SDN-Assisted Mobile Edge Computing for Collaborative Computation Offloading in Industrial Internet of Things

Chaogang Tang, *Member, IEEE*, Chunsheng Zhu, *Member, IEEE*, Ning Zhang, *Senior Member, IEEE*, Mohsen Guizani, *Fellow, IEEE*, and Joel J. P. C. Rodrigues, *Fellow, IEEE*

***Abstract*—Mobile edge computing (MEC) can provision augmented computational capacity in proximity so as to better support Industrial Internet of Things (IIoT). Tasks from the IIoT devices can be outsourced and executed at the accessible computational access point (CAP). This computing paradigm enables the computing resources much closer to the IIoT devices, and thus satisfy the stringent latency requirement of the IIoT tasks. However, existing works in MEC that focus on task offloading and resource allocation seldom consider the load balancing issue. Therefore, load balance aware task offloading strategies for IIoT devices in MEC are urgently needed. In this article, software-defined network (SDN) technology is adopted to address this issue, since the rule-based forwarding policy in SDN can help determine the most suitable offloading path and CAP for undertaking the computation. To this end, we formulate an optimization problem to minimize the response latency in the proposed SDN-assisted MEC architecture. A greedy algorithm is put forward to obtain the approximate optimal solution in polynomial time. Simulation has been carried out to evaluate the performance of the proposed approach. The simulation results reveal that our approach outstands other approaches in terms of the response latency.**

***Index Terms*—Computation offloading, Industrial Internet of Things (IIoT), load balancing, mobile edge computing (MEC), response latency optimization, software-defined network (SDN).**

Manuscript received 24 December 2021; revised 14 May 2022; accepted 1 July 2022. Date of publication 13 July 2022; date of current version 21 November 2022. This work was supported in part by the Chinese National Research Fund (NSFC) Key Project under Grant 61532013 and Grant 61872239; in part by FCT/MCTES through National Funds and when applicable Co-Funded EU Funds under Project UIDB/50008/2020; and in part by the Brazilian National Council for Scientific and Technological Development—CNPq under Grant 313036/2020-9. *(Chaogang Tang and Chunsheng Zhu contributed equally to this work.) (Corresponding author: Chunsheng Zhu.)*

Chaogang Tang is with the School of Computer Science and Technology and the Mine Digitization Engineering Research Center, Ministry of Education, China University of Mining and Technology, Xuzhou 221116, China (e-mail: cgtang@cumt.edu.cn).

Chunsheng Zhu is with the College of Big Data and Internet, Shenzhen Technology University, Shenzhen 518118, China (e-mail: chunsheng.tom.zhu@gmail.com).

Ning Zhang is with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON N9B 3P4, Canada (e-mail: ning.zhang@uwindsor.ca).

Mohsen Guizani is with the Machine Learning Department, Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE (e-mail: mguizani@ieee.org).

Joel J. P. C. Rodrigues is with the College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266555, China, and also with the Instituto de Telecomunicações, 3810-193 Aveiro, Portugal (e-mail: joeljr@ieee.org).

Digital Object Identifier 10.1109/JIOT.2022.3190281

I. Introduction

THE INDUSTRIAL Internet of Things (IIoT), benefitting from the Internet of Things (IoT), has been playing a key role in coordination, orchestration, and management of manufacturing resources [1]. By integrating the industrial communication, computing, and controlling technologies, IIoT enables ubiquitous connections among robots, actuators, sensors, and wearable equipment terminals. In this context, these intelligent IIoT devices at industry locations generate the massive amount of data and a wide variety of tasks every second. Some data and tasks, dedicated to real-time monitoring by perceiving physical resources, need prompt data analysis and real-time decision making [2]. The IIoT devices in general have constrained computing capabilities, due to small physical sizes. Thus, computation offloading is the primary means of task execution for these intelligent devices. However, the sensor-to-cloud computing paradigm, which provides computational resources for these IIoT devices at a centralized data center, incurs long transmission delay, thus making it inappropriate for these time-sensitive IIoT tasks.

To overcome the above issue, mobile edge computing (MEC) [3] can be considered as a promising approach, since it brings the computational resources to the network edges and thus provides computing services in proximity to the IIoT devices. In particular, integration of MEC and an access point (AP) can empower AP with computing and storage resources, giving birth to an accessible computational AP (CAP). CAP can undertake the processing of IIoT tasks when they are offloaded from industry locations such as smart factory. The response latency can thus be substantially reduced. As a result, computation offloaded and undertaken in MEC caters to the strict latency requirement of IIoT devices. However, strategies on task offloading and resource allocation in MEC seldom consider load balancing, which on the other hand, is an important issue in the industrial field. The surge in IIoT tasks can cause tremendous pressure on the limited computing capabilities of CAP. Moreover, it is possible that one CAP is overwhelmed by huge computation while another adjacent CAP has sufficient resources but fails to participate in the computation owing to limited wireless coverage. It is of vital importance to avoid such a situation. In other words, a load balance aware task offloading strategy for IIoT devices in MEC is urgently needed.

2327-4662 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on March 01,2023 at 12:50:15 UTC from IEEE Xplore. Restrictions apply.

We can solve the above load imbalance among CAPs by further enhancing the intelligence of the industrial networks. Specifically, software-defined network (SDN) [4] can provide an efficient solution to this issue by enabling the computation loads to be evenly distributed among CAPs. In general, the traditional networks have integrated and modularized the control and data planes in the network elements (e.g., routers and switches), so changing the forwarding policy requires the manual reconfiguration of these network elements. Such operations are prohibitively costly when the number of network elements explosively increases. In contrast, SDN decouples the control plan from the data plan, and thus increases the network intelligence and flexibility by programmatically configuring the network traffic. When it comes to the computation offloading for IIoT tasks in MEC, rule-based forwarding policy in SDN can help determine the most suitable offloading path and CAP for undertaking the computation [5]. By virtue of a global network view, the SDN controller on the one hand takes charge of gathering state information including computing capabilities and waiting queues, and on the other hand, makes decisions on the selection of the offloading path and destination CAP.

In this article, we put forward an SDN-assisted MEC architecture, where IIoT tasks are offloaded and executed at other CAP with the assistance of SDN, with the purposes of: 1) balancing the computation loads among CAPs and 2) reducing the response latency for the time-critical IIoT tasks. In particular, the major contributions of this article are summarized as follows.

1) In view of the stringent latency requirement of the tasks generated by the IIoT devices, an SDN-assisted MEC architecture is proposed in this article, in the hope to address the workload imbalance and further reduce the response latency. To this end, the SDN controller makes decisions on selecting lightly loaded CAP as the destination CAP for the task execution.
2) An optimization problem is given to minimize the response latency for all the IIoT tasks in the industrial networks. In the meanwhile, we take into account the constraints such as the energy consumptions of each CAP for the task execution, and the computing capabilities of each CAP. It takes exponential time to obtain the optimal solution, which is prohibitively costly in reality especially considering the time-sensitive characteristics of the tasks. We propose a greedy approach for solving this problem and hope to obtain the approximate solution in polynomial time.
3) Extensive simulation has been conducted for evaluating the proposed approach. In comparison to other approaches, the simulation results reveal that the proposed approach outstands them in terms of response latency.

The remainder of this article is organized as follows. We survey some related works in Section II. An SDN-assisted MEC architecture is proposed in Section III. The problem is described in Section IV, followed by the problem formulation in Section V. A task offloading strategy is put forward in Section VI. The simulation settings and results analysis are provided in Section VII, and the conclusion is drawn in Section VIII.

II. Related Works

There are lots of works in edge computing which focus on task offloading from different angles such as [6]–[9]. In this section, the related works in edge computing are investigated that leverage the SDN technology for various purposes, such as the optimization of latency and energy consumption.

It becomes increasingly popular that the tasks generated by the mobile devices are outsourced and executed outside the devices themselves, as tasks are often computationally intensive and the mobile devices have limited computing capabilities. For instance, Chalapathi *et al.* [10] strived to address the task assignment problem in a cloudlet-based network. The performance of the multicloudlet network is enhanced by a wireless SDN network. In particular, the SDN network can handle the task offloading requests efficiently, so as to minimize the response latency for mobile devices. Liu *et al.* [11] investigated the cooperative data sharing and forwarding in vehicular networks, and they formulated this cooperative data scheduling problem to optimize the number of vehicles which can obtain the data. In particular, they adopted the SDN technology to solve this problem. In the context of the fog computing, the fog node provides computing resources to the devices. However, one fog node may be overwhelmed by a massive number of task offloading requests. To overcome this issue, Phan *et al.* [12] leveraged the SDN technology to realize a collaborative task offloading in fog computing. They tried to choose an optimal offloading node based on SDN technology.

Misra and Bera [5] proposed a scheme called Soft-VAN, which supports the mobility-aware task offloading. Furthermore, they strived to optimize the response latency in a software-defined vehicular networks. It is well known that some IIoT applications, such as emergency alert relay, need to maintain the newly delivered packets, which requires high reconfigurability. Balasubramanian *et al.* [13] put forward an online algorithm to jointly optimize the assignment, delivery, and admission control, which leverages the SDN technology to improve the management of the networks. The critical IIoT applications have posed tremendous pressure on the traditional Internet architecture, because of the strict Quality-of-Service (QoS) requirements. Therefore, Naeem *et al.* [14] put forward an SDN-enabled parallel routing framework, which leverages GPU to optimize the IIoT applications from the QoS viewpoint. In particular, three different levels of QoS, exemplified by the smart healthcare traffic, are included in the framework. They are loss-sensitive, delay-sensitive, and jitter-sensitive. In view of extensive attention paid to the vehicular networks and vehicular edge computing [15], researchers try to apply SDN technologies to this newly emergent computing paradigm. For example, Mahmood *et al.* [16] surveyed the state-of-the-art software-defined heterogeneous vehicular networking architecture.

A surge in the number of smart devices, such as user terminals and IoT devices, has posed tremendous pressure over the service providers. More attention has been paid to

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on March 01,2023 at 12:50:15 UTC from IEEE Xplore. Restrictions apply.

QoS provisioning, i.e., service providers strive to satisfy the diverse QoS requirements, such as latency and networking resources from the devices [17]. In this context, an SDN-assisted and QoS-guaranteed routing algorithm is required for solving delay constrained least cost (DCLC) problem. Particularly, BinSahaq *et al.* [18] put forward an enhanced Lagrange relaxation-based aggregated cost (LARAC) algorithm to solve this NP-hard DCLC problem. They further filter nonuseful Dijkstra calls leveraging the stop criteria. Simulation results have proven its advantages over other existing routing approaches, such as LARAC and BiLAD algorithms.

Other works, such as [19]–[23], have also concentrated their attention on the combination of both the networking and computing paradigms. Due to the little attention to the application of the SDN technology in the automobile industry, Deng *et al.* [19] have reviewed works which combine SDN technologies with vehicular networks for driving safety, and further designed edge-up software-defined networking architectures, especially for real-time control function. Similarly, to enhance the performance of vehicle ad hoc network (VANET), SDN and named-data networking are incorporated into VANET in [21]. Specifically, SDN takes advantage of its global view in the entire network, and enables IP addressing to be resolved easily in named-data networking. Furthermore, policy-based bifold classifier can be used for packet classification.

A vehicular task offloading framework was proposed in [24] where SDN-enabled UAV is applied for enhancing the performance of the framework. In particular, they aim to minimize the total cost for vehicular tasks and applications. In parallel, there are several works that have applied SDN technologies to the IoMT networks for multiple purposes. For instance, Khan and Akhunzada [25] put forward an efficient algorithm and highly scalable framework to detect the sophisticated IoMT malwares, by combining the deep learning with SDN technologies. The evaluation results reveal that this scheme outstands others in terms of the detection accuracy, fairness, and efficiency. On the other hand, IoMT incorporates the intelligence into IoMT devices, with the purpose of detecting various diseases, which has become a hot research topic in the past few years. Taking the task for detecting the arrhythmia as an example, it usually utilizes the electrocardiogram (ECG) trace as the study object. However, manual feature extraction and noise-filtering often make existing approaches inefficient. To tackle this issue, Sakib *et al.* [26] put forward a deep learning-based lightweight arrhythmia classification approach leveraging deep learning technology (e.g., 1-D convolutional neural network architecture). Particularly, by single-lead ECG trace, the approach can avoid tedious operations, such as noise-filtering and feature extraction. The simulation has shown that their approach is much better than other existing approaches, such as K-nearest neighbor and random forest.

It shall be noted that the ubiquitous connections and communications among IoT device can bring about multivector malware attacks. Therefore, it is of vital importance to prevent the IoT industrial from a variety of cyber threats and attacks. For instance, one SDN-enabled Internet of Medical Things framework is proposed in [27], which tries to adopt hybrid neural networks to detect the multivector malware botnets. For example, this kind of neural networks can combine the advantages of both CNN and Cuda Deep LSTM for threat and attack detection.

The aforementioned works combine the SDN technology with MEC, cloud computing, and vehicular edge computing, so as to optimize the task offloading with multiple purposes. Different from these works, we, in this article, propose a horizontal cooperation among CAPs with the aid of the SDN technology. By doing this, we can address the issue of workload imbalance and further reduce the response latency for all the tasks generated by IIoT devices.

III. Proposed SDN-Assisted MEC Architecture

Fig. 1 illustrates the SDN-assisted MEC architecture for IIoT in this article. In this architecture, the industrial area is divided into several subregions also called communities. Each community is composed of numerous scattered IIoT devices and at least one large intelligent equipment manufactory, such as a smart factory and a smart grid. The IIoT devices usually include robots, wearable terminals, sensors, and actuators. Each community is served by one CAP, which means that the tasks generated by the IIoT devices in the community can be offloaded to the serving CAP for execution. CAPs are interconnected via the fiber-optic networks. Meanwhile, the CAPs can access the resources in cloud center using the backhaul networks. In contrast to CAPs, the cloud center can provide unlimited computing and storage resources. Thus, the cloud computing is specially suitable for tasks demanding complex computations and massive data storage. In particular, for the industrial field, some tasks that are time-insensitive can be offloaded and executed in the industrial cloud.

To overcome the workload imbalance among CAPs and further improve the response latency for the IIoT tasks, the SDN technology is adopted in this architecture to enhance the intelligence and flexibility of the industrial networks. Owing to its friendly programmability for network protocols, the decoupled control plane can realize more efficient forwarding policy by the OpenFlow Protocol [28]. CAPs are deployed with the controller interface for establishing the communication links between CAPs and the SDN controller. The controller, as the core component of SDN, takes charge of data transmission control. To be more specific, CAPs receive the task offloading requests and forward them to the controller that will make decisions on destination CAPs for the tasks and the transmission path if necessary.

The task offloading requests mainly include the task information on the input data, the required workloads (e.g., CPU cycles), and the delay constraint. On the other hand, each CAP periodically reports its status into the SDN subsystem. The status usually includes information on computational resources, task queue, latency, and bandwidth. Such information can be gathered, handled, and analyzed by a module called edge orchestrator as shown in Fig. 1, with the purpose of assisting the SDN controller for decision making.

As shown in the figure, three specific functions for offloading are abstracted and performed in the proposed SDN-assisted

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on March 01,2023 at 12:50:15 UTC from IEEE Xplore. Restrictions apply.

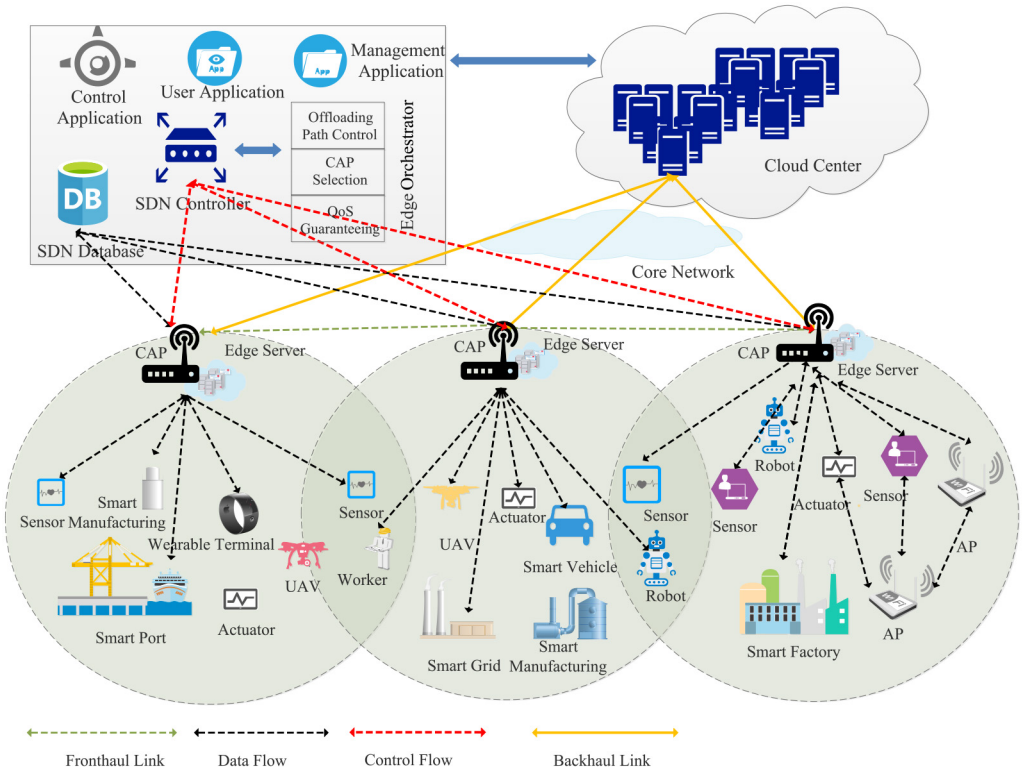


Fig. 1. Proposed SDN-assisted MEC architecture for IIoT tasks offloading.

MEC architecture, i.e., offloading path control, destination CAP selection, and QoS guaranteeing. The offloading path control usually determines the migration path of one task when it is performed by a lightly loaded CAP instead of its serving CAP. The destination CAP selection is to determine the lightly loaded CAP for accomplishing the migrated task. QoS guaranteeing is to satisfy the nonfunctional requirements of the tasks, such as deadline, reliability, and success rate. In the meanwhile, it can also improve the Quality of Experience (QoE) for the resource requestors. Note that the SDN controller needs to collaborate with the edge orchestrator to fulfil these functions.

IV. Problem Description

The considered system model is depicted in Fig. 1, which includes a set of CAPs, a set of IIoT devices scattered across different communities and served by respective CAPs, the SDN subsystem, and the remote cloud center. Let $\mathcal{C} = \{c_1, \ldots, c_N\}$ denote the set of CAPs. There is a set of tasks $t_i = \{t_{i1}, \ldots, t_{iN_i}\}$ that are generated by IIoT devices under the coverage of c_i, and N_i is the total number of generated IIoT tasks for c_i. Each task t_{ij} can be further represented by a tuple $(ip_{ij}, s_{ij}, op_{ij})$, where ip_{ij} denotes the input size of t_{ij}, s_{ij} denotes the required amount of CPU cycles for finishing the task, and op_{ij} denotes the output size of t_{ij}.

Without the coordination of the SDN technology, the tasks in t_i can be performed in two ways. One way is that the tasks are offloaded and executed at c_i, regardless of the workload balance and queuing delay. The other way is that the tasks are executed at the cloud center via offloading operations, if the current CAP lacks sufficient computational resources. The former can incur long response latency if the number of IIoT tasks is large, and the latter can also incur long transmission delay via the core network. With the assistance of the SDN technology, we can overcome the above drawbacks well. In particular, the decision making is handed over to the SDN controller from each CAP. Thus, with a global view of industrial network and computational resource distribution, SDN can make the best offloading decisions for the IIoT tasks with regards to the response latency. In the next sections, we will detail the communication and computation models, respectively.

A. Communication Model

Define a binary variable x_{ij}^k to denote whether the jth task within the coverage of c_i is performed by c_k. If the jth task within the coverage of c_i is performed by c_k, $x_{ij}^k = 1$ and 0, otherwise. If $x_{ij}^k = 1$ and $k \neq i$, the task t_{ij} is first offloaded to c_i and then forwarded to c_k for the execution. In this context, the response latency for task t_{ij} mainly consists of the following five parts: 1) the offloading delay d_{ij}^{off}; 2) the propagation delay d_{ij}^{pro}; 3) the queueing delay d_{ij}^{que}; 4) the execution delay d_{ij}^{exe}; and 5) the result delivery delay d_{ij}^{rd}. We will analyze them in what follows.

The offloading delay denotes the time taken to offload the task t_{ij} to c_i and can be expressed as

$$d_{ij}^{\text{off}} = ip_{ij}/r_{ji} \tag{1}$$

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on March 01,2023 at 12:50:15 UTC from IEEE Xplore. Restrictions apply.

where r_{ji} is the upload data transmission rate between the IIoT device generating t_{ij} and c_i. r_{ji} can be given as

$$r_{ji} = B\log_2\left(1+\frac{p_{ij}g_{ij}}{\sigma^2+I_{ij}}\right) \tag{2}$$

where B denotes the bandwidth of the wireless channel, p_{ij} is the transmission power of the IIoT device that generates t_{ij}, g_{ij} is the channel gain, and σ^2 and I_{ij} are the noise power and the interference power, respectively.

The propagation delay denotes the time taken to transmit t_{ij} from c_i to c_k, which can be defined as

$$d_{ij}^{\text{pro}} = h_{ik}\eta \tag{3}$$

where h_{ik} denotes the hop counts between c_i and c_k, and η is the average propagation delay for each hop count. Both h_{ik} and η are known to the SDN controller *a priori*. For instance, due to the periodical dissemination of the industrial network status, such as the number of available links, the network traffic, and the number of hops between two different CAPs, η can be obtained based on statistical data and empirical knowledge.

On the other hand, the execution result of task t_{ij} at c_k should be delivered back to the IIoT device that generates it. The time taken for this process actually includes the transmission delay for sending the result of t_{ij} from c_k, the propagation delay from c_k to c_i, and the transmission delay for result delivery from c_i to the IIoT device generating the task. Owing to the high download transmission rate and the smaller size of output in comparison to the input size of the task, the two kinds of transmission delay are much shorter in most cases. Following other works, such as [29] and [30], we also ignore the two kinds of transmission delays in this article, when the result is sent back to the IIoT device. Note that the propagation delay for result delivery is the same as d_{ij}^{pro}. As a result, the delivery delay d_{ij}^{rd} can be expressed as

$$d_{ij}^{rd} = h_{ik}\eta. \tag{4}$$

B. Computation Model

The computational resources at CAPs may be insufficient to support the parallel processing of all the tasks, especially when the number of tasks offloaded to the CAPs is large. Thus, the queueing time cannot be ignored. In particular, the queueing delay is one of the main causes for long response latency and workload imbalance. To evaluate the queueing delay of the IIoT tasks, we assume that the task arrivals at CAP c_k follow a Poisson process with an average rate λ_k [31]. λ_k can be estimated as

$$\lambda_k = \frac{\sum_i \sum_j x_{ij}^k}{T} \tag{5}$$

where T is the total time period. In addition, the average execution time for one offloaded task at c_k, i.e., the service time, can be given as $\sum_i \sum_j x_{ij}^k s_{ij}/f_k \sum_i \sum_j x_{ij}^k$. As a result, the service rate β_k is

$$\beta_k = \frac{f_k \sum_i \sum_j x_{ij}^k}{\sum_i \sum_j x_{ij}^k s_{ij}} \tag{6}$$

where f_k denotes the processing frequency of the CAP c_k. Given the M/M/1 model, the queueing delay for task t_{ij} at c_k can be mathematically given as [5]

$$d_{ij}^{\text{que}} = \frac{\lambda_k}{\beta_k(\beta_k-\lambda_k)}. \tag{7}$$

The execution delay for t_{ij} at c_k can be expressed as

$$d_{ij}^{\text{exe}} = \frac{s_{ij}}{f_k}. \tag{8}$$

For the IIoT task t_{ij}, the total response delay d_{ij} can be expressed as

$$\begin{aligned} d_{ij}^{\text{case1}} &= d_{ij}^{\text{off}} + d_{ij}^{\text{pro}} + d_{ij}^{\text{que}} + d_{ij}^{\text{exe}} + d_{ij}^{rd} \\ &= \frac{ip_{ij}}{r_{ji}} + 2h_{ik}\eta + \frac{\lambda_k}{\beta_k(\beta_k-\lambda_k)} + \frac{s_{ij}}{f_k}. \end{aligned} \tag{9}$$

Note that (9) represents the response latency for the case that the task is offloaded to other lightly loaded CAP instead of its serving CAP, i.e., $x_{ij}^k = 1$ and $k \neq i$. When the task t_{ij} is executed at its serving CAP, i.e., $x_{ij}^k = 1$ and $k = i$, the response delay only includes the transmission delay, the queueing delay, the execution delay, and the result delivery delay. Based on the above analysis, d_{ij} can be given as

$$\begin{aligned} d_{ij}^{\text{case2}} &= d_{ij}^{\text{off}} + d_{ij}^{\text{que}} + d_{ij}^{\text{exe}} \\ &= \frac{ip_{ij}}{r_{ji}} + \frac{\lambda_i}{\beta_i(\beta_i-\lambda_i)} + \frac{s_{ij}}{f_i} \end{aligned} \tag{10}$$

where β_i, λ_i, and f_i are the service rate, the task arrival rate, and the processing frequency of c_i, respectively.

C. Energy Consumption Model

We have taken into account the energy consumption for each CAP in this article. From the viewpoint of the CAP, the energy is mainly consumed for task offloading and execution. For example, if the task t_{ij} is offloaded by c_i, the energy consumption for task offloading can be given as [32]

$$e_{ij}^{\text{off}} = p_i d_{ij}^{\text{off}} \tag{11}$$

where p_i is the transmission power of the CAP c_i. On the other hand, if t_{ij} is performed by c_i, the energy consumption for task execution can be given as [29]

$$e_{ij}^{\text{exe}} = \kappa\varepsilon s_{ij}(f_i)^2 \tag{12}$$

where κ denotes the effective switched capacitance coefficient, and ε is the number of cycles needed to perform one task-input bit at c_i.

V. Problem Formulation

Considering the strict delay requirement of task execution in IIoT, the goal in this article is to optimize the response time for all the tasks generated at the industrial locations. Therefore, the optimization problem can be formulated as

$$(\mathcal{P}1)\ \min \sum_{i=1}^{N}\sum_{j=1}^{N_i}\sum_{k=1}^{N} \overbrace{I_{ik}x_{ij}^k d_{ij}^{\text{case1}}}^{t_{ij}\text{ offloaded to }c_k} + \overbrace{(1-I_{ik})d_{ij}^{\text{case2}}}^{t_{ij}\text{ offloaded to }c_i} \tag{13}$$

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on March 01,2023 at 12:50:15 UTC from IEEE Xplore. Restrictions apply.

$$\text{s.t.} \quad \sum_{k=1}^{N} x_{ij}^{k} = 1 \quad \forall i \in \{1, \ldots, N\} \quad \forall j \in \{1, \ldots, N_i\} \tag{14}$$

$$\sum_{i=1}^{N} \sum_{j=1}^{N_i} x_{ij}^{k} s_{ij} \leq B_k \quad \forall k \in \{1, \ldots, N\} \tag{15}$$

$$\sum_{i=1}^{N} \sum_{j=1}^{N_i} x_{ij}^{k} \kappa \varepsilon s_{ij} (f_k)^2 + \sum_{i=1}^{N} \sum_{j=1}^{N_k} x_{kj}^{i} p_k d_{kj}^{\text{off}} \leq E_k \quad \forall k \in \{1, \ldots, N\} \tag{16}$$

$$f_{k,\min} \leq f_k \leq f_{k,\max} \quad \forall k \in \{1, \ldots, N\} \tag{17}$$

$$I_{ik} \in \{0, 1\} \quad \forall i, k \in \{1, \ldots, N\} \tag{18}$$

$$x_{ij}^{k} \in \{0, 1\} \quad \forall i, k \in \{1, \ldots, N\} \ \forall j \in \{1, \ldots, N_i\} \tag{19}$$

where (14) guarantees that one task can only be performed at one CAP. In other words, the task can be performed either at the "local" CAP or some other lightly loaded CAP. Constraint (15) ensures that the tasks assigned to c_k for performing should be always within its computing capacity. Considering the usage of CAPs for other purposes, e.g., serving as the APs, the energy consumed for task performing should not exceed the given threshold, which can be guaranteed by (16). We assume that I_{ik} is an indicator to denote whether the task is performed at its serving CAP. It can be defined as

$$I_{ik} = \begin{cases} 0, & i = k \\ 1, & i \neq k. \end{cases} \tag{20}$$

For example, when the CAP c_k performs the task t_{ij} and c_k is its serving CAP (i.e., $k = i$), $I_{ik} = 0$, and $I_{ik} = 1$ for the other case.

Proposition 1: The optimization problem ($\mathcal{P}1$) is *NP-hard*.

Proof: Before going further, let us introduce the multiple knapsack problem (MKP). Considering a set of knapsacks $\mathcal{K}$ and items $\mathcal{I}$, respectively, each knapsack $k \in \mathcal{K}$ can pack finite items based on its knapsack capacity a_k and each item $i \in \mathcal{I}$ has a size s_i and profit p_i. MKP aims to find a way to assign items into the knapsacks such that the total profits can be maximized, i.e.,

$$(\mathcal{P}2) \max \quad \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I}} x_{k,i} p_i \tag{21}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{I}} x_{k,i} s_i \leq a_k \quad \forall k \in \mathcal{K} \tag{22}$$

$$\sum_{k \in \mathcal{K}} x_{k,i} = 1 \quad \forall i \in \mathcal{I} \tag{23}$$

$$x_{k,i} \in \{0, 1\} \quad \forall k \in \mathcal{K} \quad \forall i \in \mathcal{I} \tag{24}$$

where (22) guarantees that each knapsack $k \in \mathcal{K}$ can only pack finite items given its knapsack capacity a_k. Constraint (23) means that each item $i \in \mathcal{I}$ can only be packed into at most one knapsack at the same time. $x_{k,i}$ is defined as a binary variable to denote whether the item i is packed into the knapsack k. Particularly, $x_{k,i} = 1$, if the item i is packed into the knapsack k and $x_{k,i} = 0$, if i is not packed into k.

It is known from [33] that the MKP ($\mathcal{P}2$) is an NP-hard problem. In the next, we will prove that our optimization problem ($\mathcal{P}1$) is actually an instance of the MKP in the sense that problem ($\mathcal{P}1$) can be transformed into problem ($\mathcal{P}2$). First, considering that problem ($\mathcal{P}2$) is a maximization problem, we need to transform our minimization problem into a maximization problem, which can be realized easily, e.g., by the inverse computation. In our system model, define $\mathcal{T} = \bigcup t_i (1 \leq i \leq N)$ as a set of tasks generated by all the IIoT devices scattered across different communities. Then, we can regard the set of CAPs (i.e., $\mathcal{C}$) and the set of tasks (i.e., $\mathcal{T}$) in our problem as the sets $\mathcal{K}$ and $\mathcal{I}$ in problem ($\mathcal{P}2$), respectively.

Each task has the required amount of CPU cycles s_{ij} that is corresponding to the item size s_i. Each CAP has a computing capability B_k corresponding to the backpack capacity a_k. The tasks assigned to each CAP should be always within the computing capacity of the CAP. Such a constraint corresponds to the knapsack constraint as shown in inequality (22). In the meanwhile, each task can only be performed by one CAP as shown in (14) at the same time, which corresponds to the fact that each item can be only packed into one knapsack as shown in (23) in problem ($\mathcal{P}2$). The response delay or the inverse of it for each task performing at the CAP can be regarded as the profit of each item in ($\mathcal{P}2$). Therefore, problem ($\mathcal{P}1$) is equivalent to problem ($\mathcal{P}2$) in which $\sum_{i=1}^{N} N_i$ items are packed to N knapsacks with the purpose of response delay optimization. If there exists an algorithm that can maximize the total profits for problem ($\mathcal{P}2$), there also exists an algorithm which can optimize the total response delay for problem ($\mathcal{P}1$). Since problem ($\mathcal{P}2$) is NP-hard, problem ($\mathcal{P}1$) is also NP-hard. ■

It is prohibitively costly to try out all the possible solutions in the solution domain, since it actually takes exponential time (i.e., $O(N^{\sum_{i=1}^{N} N_i})$) to obtain the optimal solution, which is fundamentally difficult in reality especially considering the strict latency requirements of IIoT tasks. On the other hand, the above approach needs the SDN controller to make decisions after the information of all the tasks is collected. However, task handling in batches will incur extra waiting time for the early generated tasks.

VI. Proposed Task Offloading Strategy

In this section, a greedy heuristic approach is put forward for seeking the approximate solution to problem ($\mathcal{P}1$). In the proposed SDN-assisted MEC architecture for IIoT tasks offloading, the decisions on the task assignment are made by the SDN controller. The algorithm executed by the SDN controller should be of low computational complexity, such that the quick decision making can cater to the strict delay requirement of the IIoT tasks. In particular, we aim to solve the problem in polynomial time. Each task can be handled as soon as it is generated by the IIoT device.

Intuitively, it is unnecessary to offload a task to another lightly loaded CAP, if the delay at its serving CAP is less than the delay in offloading. It is possible that the offloading delay, the propagation delay, and the result delivery delay reduce the advantage of the lightly loaded CAP. Based on this heuristic

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on March 01,2023 at 12:50:15 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: GATO in SDN-Assisted IIoT

Input: $\mathcal{C}$, η, σ^2, $\boldsymbol{p}$, $\boldsymbol{f}$, κ, ε
Output: Solution to $\mathcal{P}1$

1 CAPs periodically report respective state information to the SDN controller;
2 **for** each generated task t_{ij} **do**
3 Calculate d_{ij}^{case2} based on Eq. (10);
4 **for** each CAP $\{c_k | c_k \in \mathcal{C}$ and $k \neq i\}$ **do**
5 Calculate d_{ij}^{case1} based on Eq. (9);
6 **end**
7 Construct $\mathcal{U} = \{d_{ij}^{case1} | d_{ij}^{case1} < d_{ij}^{case2}\ \forall k \in \mathcal{C} \backslash \{i\}\}$;
8 **if** $\mathcal{U} \neq \emptyset$ **then**
9 Determine the CAP index $k^* = \arg\min_k \mathcal{U}$;
10 **while** the CAP c_{k^*} violates the constraints **do**
11 Update $\mathcal{U}$ by $\mathcal{U} = \mathcal{U} \backslash \{d_{ij}^{case1} | k = k^*\}$;
12 Determine the CAP index $k^* = \arg\min_k \mathcal{U}$;
13 **end**
14 **if** $\mathcal{U} \neq \emptyset$ **then**
15 Offload the task t_{ij} to c_{k^*} for the execution;
16 Record its response latency;
17 **else**
18 t_{ij} is executed at its serving CAP c_i;
19 Record its response latency;
20 **end**
21 **else**
22 t_{ij} is executed at its serving CAP c_i;
23 Record its response latency;
24 **end**
25 **end**
26 Return the total response latency;

rule, we can redefine the binary decision variable x_{ij}^k as

$$x_{ij}^k = \begin{cases} 0, & i = k \\ 1, & d_{ij}^{\text{case1}} < d_{ij}^{\text{case2}} \text{ and } i \neq k. \end{cases} \tag{25}$$

The SDN controller can gather the status of each CAP including queuing length, the current workloads, and so on. With the global view of the industrial networks, some AI-based technologies can be adopted to train the collected data offline. By doing so, the controller can further predict some valuable information, such as average propagation delay, queueing delay, and the result delivery delay. Such predictions can greatly help evaluate the response latency of a task when it is executed by other CAP instead of its serving CAP.

A greedy algorithm for solving problem ($\mathcal{P}1$) is proposed in Algorithm 1. With the aid of the SDN technology, some parameters such as the average propagation delay for each hop count (i.e., η) can be well predicted. The SDN controller knows each task offloading request from the IIoT device. When one offloading request t_{ij} arrives at its serving CAP, the controller is responsible for calculating the response latency in two cases (i.e., d_{ij}^{case1} and d_{ij}^{case2}) (lines 3 and 5). According to our heuristic rule shown in (25), a set $\mathcal{U}$ is constructed to contain all the values of d_{ij}^{case1} that is better than d_{ij}^{case2} (line 7). By doing so, $\mathcal{U}$ contains all the potential solutions. Then, the lightly loaded CAP is determined by a greedy rule, i.e., the CAP with the minimal value of d_{ij}^{case1} is chosen each time (line 9). However, due to the possible constraint violations, each chosen c_{k^*} should be checked (lines 10–13). If the set $\mathcal{U}$ is not empty (line 14), c_{k^*} will be the valid CAP for performing the task t_{ij}. If the set $\mathcal{U}$ is empty, there will be no suitable CAP for t_{ij}. In such a case, t_{ij} will be performed by its serving CAP c_i. The algorithm records the true response latency for each task t_{ij} and finally returns the total response latency for all the tasks.

Complexity Analysis: Time complexity of this greedy algorithm for task offloading is analyzed as follows. Greedy algorithm for task offloading (GATO) handles the IIoT tasks sequentially. For each task t_{ij}, GATO checks all the other CAPs expect c_i with the aim to find the best CAP to perform the task, which takes the time of $O(N)$ and there are total $\sum_{i=1}^{N} N_i$ tasks to handle. On the other hand, the set $\mathcal{U}$ is constructed, which takes the time of $O(N)$. In the worst case, each candidate in $\mathcal{U}$ is invalid owing to the constraint violations, which takes the time of $O(N)$ to check all the candidates in $\mathcal{U}$. Therefore, the total time GATO needs to accomplish the tasks is $O(\sum_{i=1}^{N} N_i * (O(N) + O(N) + O(N))) = O(N \sum_{i=1}^{N} N_i)$. Such low computational complexity can satisfy the strict delay requirement of the IIoT tasks.

TABLE I
PARAMETER SETTINGS

Parameter	Descriptions	Values
N	Number of CAPs	[5, 10]
N_i	Number of offloading requests in c_i	[10,50]
P_i	Transmission power of the CAPs(mW)	[120, 150]
I_{ij}	Noise power (dBm)	[-30, -10]
σ^2	Interference power (dBm)	-30
$\kappa\varepsilon$	Coefficients for the energy consumption	1e-5
f_i	Processing frequency of the CAP c_i	[5MHz, 3.5GHz]
ip_{ij}	The input size of t_{ij}	[10,60]
s_{ij}	Required number of CPU cycles for t_{ij}	[500, 3000]
op_{ij}	The output size of t_{ij}	[30, 40]

VII. PERFORMANCE EVALUATION

To evaluate the proposed algorithm GATO, a simulator is developed, which integrates the SDN technology to MEC for task offloading in IIoT. The simulation is run on a notebook with 1.8 GHz Intel Core i5-8250U CPU, 8 GB of RAM, Microsoft Windows 10 Operating System, Python 3.7. The sections will discuss the simulation settings and analyze the results, respectively.

A. Experimental Settings

The values of the key parameters are listed in Table I. These parameters are mainly initialized experientially, e.g., based on our previous works [29], [30]. In the meanwhile, several assumptions are made as follows. First, given the number of CAPs, the topology of these CAPs is supposed to be random, such that the number of hops between arbitrary two CAPs is different. The SDN controller is assumed to know the topology, and thus, the hop counts between arbitrary two CAPs in the front-haul networks. Second, as mentioned

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on March 01,2023 at 12:50:15 UTC from IEEE Xplore. Restrictions apply.

earlier, some valuable information can be obtained with the aid of AI-based learning technologies such as auto-regression analysis. Accordingly, some parameters, such as the average propagation delay for each hop count (i.e., η), are known to the controller *a priori*. Third, we assume that the number of tasks generated by the IIoT devices in each CAP is the same. However, it can also be easily extended to the case with different number of task offloading requests for each CAP.

In terms of the performance comparison, several approaches are compared to the proposed greedy approach. On the one hand, we choose as the benchmark the approach that all the task offloading requests are served by its serving CAPs. This approach does not apply the SDN technology. It does not address the issue of workload imbalance among CAPs either. We denote this approach by the local approach in the experiment.

On the other hand, in addition to our proposed greedy rule, there are also other greedy rules to guide the solution searching in the solution space. When the SDN controller detects the workload imbalance in the industrial network, some tasks may be offloaded to a lightly loaded CAP for the execution. As mentioned earlier, the main purpose is to reduce the response latency and improve the QoE for the IIoT devices when the tasks are executed by other CAPs instead of its serving CAP.

However, task offloading in this way sometimes has failed to meet the expectation. This is because compared to the local approach, the new delays could reduce the advantage of the lightly loaded CAP. Such delays usually include the offloading delay for the serving CAP, the propagation delay between the serving CAP and the destination CAP, and the result delivery delay from the destination CAP to the IIoT device. Namely, many factors can affect the performance of the task offloading. Therefore, there are always some heuristic rules which can be adopted to improve the performance of the task offloading.

For instance, when tasks are offloaded to other CAPs for the execution, the criteria for the destination CAP selection, in addition to our greedy rule, can be the least propagation delay, the least queuing delay, and the maximal computing capabilities. For instance, the destination CAP selection with the least propagation delay always chooses the CAP which has the least propagation from the source CAP. The destination CAP selection with the least queuing delay always chooses the CAP which has the least queuing delay while the last one always chooses the CAP that has the maximal computing capability. Specifically, we in the experiment choose the destination CAP with the minimal queueing delay and the maximal computing capability as the comparison approaches, and we denote the two approaches by "Min_Q" and "Max_C," respectively.

B. Simulation Result and Analysis

Several sets of simulation are conducted to investigate the greedy approach in terms of the optimal values and the running time taken to obtain them. It shall be noted that the optimal values as well as the running time can be influenced by several parameters. These parameters include the number of task offloading requests for each CAP, the number of CAPs, and the computing capabilities of each CAP. Therefore, we should inspect the influence of these parameters upon GATO. In the next, we will show and discuss the corresponding simulation results, respectively.

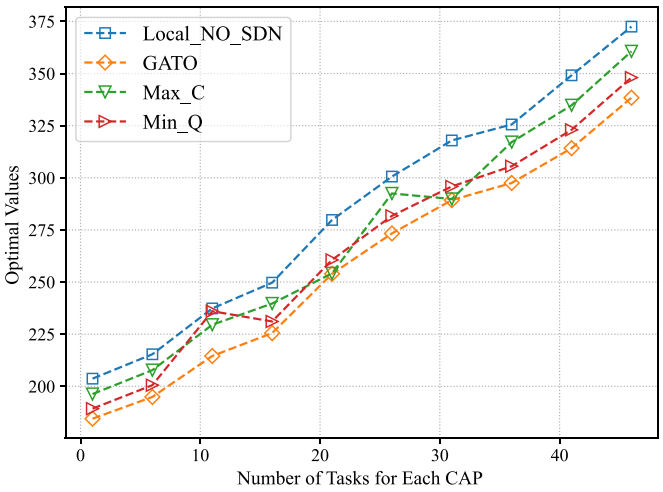


Fig. 2. Response latency comparison with different number of tasks for each CAP.

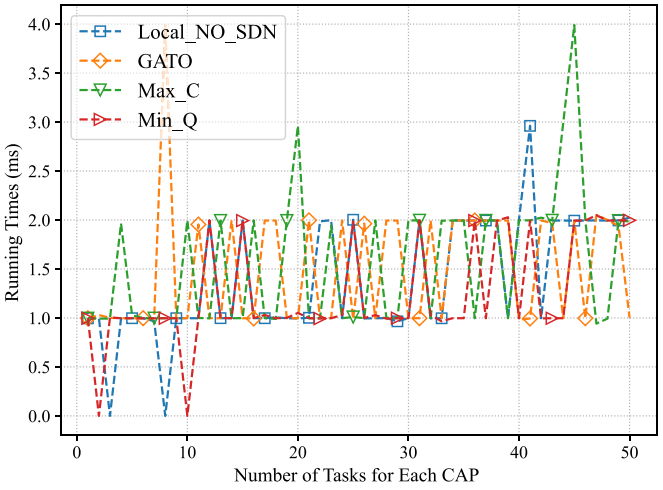


Fig. 3. Running time comparison with different number of tasks for each CAP.

First, we aim to investigate the influence of the number of tasks upon the performance of GATO. Three other approaches are used as a contrast, i.e., the local approach, two other greedy approaches mentioned above. We denote them by Local_NO_SDN, GATO, Max_C, and Min_Q, respectively. The simulation results are demonstrated in Figs. 2 and 3, respectively. In particular, the y-coordinate in Fig. 2 denotes the optimal values (i.e., the response latency for all the tasks) and the x-coordinate in Fig. 2 denotes the number of tasks for each CAP. On the other hand, the y-coordinate in Fig. 3 is the amount of time taken to obtain the corresponding optimal values.

We can easily observe that GATO can achieve the best performance among the four approaches with regards to the optimal values and the local approach without heuristic rules has the worst performance. It is understandable and acceptable since all the tasks are handled in respective serving CAPs. This

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on March 01,2023 at 12:50:15 UTC from IEEE Xplore. Restrictions apply.

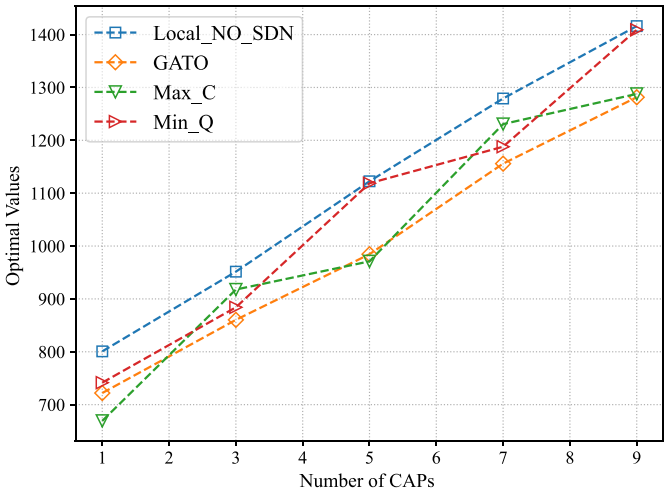


Fig. 4. Response latency comparison with different number of CAPs.

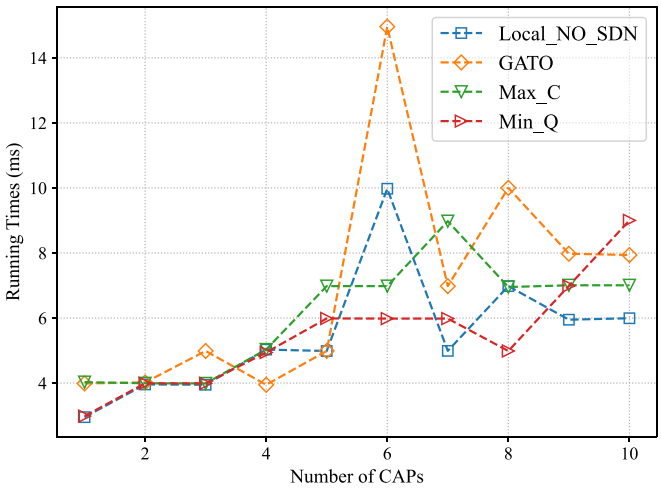


Fig. 5. Running time comparison with different number of CAPs.

way does not take into account the workload imbalance and thus can bring down the QoE for the IIoT devices. With the increasing number of tasks for each CAP, the response latency for all the tasks in the four approaches generally increase.

Furthermore, GATO seems more stable than the other two greedy approaches. The performance of Max_C sometimes is better than Min_Q and sometimes not. Min_Q seems to have the greatest volatility among the four approaches. For example, when the number of tasks for each CAP equals 14, Min_Q yields the same performance as GATO. However, when the number of tasks for each CAP comes to 48, Min_Q yields the worst performance that is almost the same as the local approach without the SDN technology. We actually think the following factors can lead to such results. Either Max_C or Min_Q only considers one factor that may influence the performance of task offloading when the tasks are assigned to lightly loaded CAPs for the execution. In particular, Max_C focuses on the CAP with maximal computing capabilities while Min_Q focuses on the CAP with minimal queueing delay. In contrast, our approach GATO considers all the factors that could affect the response latency of one task when it is offloaded. On the other hand, the results shown in Fig. 3 reveal that the four approaches can achieve real-time response, which can cater to the strict latency requirement of IIoT devices.

Second, we aim to investigate the influence of the number of CAPs upon the performance of task offloading approaches. Similarly, three approaches are compared, i.e., Local_NO_SDN, GATO, Max_C, and Min_Q, respectively. The simulation results are shown in Figs. 4 and 5, respectively. The y-coordinate in Fig. 4 denotes the optimal values and the x-coordinate denotes the number of CAPs.

Similar to the results shown in the first set of experiments, among the four approaches, GATO can achieve the best performance while the local approach without heuristic rules has the worst performance. This is because the local approach does not apply the SDN technology and does not pay attention to the workload imbalance either. With the increasing number of CAPs, the response latency for all the tasks in the four approaches also generally increase.

In addition, GATO is much more stable than Max_C or Min_Q with regards to the optimal values. Comparatively speaking, both Max_C or Min_Q have demonstrated relatively great volatility among the four approaches. Furthermore, there is no particular advantage to utilizing Max_C over Min_Q, this is because they are not always dominated by each other all the time. Specifically, Max_C is sometimes better than Min_Q while Min_Q is sometimes better than Max_C. One of the main causes is that neither of them considers all the factors that could influence the performance of task offloading. Instead, Max_C prefers the CAP with maximal computing capabilities while Min_Q prefers the CAP with minimal queueing delay.

The results shown in Fig. 5 denote the running time of decision algorithms run by the SDN controller. It seems that all the four approaches have demonstrated great fluctuation as the number of tasks for each CAP increases. However, the time taken to obtain the results for the four approaches is all milliseconds and thus the fluctuation can be ignored. To sum up, the results have revealed that the four approaches can achieve real-time response. Compared to some evolutionary algorithms, such as genetic algorithm and particle swarm optimization algorithm, which require response latency in subseconds and even seconds, these heuristic algorithms can better cater to the strict latency requirement of the IIoT devices.

In the last simulation, we aim to investigate the effects of computing capabilities of each CAP upon the performance of task offloading approaches. We assume that the processing frequency f_k allocated by c_k for task performing ranges from $f_{k,\min}$ to $f_{k,\max}$. Usually, $f_{k,\max}$ denotes that c_k performs tasks at the maximum CPU speed while $f_{k,\min}$ denotes that c_k performs tasks at the minimum CPU speed. Such a variation allows for multiobjective behaviors at the edge server. After all, it is unpractical to process tasks at the maximum CPU speed all the time, and there are supposed to be residual computing resources for other purposes. In the experiment, we compare GATO with Max_C and Min_Q. Fig. 6 demonstrates the simulation results, in which the y-coordinate denotes the optimal values and x-coordinate denotes the computing frequencies of each CAP. In particular, the maximal value of f_i is set to

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on March 01,2023 at 12:50:15 UTC from IEEE Xplore. Restrictions apply.

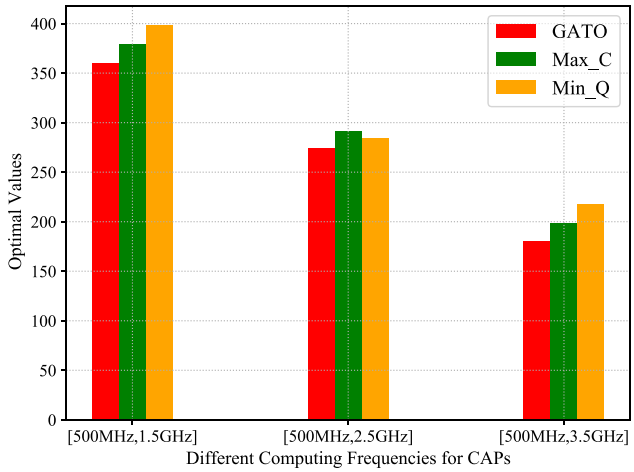


Fig. 6. Performance comparison with different computing frequencies.

be 3.5 GHz and the minimal value is set to be 500 MHz. Generally, as the processing frequency increases, the execution delay decreases, which can be easily observed in the figure. In addition, GATO achieves the best performance among three approaches no matter how the processing frequencies change. Both Max_C and Min_Q fluctuate a lot, which is in accordance with the simulation result analysis in the former sets of experiments. Accordingly, Max_C is sometimes better than Min_Q and sometimes not.

VIII. Conclusion

This article proposed an SDN-assisted MEC architecture, where CAPs can cooperatively execute the IIoT tasks, with the assistance of the SDN technology. In particular, by virtue of a global industrial network view, SDN is aware of each CAP. Thus, the SDN controller is responsible for making decisions on task offloading and resource allocation when task offloading requests arrive. The workload imbalance can be mitigated to a great extent, since the tasks can be offloaded to other lightly loaded CAP for the execution instead of its serving CAP. The optimization objective is minimization of the response latency for all the tasks in IIoT. We have put forward a greedy algorithm to solve the problem, with the aim to obtain the approximate solution in polynomial time. The experimental evaluation has revealed its advantages over other approaches in terms of response latency. Furthermore, the proposed approach can respond to the task offloading requests in real time.

References

[1] W. Xia, J. Zhang, T. Q. S. Quek, S. Jin, and H. Zhu, "Mobile edge cloud-based industrial Internet of Things: Improving edge intelligence with hierarchical SDN controllers," *IEEE Veh. Technol. Mag.*, vol. 15, no. 1, pp. 36–45, Mar. 2020.

[2] J. Wan *et al.*, "Toward dynamic resources management for IoT-based manufacturing," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 52–59, Feb. 2018.

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

[4] A. Voellmy and J. Wang, "Scalable software defined network controllers," in *Proc. ACM SIGCOMM*, Helsinki, Finland, Aug. 2012, pp. 289–290.

[5] S. Misra and S. Bera, "Soft-VAN: Mobility-aware task offloading in software-defined vehicular network," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2071–2078, Feb. 2020.

[6] Z. Ma *et al.*, "Towards revenue-driven multi-user online task offloading in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 5, pp. 1185–1198, May 2022.

[7] K. Peng *et al.*, "Joint optimization of service chain caching and task offloading in mobile edge computing," *Appl. Soft Comput.*, vol. 103, May 2021, Art. no. 107142.

[8] J. Xue, Q. Hu, Y. An, and L. Wang, "Joint task offloading and resource allocation in vehicle-assisted multi-access edge computing," *Comput. Commun.*, vol. 177, pp. 77–85, Sep. 2021.

[9] X. Li, Y. Qin, H. Zhou, and Z. Zhang, "An intelligent collaborative inference approach of service partitioning and task offloading for deep learning based service in mobile edge computing networks," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 9, p. e4263, 2021.

[10] G. S. S. Chalapathi, V. Chamola, C.-K. Tham, S. Gurunarayanan, and N. Ansari, "An optimal delay aware task assignment scheme for wireless SDN networked edge cloudlets," *Future Gener. Comput. Syst.*, vol. 102, pp. 862–875, Jan. 2020.

[11] K. Liu, J. K. Y. Ng, V. C. S. Lee, S. H. Son, and I. Stojmenovic, "Cooperative data scheduling in hybrid vehicular ad hoc networks: VANET as a software defined network," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1759–1773, Jun. 2016.

[12] L.-A. Phan, D.-T. Nguyen, M. Lee, D.-H. Park, and T. Kim, "Dynamic fog-to-fog offloading in SDN-based fog computing systems," *Future Gener. Comput. Syst.*, vol. 117, pp. 486–497, Apr. 2021.

[13] V. Balasubramanian, M. Aloqaily, and M. Reisslein, "An SDN architecture for time sensitive industrial IoT," *Comput. Netw.*, vol. 186, Feb. 2021, Art. no. 107739.

[14] F. Naeem, M. Tariq, and H. V. Poor, "SDN-enabled energy-efficient routing optimization framework for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5660–5667, Aug. 2021.

[15] A. Boukerche and V. Soto, "Computation offloading and retrieval for vehicular edge computing: Algorithms, models, and classification," *ACM Comput. Surv.*, vol. 53, no. 4, pp. 1–35, 2020.

[16] A. Mahmood, W. E. Zhang, and Q. Z. Sheng, "Software-defined heterogeneous vehicular networking: The architectural design and open challenges," *Future Internet*, vol. 11, no. 3, p. 70, 2019.

[17] N. Nasser, L. Karim, A. Ali, M. Anan, and N. Khelifi, "Routing in the Internet of Things," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2017, pp. 1–6.

[18] A. BinSahaq, T. R. Sheltami, A. S. Mahmoud, and N. Nasser, "Bootstrapped LARAC algorithm for fast delay-sensitive QoS provisioning in SDN networks," *Int. J. Commun. Syst.*, vol. 34, no. 11, p. e4880, 2021.

[19] D.-J. Deng, S.-Y. Lien, C.-C. Lin, S.-C. Hung, and W.-B. Chen, "Latency control in software-defined mobile-edge vehicular networking," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 87–93, Aug. 2017.

[20] I. Yaqoob, I. Ahmad, E. Ahmed, A. Gani, M. I. Razzak, and N. Guizani, "Overcoming the key challenges to establishing vehicular communication: Is SDN the answer?" *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 128–134, Jul. 2017.

[21] M. Alowish, Y. Shiraishi, Y. Takano, M. Mohri, and M. Morii, "A novel software-defined networking controlled vehicular named-data networking for trustworthy emergency data dissemination and content retrieval assisted by evolved interest packet," *Int. J. Distrib. Sens. Netw.*, vol. 16, no. 3, 2020, Art. no. 1550147720909280.

[22] S. H. Ahmed, S. H. Bouk, D. Kim, D. B. Rawat, and H. Song, "Named data networking for software defined vehicular networks," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 60–66, Aug. 2017.

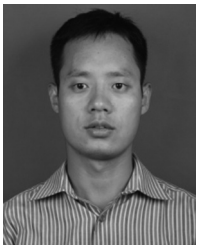
[23] A. Alioua, S.-M. Senouci, S. Moussaoui, H. Sedjelmaci, and M.-A. Messous, "Efficient data processing in software-defined UAV-assisted vehicular networks: A sequential game approach," *Wireless Pers. Commun.*, vol. 101, no. 4, pp. 2255–2286, 2018.

[24] L. Zhao, K. Yang, Z. Tan, X. Li, S. Sharma, and Z. Liu, "A novel cost optimization strategy for SDN-enabled UAV-assisted vehicular computation offloading," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3664–3674, Jun. 2021.

[25] S. Khan and A. Akhunzada, "A hybrid DL-driven intelligent SDN-enabled malware detection framework for Internet of medical things (IoMT)," *Comput. Commun.*, vol. 170, pp. 209–216, Mar. 2021.

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on March 01,2023 at 12:50:15 UTC from IEEE Xplore. Restrictions apply.

[26] S. Sakib, M. M. Fouda, Z. M. Fadlullah, N. Nasser, and W. Alasmary, "A proof-of-concept of ultra-edge smart IoT sensor: A continuous and lightweight arrhythmia monitoring approach," *IEEE Access*, vol. 9, pp. 26093–26106, 2021.
[27] S. Liaqat, A. Akhunzada, F. S. Shaikh, T. Giannetsos, and M. A. Jan, "SDN orchestration to combat evolving cyber threats in Internet of Medical Things (IoMT)," *Comput. Commun.*, vol. 160, pp. 697–705, Jul. 2020.
[28] U. Ashraf and C. Yuen, "Capacity-aware topology resilience in software-defined networks," *IEEE Syst. J.*, vol. 12, no. 4, pp. 3737–3746, Dec. 2018.
[29] C. Tang, C. Zhu, H. Wu, Q. Li, and J. J. P. C. Rodrigues, "Toward response time minimization considering energy consumption in caching-assisted vehicular edge computing," *IEEE Internet Things J.*, vol. 9, no. 7, pp. 5051–5064, Apr. 2022.
[30] C. Tang, X. Wei, C. Zhu, Y. Wang, and W. Jia, "Mobile vehicles as fog nodes for latency optimization in smart cities," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 9364–9375, Sep. 2020.
[31] X. Wei, C. Tang, J. Fan, and S. Subramaniam, "Joint optimization of energy consumption and delay in cloud-to-thing continuum," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2325–2337, Apr. 2019.
[32] C. Tang, M. Hao, X. Wei, and W. Chen, "Energy-aware task scheduling in mobile cloud computing," *Distrib. Parallel Databases*, vol. 36, no. 3, pp. 529–553, 2018.
[33] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 1st ed. Berlin, Germany: Springer, 2000.

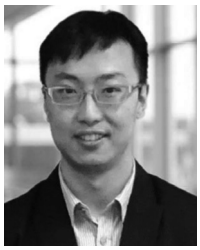


Chaogang Tang (Member, IEEE) received the Ph.D. degree from the School of Information Science and Technology, University of Science and Technology of China, Hefei, China, and the Department of Computer Science, City University of Hong Kong, Hong Kong, under a joint Ph.D. Program, in 2012.

He is currently with the China University of Mining and Technology, Xuzhou, China. His research interests include vehicular edge computing, vehicular fog computing, and Internet of Things.

Chunsheng Zhu (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of British Columbia, Vancouver, BC, Canada, in 2016.

He is an Associate Professor with the College of Big Data and Internet, Shenzhen Technology University, Shenzhen, China. He has authored more than 100 publications. His research interests mainly include Internet of Things, wireless sensor networks, cloud computing, big data, social networks, and security.



Ning Zhang (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2015.

He is an Associate Professor with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada. His research interests include connected vehicles, mobile edge computing, wireless networking, and machine learning.

Dr. Zhang received six best paper awards from conferences and journals, such as IEEE Globecom and IEEE ICC.

Mohsen Guizani (Fellow, IEEE) received the B.S. (with Distinction) and M.S. degrees in electrical engineering and the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively.

He is currently a Professor with the Machine Learning Department, Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE. His research interests include wireless communications and mobile computing, computer networks, mobile cloud computing, security, and smart grid.

Dr. Guizani is a Senior Member of ACM.

Joel J. P. C. Rodrigues (Fellow, IEEE) is a Professor with the College of Computer Science and Technology, China University of Petroleum (East China), Qingdao, China. He is also a Senior Researcher with the Instituto de Telecomunicações, Aveiro, Portugal. He has authored and coauthored over 780 publications in refereed international journals and conferences, three books, two patents, and one ITU-T Recommendation.

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on March 01,2023 at 12:50:15 UTC from IEEE Xplore. Restrictions apply.