offloaded to both adjacent edges and the cloud. In contrast, in the w/o horizontal offloading scenario, an edge can only offload a request to the cloud but not to other edges, resulting in faster processing times. As a consequence, the average sojourn time is higher in the w/ horizontal offloading compared to the w/o horizontal offloading.

7 Conclusions

In this paper, we have proposed a generic architecture for a cloud-edge-fog federated system that provides two-hop, horizontal, and bidirectional vertical offloading. We identified an optimization problem of minimizing the total cost with latency as a constraint and derived a simulated annealing algorithm to solve it. Our results show that with two-hop offloading, cost savings of approximately 10–20% can be achieved compared to one-hop offloading. However, two-hop offloading introduces additional latency. Therefore, for tasks with loose latency constraints, two-hop offloading offers relatively lower computation costs than one-hop offloading. On the other hand, for tight latency jobs, one-hop offloading is more suitable. We also found that the horizontal offloading architecture can save nearly 12% of total cost compared to that without horizontal offloading, and the architecture with bidirectional vertical offloading can save nearly 20% of total cost compared to that with only unidirectional vertical offloading.

The results presented here are contingent upon given parameter settings, which may be subject to change in different scenarios. Moving forward, we aim to conduct further investigations to understand how various system parameters influence the outcomes. Additionally, we plan to leverage machine learning-based approaches [36, 37] capable of effectively mapping any input parameters to achieve the desired output.

Author Contributions All authors contributed equally to the research. B-SL, BK, and C-YC wrote the main manuscript text. B-SL prepared all the figures. All authors reviewed the manuscript.

Funding This work was supported by the National Science and Technology Council (NSTC), Taiwan, under Grant 109-2221-E-011-104-MY3.

Declarations

Conflict of interest The authors declare no competing interests.

Appendix A: Two-hop offloading

A.1: Communication latency

Two-hop offloading can be divided into first-hop and second-hop. For example, in an offloading scenario from $f_{j,i}$ through e_j to C; first-hop is from $f_{j,i}$ to e_j, and second-hop is from e_j to C. Since two-hop offloading is very similar to one-hop offloading, all offloading cases of one-hop are also available for two-hop offloading, along with five extra offloading options: (1) from $f_{j,i}$ through e_j to C, (2) from $f_{j,i}$ through e_j to $e_{j'}$, (3) from $f_{j,i}$ through e_j to $f_{j,i'}$, (4) from e_j through $e_{j'}$ to $f_{j',i}$, and (5) from C through e_j to $f_{j,i}$. Since the estimation of first-hop communication latency is the same as one-hop offloading, the calculation of second-hop communication latency is discussed here. Let D_C be the second-hop of communication latency from $f_{j,i}$ to C, which can be represented as

$$D_C = \frac{1}{r'_j - \lambda_{j,i}\beta_{j,i}\beta'_j}, \quad r'_j \geq \lambda_{j,i}\beta_{j,i}\beta'_j, \quad \forall j, i, \tag{A1}$$

where $\lambda_{j,i}\beta_{j,i}\beta'_j$ is the request rate offload from $f_{j,i}$ to C via e_j. Let $D_{j'}$ be the second-hop of communication latency from $f_{j,i}$ to $e_{j'}$, which can be represented as

$$D_{j'} = \frac{1}{r'_{j,j'} - \lambda_{j,i}\beta_{j,i}\beta^{\star}_{j,j'}}, \quad j \neq j', \quad r'_{j,j'} \geq \lambda_{j,i}\beta_{j,i}\beta^{\star}_{j,j'}, \quad \forall j, j', i, \tag{A2}$$

where $\lambda_{j,i}\beta_{j,i}\beta^{\star}_{j,j'}$ is the request rate offload from $f_{j,i}$ to $e_{j'}$ via e_j. Let $D_{j,i'}$ be the second-hop of communication latency from $f_{j,i}$ to $f_{j,i'}$, which can be represented as

$$D_{j,i'} = \frac{1}{r_{j,i'} - \lambda_{j,i}\beta_{j,i}\beta'_{j,i'}}, \quad i \neq i', \quad r_{j,i'} \geq \lambda_{j,i}\beta_{j,i}\beta'_{j,i'}, \quad \forall j, i, i', \tag{A3}$$

where $\lambda_{j,i}\beta_{j,i}\beta'_{j,i'}$ is the request rate offload from $f_{j,i}$ to $f_{j,i'}$ via e_j. Let $D'_{j',i}$ be the second-hop of communication latency from e_j to $f_{j',i}$, which can be represented as

$$D'_{j',i} = \frac{1}{r_{j',i} - \lambda'_j\beta^{\star}_{j,j'}\beta'_{j',i}}, \quad j \neq j', \quad r_{j',i} \geq \lambda'_j\beta^{\star}_{j,j'}\beta'_{j',i}, \quad \forall j, j', i, \tag{A4}$$

where $\lambda'_j\beta^{\star}_{j,j'}\beta'_{j',i}$ is the request rate offload from e_j to $f_{j',i}$ via e_j. Let $D''_{j,i}$ be the second-hop of communication latency from C to $f_{j,i}$, which can be represented as

$$D''_{j,i} = \frac{1}{r_{j,i} - \lambda''\beta''_j\beta'_{j,i}}, \quad r_{j,i} \geq \lambda''\beta''_j\beta'_{j,i}, \quad \forall j, i, \tag{A5}$$

where $\lambda''\beta''_j\beta'_{j,i}$ is the request rate offload from C to $f_{j,i}$ via e_j.

A.2: Computation latency

The computation latency of two-hop offloading is also based on the M/M/c queuing model, and the methods of $l_{j,i}$, l'_j, and l'' are same as (9), (10), and (11), respectively. However, since we are considering the second-hop, the calculation of $R_{j,i}$, R'_j, and R'' are not same as given in (6), (7), and (8). $R_{j,i}$ in two-hop offloading can be represented as

$$\begin{aligned} R_{j,i} = {} & \lambda_{j,i} - \lambda_{j,i}\beta_{j,i} \\ & + \lambda'_j\beta'_{j,i} + \sum_{i'=1, i'\neq i}^{n_j} \lambda_{j,i'}\beta_{j,i}\beta'_{j,i} \\ & + \sum_{j'=1, j'\neq j}^{m} \lambda'_{j'}\beta^{\star}_{j',j}\beta'_{j,i} + \lambda''_j\beta''_j\beta'_{j,i}, \end{aligned} \tag{A6}$$

where $\lambda_{j,i'}\beta_{j,i'}\beta'_{j,i}$ is the request rate offloaded from $f_{j,i'}$ to $f_{j,i}$, $\lambda'_{j'}\beta^{\star}_{j',j}\beta'_{j,i}$ is the request rate offloaded from $e_{j'}$ to $f_{j,i}$, and $\lambda''\beta''_j\beta'_{j,i}$ is the request rate offloaded from C to $f_{j,i}$. The R'_j in two-hop offloading can be represented as

$$\begin{aligned} R'_j = {} & \lambda'_j - \lambda'_j\beta'_j \\ & + \lambda''\left(1 - \sum_{i=1}^{n_j}\beta'_{j,i}\right)\beta''_j - \sum_{i=1}^{n_j}\lambda'_j\beta'_{j,i} \\ & + \sum_{i=1}^{n_j}\lambda_{j,i}\left(1 - \beta'_j - \sum_{i'=1, i'\neq i}^{n_j}\beta'_{j,i'} - \sum_{j'=1, j'\neq j}^{m}\beta^{\star}_{j,j'}\right)\beta_{j,i} \\ & - \sum_{j'=1, j'\neq j}^{m}\lambda'_j\beta^{\star}_{j,j'} \\ & + \sum_{j'=1, j'\neq j}^{m}\lambda'_{j'}\left(1 - \sum_{i=1}^{n_j}\beta'_{j,i}\right)\beta^{\star}_{j',j} \\ & + \sum_{j'=1, j'\neq j}^{m}\sum_{i=1}^{n_j}\lambda_{j',i}\beta_{j',i}\beta^{\star}_{j',j}, \end{aligned} \tag{A7}$$

where $\lambda_{j',i}\beta_{j',i}\beta^{\star}_{j',j}$ is the request rate offloaded from $f_{j',i}$ to e_j. $\lambda''\left(1 - \sum_{i=1}^{n_j}\beta'_{j,i}\right)\beta''_j$, $\lambda_{j,i}(1 - \beta'_j - \sum_{i'=1, i'\neq i}^{n_j}\beta'_{j,i'} - \sum_{j'=1, j'\neq j}^{m}\beta^{\star}_{j,j'})\beta_{j,i}$, and $\lambda'_{j'}\left(1 - \sum_{i=1}^{n_j}\beta'_{j,i}\right)\beta^{\star}_{j',j}$ are request rate received by e_j from C, $f_{j,i}$, and $e_{j'}$, respectively.

The R'' in two-hop offloading can be represented as

$$R'' = \lambda'' - \sum_{j=1}^{m} \lambda''\beta''_j + \sum_{j=1}^{m} \lambda'_j\beta'_j + \sum_{j=1}^{m}\sum_{i=1}^{n_j} \lambda_{j,i}\beta_{j,i}\beta'_j, \tag{A8}$$

where $\lambda_{j,i}\beta_{j,i}\beta'_j$ is the request rate offloaded from $f_{j,i}$ to C.

A.3: Communication cost

Here, we calculated the communication cost between tiers. Since two-hop offloading has an extra second-hop offloading, its estimation is slightly different from the one-hop offloading. The $S'_{C,E}$ in two-hop offloading can be represented as

$$S'_{C,E} = s'_{C,E}\left(\sum_{j=1}^{m} \lambda''\beta''_j + \sum_{j=1}^{m} \lambda'_j\beta'_j + \sum_{j=1}^{m}\sum_{i=1}^{n_j} \lambda_{j,i}\beta_{j,i}\beta'_j\right), \tag{A9}$$

where $\lambda_{j,i}\beta_{j,i}\beta'_j$ is the second-hop of the offloading from $f_{j,i}$ to C. The $S'_{E,E'}$ in two-hop offloading can be represented as

$$S'_{E,E'} = s'_{E,E'}\left(\sum_{j=1}^{m}\sum_{j'=1,j'\neq j}^{m} \lambda'_j\beta^{\star}_{j,j'} + \sum_{j=1}^{m}\sum_{i=1}^{n_j}\sum_{j'=1,j'\neq j}^{m} \lambda_{j,i}\beta_{j,i}\beta^{\star}_{j,j'}\right), \tag{A10}$$

where $\lambda_{j,i}\beta_{j,i}\beta^{\star}_{j,j'}$ is the second-hop of the offloading from $f_{j,i}$ to $e_{j'}$. The $S_{E,F}$ in two-hop offloading can be represented as

$$\begin{aligned} S_{E,F} = s_{E,F}\Bigg(&\sum_{j=1}^{m}\sum_{i=1}^{n_j} \lambda'_j\beta'_{j,i} + \sum_{j=1}^{m}\sum_{i=1}^{n_j} \lambda_{j,i}\beta_{j,i} \\ &+ \sum_{j=1}^{m}\sum_{i=1}^{n_j}\sum_{i'=1,i'\neq i}^{n_j} \lambda_{j,i}\beta_{j,i}\beta'_{j,i'} \\ &+ \sum_{j=1}^{m}\sum_{j'=1,j'\neq j}^{m}\sum_{i=1}^{n_j} \lambda'_j\beta^{\star}_{j,j'}\beta'_{j',i} \\ &+ \sum_{j=1}^{m}\sum_{i=1}^{n_j} \lambda''\beta''_j\beta'_{j,i}\Bigg), \end{aligned} \tag{A11}$$

where $\lambda_{j,i}\beta_{j,i}\beta'_{j,i'}$ is the second-hop of the offloading from $f_{j,i}$ to $f_{j,i'}$, $\lambda'_j\beta^{\star}_{j,j'}\beta'_{j',i}$ is the second-hop of the offloading from e_j to $f_{j',i}$, and $\lambda''\beta''_j\beta'_{j,i}$ is the second-hop of the offloading from C to $f_{j,i}$.

A.4: Computation cost

The evaluation of computing cost in two-hop offloading is the same as one-hop offloading as shown in (15) for the cloud tier, (16) for the edge tier, and (17) for the fog tier.

A.5: Objective and constraints

The objective function in two-hop offloading is the same as one-hop offloading, as shown in (18). The constraint of two-hop offloading will be more complicated because of the second-hop compared to the one-hop offloading, and the objective function must meet the (22)–(27), along with the following constraints.

$$\hat{\beta}l'' + \sum_{j=1}^{m}\beta''_j\left(1 - \sum_{i=1}^{n_j}\beta'_{j,i}\right)\left(l'_j + T''_j\right) + \sum_{j=1}^{m}\sum_{i=1}^{n_j}\beta''_j\beta'_{j,i}\left(l_{j,i} + D''_{j,i}\right) \leq \mathcal{L}_{max}, \tag{A12}$$

$$\begin{aligned} &\hat{\beta}_j l'_j + \beta'_j\left(l'' + T'_j\right) + \sum_{i=1}^{n_j}\beta'_{j,i}\left(l_{j,i} + T'_{j,i}\right) \\ &+ \sum_{j'=1,j'\neq j}^{m}\beta^{\star}_{j,j'}\left(1 - \sum_{i=1}^{n_j}\beta'_{j',i}\right)\left(l'_{j'} + T^{\star}_{j,j'}\right) \\ &+ \sum_{j'=1,j'\neq j}^{m}\sum_{i=1}^{n_j}\beta^{\star}_{j,j'}\beta'_{j',i}\left(l_{j',i} + D'_{j',i}\right) \leq \mathcal{L}_{max}, \end{aligned} \tag{A13}$$

$$\begin{aligned} &\hat{\beta}_{j,i} l_{j,i} + \beta_{j,i}(1 - \beta'_j - \sum_{i'=1,i'\neq i}^{n_j}\beta'_{j,i'} \\ &- \sum_{j'=1,j'\neq j}^{m}\beta^{\star}_{j,j'})\left(l'_j + T_{j,i}\right) \\ &+ \sum_{i'=1,i'\neq i}^{n_j}\beta_{j,i}\beta'_{j,i'}\left(l_{j,i'} + D_{j,i'}\right) \\ &+ \sum_{j'=1,j'\neq j}^{m}\beta_{j,i}\beta^{\star}_{j,j'}\left(l'_{j'} + D_{j'}\right) \\ &+ \beta_{j,i}\beta'_j\left(l'' + D_C\right) \leq \mathcal{L}_{max}. \end{aligned} \tag{A14}$$

The constraints in (A12), (A13), and (A14) ensure that the total communication latency plus the total computation latency of the cloud, edge, and fog in the case of two-hop offloading do not exceed the maximum latency limit.

An appendix contains supplementary information that is not an essential part of the text itself but which may be helpful in providing a more comprehensive understanding of the research problem or it is information that is too cumbersome to be included in the body of the paper.

References

1. Shi, W., & Dustdar, S. (2016). The promise of edge computing. *Computer, 49*(5), 78–81.
2. Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal, 3*, 637–646.
3. Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012). Fog computing and its role in the internet of things. In *MCC '12* (pp. 9600–9609).
4. Vaquero, L. M., & Rodero-Merino, L. (2014). Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review, 44*(5), 27–32.
5. Liu, L., Chang, Z., Guo, X., Mao, S., & Ristaniemi, T. (2017). Multiobjective optimization for computation offloading in fog computing. *IEEE Internet of Things Journal, 5*(1), 283–294.
6. Dustdar, S., Avasalcai, C., & Murturi, I. (2019). Invited paper: Edge and fog computing: Vision and research challenges. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)* (pp. 9600–9609).
7. Assis, M. R. M., & Bittencourt, L. F. (2016). A survey on cloud federation architectures: Identifying functional and non-functional properties. *Journal of Network and Computer Applications, 72*, 51–71.
8. Aijaz, A., Aghvami, H., & Amani, M. (2013). A survey on mobile data offloading: Technical and business perspectives. *IEEE Wireless Communications, 20*(2), 104–112.
9. Akutsu, K., Phung-Duc, T., Lai, Y.-C., & Lin, Y.-D. (2022). Analyzing vertical and horizontal offloading in federated cloud and edge computing systems. *Telecommunication Systems, 79*(3), 447–459.
10. Chen, X., Jiao, L., Li, W., & Fu, X. (2015). Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking, 24*(5), 2795–2008.
11. Zhang, H., Xiao, Y., Bu, S., Niyato, D., Yu, R., & Han, Z. (2016). Fog computing in multi-tier data center networks: A hierarchical game approach. In *2016 IEEE international conference on communications (ICC)* (pp. 1–6).
12. Lin, Y., Hu, J., Kar, B., & Yen, L. (2019). Cost minimization with offloading to vehicles in two-tier federated edge and vehicular-fog systems. In *2019 IEEE 90th vehicular technology conference (VTC2019-Fall)* (pp. 1–6).
13. Lin, Y., Lai, Y., Huang, J., & Chien, H. (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), 923–933.
14. Wang, N., Varghese, B., Matthaiou, M., & Nikolopoulos, D. S. (2017). Enorm: A framework for edge node resource management. *IEEE Transactions on Services Computing, 13*(6), 1086–1099.
15. Kar, B., Yahya, W., Lin, Y.-D., & Ali, A. (2023). Offloading using traditional optimization and machine learning in federated cloud-edge-fog systems: A survey. *IEEE Communications Surveys & Tutorials*.
16. Zahid, N., Alkhayyat, A., Ismail, M., & Sodhro, A. H. (2022). An effective traffic management approach for decentralized bsns. In *2022 IEEE 96th vehicular technology conference (VTC2022-Fall)* (pp. 1–5). IEEE.
17. Lakhan, A., Sodhro, A. H., Majumdar, A., Khuwuthyakorn, P., & Thinnukool, O. (2022). A lightweight secure adaptive approach for internet-of-medical-things healthcare applications in edge-cloud-based networks. *Sensors, 22*(6), 2379.
18. Chekired, D. A., Khoukhi, L., & Mouftah, H. T. (2018). Industrial iot data scheduling based on hierarchical fog computing: A key for enabling smart factory. *IEEE Transactions on Industrial Informatics, 14*(10), 4590–4602.
19. Thai, M., Lin, Y., Lai, Y., & Chien, H. (2019). Workload and capacity optimization for cloud-edge computing systems with vertical and horizontal offloading. *IEEE Transactions on Network and Service Management, 17*(1), 227–238.
20. Deng, S., Zhang, C., Li, C., Yin, J., Dustdar, S., & Zomaya, A. Y. (2021). Burst load evacuation based on dispatching and scheduling in distributed edge networks. *IEEE Transactions on Parallel and Distributed Systems, 32*(8), 1918–1932.
21. Aburukba, R. O., Landolsi, T., & Omer, D. (2021). A heuristic scheduling approach for fog-cloud computing environment with stationary iot devices. *Journal of Network and Computer Applications, 180*, 102994.
22. Kar, B., Lin, Y., & Lai, Y. (2020). Omni: Omni-directional dual cost optimization of two-tier federated cloud-edge systems. In *2020 IEEE International Conference on Communications (ICC)* (pp. 1–7).
23. Kar, B., Lin, Y.-D., & Lai, Y.-C. (2023). Cost optimization of omni-directional offloading in two-tier cloud-edge federated systems. *Journal of Network and Computer Applications, 215*, 103630.
24. Cao, X., Tang, G., Guo, D., Li, Y., & Zhang, W. (2020). Edge federation: Towards an integrated service provisioning model. *IEEE/ACM Transactions on Networking, 28*(3), 1116–1129.
25. Ascigil, O., Tasiopoulos, A., Phan, T. K., Sourlas, V., Psaras, I., & Pavlou, G. (2021). Resource provisioning and allocation in function-as-a-service edge-clouds. *IEEE Transactions on Services Computing*.
26. Dong, Y., Xu, G., Zhang, M., & Meng, X. (2021). A high-efficient joint 'cloud-edge' aware strategy for task deployment and load balancing. *IEEE Access, 9*, 12791–12802.
27. Tong, L., Li, Y., & Gao, W. (2019). A hierarchical edge cloud architecture for mobile computing. In *35th IEEE international conference on computer communications* (pp. 1–9).
28. Faticanti, F., Savi, M., Pellegrini, F. D., Kochovski, P., Stankovski, V., & Siracusa, D. (2020). Deployment of application microservices in multi-domain federated fog environments. In *2020 international conference on omni-layer intelligent systems (COINS)* (pp. 1–6).
29. Razaq, M. M., Tak, B., Peng, L., & Guizani, M. (2021). Privacy-aware collaborative task offloading in fog computing. *IEEE Transactions on Computational Social Systems*.
30. Sharmin, Z., Malik, A. W., Rahman, A. U., & Noor, R. D. (2020). Toward sustainable micro-level fog-federated load sharing in internet of vehicles. *IEEE Internet of Things Journal, 7*(4), 3614–3622.
31. Yen, L., Hu, J., Lin, Y., & Kar, B. (2020). Decentralized configuration protocols for low-cost offloading from multiple edges to multiple vehicular fogs. *IEEE Transactions on Vehicular Technology, 70*(1), 872–885.
32. Mourad, A., Tout, H., Wahab, O. A., Otrok, H., & Dbouk, T. (2020). Ad hoc vehicular fog enabling cooperative low-latency intrusion detection. *IEEE Internet of Things Journal, 8*(2), 829–843.
33. Kar, B., Shieh, K., Lai, Y., Lin, Y., & Ferng, H. (2021). Qos violation probability minimization in federating vehicular-fogs with cloud and edge systems. *IEEE Transactions on Vehicular Technology, 70*(12), 13270–13280.
34. Angus, I. (2001). An introduction to erlang b and erlang c. *Telemanagement, 187*, 6–8.
35. Haddock, J., & Mittenthal, J. (1992). Simulation optimization using simulated annealing. *Computers & industrial engineering, 22*(4), 387–395.

36. Kar, B., Yahya, W., Lin, Y.-D., & Ali, A. (2022). A survey on offloading in federated cloud-edge-fog systems with traditional optimization and machine learning. arXiv:2202.10628
37. Ahmad, I., Shahabuddin, S., Malik, H., Harjula, E., Leppänen, T., Loven, L., Anttonen, A., Sodhro, A. H., Alam, M. M., Juntti, M., & Ylä-Jääski, A. (2020). Machine learning meets communication networks: Current trends and future challenges. *IEEE Access, 8*, 223418–223460.

Bo-Shiuan Lin received his B.S. degree in computer science from National Taipei University of Education, Taiwan, in 2019, and his M.S. degree in computer science and information engineering from the National Taiwan University of Science and Technology, Taiwan. His research interests include cloud computing, mobile networks, and federation systems.

Binayak Kar is an Assistant Professor of computer science and information engineering at National Taiwan University of Science and Technology (NTUST), Taiwan. He received his Ph.D. degree in computer science and information engineering from the National Central University (NCU), Taiwan in 2018. He was a post-doctoral research fellow in computer science with National Chiao Tung University (NCTU), Taiwan, from 2018 to 2019. His research interests include network softwarization, cloud/edge/fog computing, optimization, queueing theory, machine learning, and cyber security.

Tai-Lin Chin received his B.S. in Computer Science and Information Engineering from the National Chiao-Tung University, Taiwan, in 1995 and his M.S. and Ph.D. in Electrical and Computer Engineering from the University of Wisconsin-Madison in 2004 and 2006, respectively. Since 2007, he has been with the Department of Computer Science and Information Engineering at the National Taiwan University of Science and Technology, where he is currently a Professor. His research interests include wireless networking, cloud/edge computing, sensor networks, and artificial intelligence for networking.

Ying-Dar Lin is Vice President of National Institute of Cyber Security (NICS), and also a Chair Professor of computer science at National Yang Ming Chiao Tung University (NYCU), 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 2007-2008, CEO at Telecom Technology Center, Taiwan, during 2010-2011, and Vice President of National Applied Research Labs (NARLabs), Taiwan, during 2017-2018. He cofounded L7 Networks Inc. in 2002, later acquired by D-Link Corp. He also founded and directed Network Benchmarking Lab (NBL) from 2002, which reviewed network products with real traffic and automated tools, also an approved test lab of the Open Networking Foundation (ONF), and spun-off O'Prueba Inc. in 2018. His recent research interests include machine learning for cybersecurity, wireless communications, network softwarization, and mobile edge computing. His work on multi-hop cellular was the first along this line, and has been cited over 1000 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 (2014-2017), ONF Research Associate (2014-2018), and received K. T. Li Breakthrough Award in 2017 and Research Excellence Award in 2017 and 2020. He has served or is serving on the editorial boards of many IEEE journals and magazines, including Editor-in-Chief of IEEE Communications Surveys and Tutorials (COMST) with impact factor increased from 9.22 to 25.3 during his term (2017-2020). He published a textbook, Computer Networks: An Open Source Approach, with Ren-Hung Hwang and Fred Baker (McGraw-Hill, 2011).

Chung-Yueh Chen received his B.S. degree in mechanical engineering from the National Taiwan University of Science and Technology, Taiwan, in 2020, and his M.S. degree in computer science and information engineering from the National Taiwan University of Science and Technology, Taiwan, in 2023. His research interests cloud computing, edge computing, sensor networks, and deep reinforcement learning.