Energy-Aware Service Function Chain Embedding in Edge–Cloud Environments for IoT Applications

Nguyen Huu Thanh, *Member, IEEE*, Nguyen Trung Kien, *Graduate Student Member, IEEE*, Ngo Van Hoa, Truong Thu Huong, *Member, IEEE*, Florian Wamser, and Tobias Hossfeld, *Member, IEEE*

***Abstract*—The implementation of Internet-of-Things (IoT) applications faces several challenges in practice, such as compliance with Quality-of-Service requirements, resource constraints, and energy consumption. In this context, the joint edge–cloud paradigm for IoT applications can resolve some of the issues arising in pure cloud computing scenarios, such as those related to latency, energy, or privacy. Therefore, an edge–cloud environment could be promising for resource and energy-efficient IoT applications that implement virtual network functions (VNFs) bound together into service function chains (SFCs). However, a resource and energy-efficient SFC placement requires smart SFC embedding mechanisms in the edge–cloud environment, as several challenges arise, such as IoT service chain modeling and evaluation, the tradeoff between resource allocation, energy efficiency and performance, and the resource dynamics. In this article, we address issues in modeling resource and energy utilization for IoT applications in edge–cloud environments. A smart traffic monitoring IP camera system is deployed as a use case for a realistic modeling of a service chain. The system is implemented in our testbed, which is designed and developed specifically to model and investigate the resource and energy utilization of SFC embedding strategies. A resource and energy-aware SFC strategy in the edge–cloud environment for IoT applications is then proposed. Our algorithm is able to cope with dynamic load and resource situations emerging from dynamic SFC requests. The strategy is evaluated systematically in terms of the acceptance ratio of SFC requests, resource efficiency and utilization, power consumption, and VNF migrations depending on the offered system load. Results show that our strategy outperforms some existing approaches in terms of resource and energy efficiency, thus it overcomes the relevant challenges from practice and meets the demands of IoT applications.**

***Index Terms*—Edge–cloud computing, Internet-of-Things (IoT) applications, network function virtualization (NFV), resource and energy-aware service chain embedding (RE-SCE), smart city.**

I. Introduction

THE Internet of Things (IoT) has evolved rapidly in the recent years, as it has birthed immense opportunities for the transition from *conventional economy* to *digital economy*. As an intrinsically distributed data-centric system, IoT can collect and provide an enormous volume of data that can be processed, analyzed, interpreted, and converted into useful information and knowledge. Thus, IoT facilitates the creation of new knowledge-based, intelligent service offerings, such as production [1], machinery [2], automotive [3], agriculture [4], environment [5], healthcare [6], energy management [7], and even wireless power transfer to inductively transfer power to devices [8].

In the context of IoT-based applications, *cloud computing* plays a crucial role. As reported by Kaur *et al.* [9], nearly 50 billion IoT devices would be connected to the Internet by 2020, which would create a billion of terabytes that have to be processed only on the cloud infrastructure in a decentralized manner. Moreover, the cloud computing paradigm allows the virtualization of physical resources, such as computing power and storage and network infrastructure, thereby facilitating a dynamic, scalable, and flexible *pay-as-you-go* provisioning of services that greatly reduces the capital expenditure (CAPEX), as well as the operational expenditure (OPEX); moreover, it allows changing the way businesses are conducted by decoupling the *service providers* from the *infrastructure providers*.

However, the recent implementation of cloud-based IoT systems shows that the conventional remote cloud infrastructure with large-sized distributed data centers (DCs) located in the core, and the IoT devices collecting data locating at the edge of the network will run into a series of difficulties as follows.

1) *Latency:* Recent technological advances, such as smart grid, healthcare, and AI-integrated systems require data to be processed and presented in real time. This stringent requirement for Quality of Service (QoS) and Quality of Experience (QoE) can hardly be met in the conventional cloud scenarios because of the high latency due to limited transmission bandwidth and increased networking heterogeneity.
2) *Resource Allocation:* As data have to be generally transferred from the edge devices to the remote DCs, it is required that resources, including the network bandwidth, storage, and computing power, are allocated along the edge-to-cloud path, which could possibly be bottlenecked, especially at the core network, considering the volume and velocity of the data expected would be generated by the IoT devices in the future.

Manuscript received November 13, 2020; revised January 21, 2021; accepted February 28, 2021. Date of publication March 9, 2021; date of current version August 24, 2021. This work was supported by the Hanoi University of Science and Technology (HUST) under Project T2020-SAHEP-009. *(Corresponding author: Nguyen Huu Thanh.)*

Nguyen Huu Thanh, Nguyen Trung Kien, Ngo Van Hoa, and Truong Thu Huong are with the School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi 10000, Vietnam (e-mail: thanh.nguyenhuu@hust.edu.vn).

Florian Wamser and Tobias Hossfeld are with the Chair of Communication Networks, University of Würzburg, 97070 Würzburg, Germany.

Digital Object Identifier 10.1109/JIOT.2021.3064986

3) *Energy Consumption:* Recent surveys have shown that energy consumption in DCs remarkably contributes to the total energy consumption of the cloud due to a large number of high-performance servers deployed, which considerably contributes to its operational costs [10], [11]. In contrast, IoT devices at the edge are normally based on energy-efficient embedded platforms. Thus, a combination of the edge–cloud environment would be a promising solution for better energy efficiency.
4) *Privacy:* Transferring the data from the edge to the core of the network through different ISPs, heterogeneous networking environments, as well as network components may pose security risks for sensitive data. Thus, for some applications, it is preferable that the data remain within the owner's network and is processed locally at the edge.

Fog computing was introduced by Cisco in 2012 [12]. Together with *edge computing*, these concepts emerged in the recent years in order to tackle the aforementioned problems. Its architecture is based on the concepts of *cloud close to the ground* in terms of the distribution of computing power and storage capacity to the edge of the network. This can be achieved by leveraging the device-to-device connection supported by the edge devices in the form of a cluster of computing machines, or a small-scale data center. Distributing workload to the network edge, such as by preprocessing raw data, helps in reducing the backbone network traffic and end-to-end latency, thus improving the overall system performance. Besides that, keeping functions that process private data out of public cloud also secures privacy for users.

To achieve such vision, several architectures for edge–cloud have been proposed in previous studies, for instance, Cloudlet [13], mobile edge computing, or Multi-access Edge Computing (MEC) [14], whose results outperform systems that involve only centralized DCs. Although these models show clear benefits as compared with the conventional centralized, data center-based cloud, the edge computing model has it own disadvantages: 1) *limited resources* at the edge in terms of storage, computing power, and energy and 2) *high complexity* in terms of service deployment and management, as the edge devices are heterogeneous in terms of both hardware and software. Thus, edge computing is often proposed as a supplement to cloud computing, so that the combination of edge and cloud computing would effectively provide a tradeoff between pure edge and cloud paradigms.

To fully utilize the advantages of the edge–cloud paradigm, network function virtualization (NFV) [15] has emerged as a promising technology, as it allows network services to be virtualized. Owing to such virtualization techniques as the virtual machine (VM) or *container*, NFV gives rise to the concept of virtual network function (VNF), a piece of software running on commodity and general hardware and responsible for handling specific network functions. By applying the edge–cloud model, VNFs can be instantiated in edge devices or in servers located in DCs; they can be migrated to other machines as an offloading method for better performance and resource efficiency. VNFs can also be chained together to form a service function chain (SFC) [16], [17]. In such a chain, typically, each VNF executes a certain function and all VNFs must be processed in a specific order. Software-defined networking (SDN) [16], on the other hand, can handle the configuration and control and steer the traffic through each VNF to form an SFC seamlessly as per the user's demand. The programmability characteristic of SDN is highly suitable for synergizing with NFV, as it could greatly reduce the costs and increase the flexibility of the VNFs in the edge–cloud.

The IoT service in the edge–cloud environment has attracted attention from the academia, as well as the industry recently [18]. On the one hand, it is the perfect solution to the aforementioned problems. On the other hand, there remain challenges, such as IoT service chain modeling, resource allocation versus performance, energy efficiency, and so forth. This article addresses the deployment of IoT applications over the SDN/NFV-enabled edge–cloud model with the following contributions.

1) An *edge–cloud framework* for IoT applications with the *testbed* based on OpenStack [19] and Kubernetes [20] is proposed. Through its open interfaces, the testbed was used as an environment for IoT SFC modeling, as well as prototyping, implementing, and testing of SFC embedding algorithms.
2) The impact of a service chain on different key performance measures of an edge–cloud system, such as resource and energy consumption, has been investigated and modeled. A traffic monitoring IP camera system for smart city was taken into account as a use case for a *realistic modeling*. The traffic monitoring camera system was selected on purpose due to its high requirements in terms of resources and the high dynamics related to the variation in traffic flow and traffic environment [21].
3) Based on the developed models, a *heuristic resource and energy-efficient SFC embedding strategy* called the resource and energy-aware service chain embedding (RE-SCE) has been designed for edge–cloud IoT environments. Evaluation results show that our approach outperforms some of the existing systems in terms of SFC request's acceptance ratio, resource utilization, as well as energy consumption.

The remainder of this article is organized as follows: Section II addresses previous work on edge–cloud deployment in IoT, as well as energy and resource-efficient SFC embedding mechanisms; in Section III, the research challenges and objectives have been discussed; next, a service architecture for resource provisioning in the edge–cloud environment and its testbed deployment have been discussed in Section IV; in this section, different test cases are also addressed to measure the performance parameters of the edge–cloud system. In Section V, system modeling is addressed, while Section VI proposes a new SFC embedding strategy, including VNF offloading, VNF mapping (VNFm), and virtual link mapping (VLm) algorithms. Section VII discusses performance results and Section VIII concludes the work.

II. Related Work

Recent advances on the edge–cloud interplay for IoT applications have been discussed in the previous literature. As covered by a special issue of the IEEE IoT Journal [22], there are several recent research aspects that draw attention from the industry and academia as the follows.

1) First, it is the realization of novel *edge–cloud architectures* that can facilitate a flexible and dynamic IoT service deployment of the edge–cloud infrastructure.
2) Based on the IoT system architecture, advanced SFC placement and resource allocation strategies are another aspect of research interest, which closely relates to advanced *SFC embedding algorithms* that satisfy *multicriteria embedding objectives*, such as QoS, resource, and energy efficiency.
3) *Use cases for IoT applications*, such as healthcare, transportation, smart city, and finance are also of interest.
4) Finally, *security* and *privacy* are also a focus, which address approaches to protect the edge–cloud IoT system from different kinds of attacks and prevent privacy leakage.

The scope of this work covers the first, second, and third research aspects mentioned above. In this section, the implementation of the edge–cloud paradigm for IoT has been discussed first. Background information, concepts, and typical previous work have been discussed as a basis for understanding this research. We have then introduced the current state-of-the-art edge–cloud architectures that involve SDN and NFV. Second, we conduct an intensive survey in the field of SFC placement's algorithm, whose goals are multicriteria embedding objectives in general and resource efficiency, particularly energy efficiency. We focused on two methods of the SFC placement algorithm, namely, offloading and mapping strategies. Table I provides a taxonomy and classification of the related work and highlights the contribution of our work.

A. Edge–Cloud Architectures in Support of IoT Services

1) Edge–Cloud Architectures in General: As stated in the first section, the edge–cloud model helps to troubleshoot the problems incurred from using the conventional cloud model or edge model for IoT applications. To this end, it has attracted much attention of both academia and industry in the recent years. There are several well-known edge–cloud models such as mobile edge computing (MEC) [14]. MEC was introduced by the European Telecommunications Standards Institute (ETSI) in 2014, and it primarily focused on a specific area of mobile networks; ETSI has also identified IoT as one of the key use cases of MEC. In this model, MEC servers can be deployed at edge sites such as long-term evolution (LTE) macro base stations (eNodeB) to support workload from end-user devices. On the other hand, the *Cloudlet* model [13], a project of Carnegie Mellon University, does not co-locate with the base station, but its reinforcement resources for the end user is presented as a small "data center in a box" close to the user's devices. The cloudlet's design enables it to work in the Wi-Fi environment. Nastic *et al.* [23] introduced the concept of software-defined IoT units—a novel approach to IoT cloud computing that encapsulates fine-grained IoT resources and IoT capabilities in well-defined APIs to provide a unified view on accessing, configuring, and operating IoT cloud systems, thereby simplifying the provisioning and enabling flexible runtime customizations of software-defined IoT cloud systems.

There are currently various opensource platforms such as OpenStack [19] and CloudStack [24] that help operators to implement the aforementioned edge–cloud concepts. For instance, the Cloudlet model was implemented as OpenStack's extension named "OpenStack++" [25], which was also used to build applications such as "GigaSight," "QuiltView," and "Gabriel." In this model, edge devices can flexibly ask the Cloudlet infrastructure to launch VMs for computing power, and VMs can be created and discarded dynamically. However, it is challenging to implement the above platforms in the edge–cloud environment, where the distributed edge devices have limited computational and storage resources. Kubernetes [20], on the other hand, has been adopted to deploy IoT services in edge devices in the form of a container, which has a lower resource's footprint than that of VM. With Kubernetes, operators can deploy IoT applications in a set of smaller services called microservices, which have many advantages over traditional IoT applications. Microservices are complementary to edge–cloud architecture, as it allows IoT applications to be distributed among different locations.

Recently, the academia has seen a research trend exploring a new computing model named *serverless computing*. Serverless can be seen as a subset of microservices that focuses more on lightweight services composed of stateless/lambda functions. This type of function, usually a small piece of code, performs specific tasks such as image preprocessing. It does not store the function state in memory to limit resource usage, as its concept favors IoT applications. Therefore, these functions are usually nested inside containers rather than VMs. Serverless provides an even more dynamic capability in delivering services than the conventional microservice, as it only operates whenever a specific task is "triggered." This behavior promises increasing resource efficiency for infrastructure providers and provides more elastically billing methods for IoT developers. The serverless platform approach has been adopted by Amazon for Amazon Web service (AWS) Lambda and by Google for Google Cloud Function. The same applies for Azure Functions of Microsoft and IBM OpenWhisk. However, a standard architecture and special workflow need to be identified for serverless computing [26].

2) Edge–Cloud Interplay Based on SDN and NFV for Next-Generation IoT Applications: The provision of services in the edge–cloud paradigm gives rise to the concept of NFV, in which a service is defined as an SFC, including several VNFs. A VNF is a piece of software that can be instantiated on a commodity server based on virtualization techniques, such as VM or a container. Thus, the NFV technology allows IoT services to be virtualized and deployed flexibly and dynamically on the edge–cloud infrastructure in an on-demand manner. The process of creating an SFC is referred to as *service function chaining*.

Cziva *et al.* [27] have proposed GLANF, a container-based NFV for the SDN framework. In the study, a GLANF agent daemon is implemented in commodity servers to communicate with Docker and other machines. The CORD architecture (Central Office Rearchitected as a Datacenter) [28] has several solutions, such as M-CORD and R-CORD for SDN/NFV-enabled edge–cloud environment similar to the one provided in this article. However, these models focus more on MEC and its related access technologies to minimize service latency, CAPEX and OPEX. However, the energy-efficiency aspect is not discussed. Moreover, CORD's solutions and documentation are not yet fully developed, which complicates the designing of a testbed for specific use case. For the same reason, a fully open API to integrate different SFC embedding algorithms is unidentified in CORD. Therefore, we utilized our own implementation of the overall architecture in this work. One of our key contributions is the proposed RE-SCE algorithm, which can also be executed within [28].

The software-defined edge computing (SDEC) architecture [29] offers an open IoT system architecture based on SDEC. It decouples upper level IoT applications from the underlying physical edge resources and builds dynamically reconfigurable smart edge services. The automated provisioning of IoT applications over cloud and fog resources is the key concept of [30], offering an IoT Platform-as-a-Service for NFV-based hybrid cloud/fog systems. A different aspect has been considered in [31], where an auction-based MEC-IoT resource trading market is proposed. The market allows the IoT devices to offload their tasks to MECs for better performance in terms of shorter response times. For the edge computing resource allocation, a reinforcement learning- and belief learning-based double auction mechanism is developed.

Up to now, most architectures with NFV and SDN are designed on Openstack and Kubernetes, which provides a strong foundation for orchestrating the virtual infrastructure. From both academia and industry, operators can implement their own project or use third-party open projects, such as OpenMANO [32], OPNFV [33], and ONOS [34] to leverage their legacy system to the SDN-NFV-based edge–cloud architecture. However, due to the variation of edge–cloud models on specific user cases, a standard architecture for the concept is undetermined at the moment.

B. Next-Generation IoT Service Provisioning Based on SDN and NFV

In this work, provisioning IoT services on the SDN/NFV-based edge–cloud infrastructure is formulated as the *SFC embedding problem*, which is closely related to multicomponent application placement [35], virtual network embedding [36], or virtual data center embedding [37]. In general, SFC embedding deals with mapping VNF requests in the edge–cloud substrate and connecting them together to find an SFC. SFC embedding can be reduced to two well-known $\mathcal{NP}$-hard optimization problems—the *facility location problem* and the generalized assignment problem (GAP). Therefore, the SFC embedding problem is $\mathcal{NP}$-hard too [38]. SFC embedding can further be formulated as an optimization problem with particular objectives, such as efficient resource utilization or minimal energy consumption. Methods for these objectives roughly fall into two categories, namely: 1) *offloading* strategies, which deal with deciding if a VNF or an SFC is placed in the edge or offloaded to the cloud and 2) *mapping* strategies that choose the exact location of the VNF or SFC in physical machines located in the edge or data center. An efficient SFC embedding is generally the combination of both offloading and mapping strategies, as they can balance the resource, energy, as well as other optimization objectives when embedding VNFs at the edge or in the cloud. The existing work on offloading and mapping has been addressed as follows.

1) Offloading Strategies for Efficient IoT Service Provisioning: As for *offloading* strategies, Jemaa *et al.* [39] introduced a heuristic approach called the baseline (BL). The algorithm basically tries to place every VNF at the edge tier, and offloading only occurs when the edge runs out of resources. Zhao *et al.* [40] proposed the SFC mapping algorithm for classifying and combining homogeneous SFCs (SFCCM) algorithm that focuses on solving live-streaming video requests by combining similar requests and mapping them into nodes (edges or servers) that have the minimum cost in terms of processing and bandwidth unit. By assuming that the costs at the edge are generally lower than that of in the cloud, SFCCM tends to allocate resources for SFC requests at the edge tier. These two papers and some other research dealing with offloading strategies suggest that placing as many VNFs as possible at the edge may result in a good performance in terms of reducing complexity, latency, or cost, when compared with that of cloud. However, choosing the minimum cost without balancing edge–cloud resource allocation in several cases will also limit the number of accepted chains in the system as, the "less-expensive" resource might always be preferred, leading to resource shortage for future SFC requests. Moreover, the strategies mentioned above usually do not deal with mapping the requests on specific precise cloud physical servers or embedded computers in a topological physical substrate.

Yang *et al.* [41] presented a joint optimal offloading strategy for large-scale IoT maritime communication combined with mobile edge computing to provide efficient computing capability. The tradeoff between latency and energy consumption is addressed by proposing a two-stage joint optimal offloading algorithm. The two stages deal on the one hand with the optimization of the computational effort, and on the other hand, with the allocation of communication resources with limited energy and sensitive latency.

An efficient offloading of SFCs with dynamic VNF placement in geo-distributed cloud systems has been provided in [42]. Furthermore, an optimization problem has also been formulated that minimizes the embedding costs and the number of placed VNF instances. Moreover, the SFC eMbedding APproach (SFC-MAP) and VNF dynamic release algorithm (VNF-DRA) have been proposed. The performance evaluation results indicate a high SFC request acceptance rate, high network throughput, and high VNF utilization. Nevertheless,

mapping has not been addressed, and an evaluation of the energy efficiency is missing.

2) Mapping Strategies for Efficient IoT Service Provisioning: With respect to *mapping* strategies, recent research has developed mapping algorithms based on solutions for the bin-packing problem. Most of them follow the concept of first fit algorithm [43], in which the requests are packed in a smallest number of physical resources so that resource utilization is optimized, and the physical resources can accommodate a maximum number of requests. Regarding the optimal solution, Dinh-Xuan *et al.* [44] formulated the SFC placement as an ILP problem with objectives to minimize service response time and resource utilization. The problem was then solved with *OPLmodel* in a network of multiple *user DCs* (edge) or *DCs* (cloud). The work assumed that all physical links within a data center have infinite bandwidth and zero delay, which is not practical in real environments. On the other hand, Eramo *et al.* [45] proposed a heuristic algorithm with the goal of maximizing the accepted SFC requests. An online algorithm and the design of an SFC scheduling architecture have been proposed to deal with SFC requests arriving in an online manner. The basic principle of the work is to map SFC requests on least loaded physical servers and links in the edge–cloud to maximize the acceptance ratio of the requests. The algorithm's objective that balances bandwidth usage does not consider the edge–cloud mapping tradeoff to optimize the resources and energy consumption. To the best of our knowledge, until now, there is little research focusing on the dynamic resource problems, in which SFC requests join and leave the system dynamically, which could lead to the resource fragmentation problem [37] that reduces the system resource utilization. To mitigate this problem, many online systems adopt the consolidation technique that migrates VNFs from low-utilized servers to other machines to optimize resources and energy. However, consolidation VNFs without a proper strategy may degrade the QoS/QoE of the SFC, especially throughput, from 12.36% to even 50.30% [46], [47]. The behavior is called *VNF interference*. The reason behind this is the competition between VM/containers for the network I/O bandwidth. Zhang *et al.* [47] suggested in their algorithm that VNFs should be consolidated into the hosts that still have large available throughput.

Sun *et al.* [48] considered also the dynamics of such a system into account in their approach. An efficient deployment of VNFs cannot be based on a rigid placement. It is important to consider the dynamics of the system when many small devices such as IoT devices communicate with each other under energy efficiency constraints. The authors proposed an energy-aware routing and adaptive delayed shutdown mechanism for improving the substrate network's energy efficiency and the delay experience of VNF.

C. Multicriteria Objectives in SFC Embedding

1) Optimal SFC Embedding With Multiple Objectives: There exists a multiobjective optimization challenge in the network deployment and operation. Upon receiving an SFC request, the network should embed the request based on several criteria, such as cost, network resources, QoS, energy consumption, and so forth. This poses a challenge in placing the SFC optimally on the physical edge–cloud substrate so that these embedding objectives are met. Moreover, as SFCs join and leave the edge–cloud system dynamically, resources in terms of server capacity, available link bandwidth may become fragmented. Thus, SFC embedding algorithms should cope with the degradation of system utilization.

NFVdeep [49] proposed an adaptive online SFC deployment with deep reinforcement learning. The optimization goal was to minimize the operation cost of NFV providers and maximize the total throughput of requests while automatically deploying SFCs for requests with different QoS requirements. An efficient path computation for SFC requests was also developed with deep learning by [50], which resulted in high performance in terms of the SFC acceptance rates and the delay of paths. However, both concepts [49], [50] were proposed for NFV in general without considering the characteristics of the edge–cloud interplay and the constraints of edge nodes. Lv and Xiu [51] discussed multiattribute decision making of the mobile edge computing migration strategy based on NFV and SDN techniques. In the end, three sets of simulation experiments were carried out based on MATLAB to validate their multiattribute decision-making strategy with migrations in MEC. The work [52] additionally considered the service chain of such a mobile edge computing use case. Based on the actual business needs, multiple VNFs were grouped into SFCs in a predefined order. The authors propose a breadth-first search (BFS)-based algorithm for efficient SFC provisioning and improving the overall utilization of the physical network. Adhikari *et al.* [53] constructed a delay-sensitive priority-aware offloading strategy for scheduling and processing the tasks based on multilevel feedback queuing. Jin *et al.* [54] also considered latency as one of their objectives besides resource efficiency. They formulated these objectives as a mixed-integer linear programming problem (MILP) and propose a heuristic algorithm with a near-optimal result. The algorithm aimed to reuse the deployed VNFs and select the VNF path with the most reused VNFs to save system's resources. However, the algorithm was designed for an offline system. Thus, the acceptance ratio was not within the evaluation scope as it is in this work. Besides, their work focused on VNF deployment amongst distributed small DCs placed at the edge, while our work considers the edge–cloud environment in which a VNF can be deployed in a server or edge-embedded system with different resource profiles suitable for IoT systems.

2) Solutions Toward Resource and Energy-Efficient IoT Service Provisioning: Among various objectives of SFC embedding, *resource* and *energy efficiency* are of interest to this research work.

Resource efficiency is achieved through the smart placement of application functions in different locations of the edge–cloud infrastructure. Recent advances in embedding virtual networks for NFV can be attributed to the use of machine learning (ML)-based approaches. The key idea of Deepvine [55] is to encode physical and virtual networks as 2-D images, which are then perceivable by a convolutional

deep neural network. Quang *et al.* [56] provided a deep reinforcement learning approach for VNF forwarding graph embedding. Pei *et al.* [57] aimed at an optimal VNF placement in SDN/NFV-enabled networks, which is realized by employing deep reinforcement learning. Other heuristic solutions of the underlying optimization problem of VNF placement utilize graphs and graph metrics such as the betweenness centrality algorithm for component orchestration of NFV platforms (BACON) for small-scale and large-scale DC networks in [58]. However, the focus is neither on SFC nor on the characteristics of the edge–cloud interplay.

Another issue in the edge–cloud paradigm is the wasted *energy* of edge–cloud systems while performing resource allocation for IoT service functions. Even if IoT devices are energy efficient in individual cases, the consumption ratio of the overall application use cases with many devices is high, taking into account the total number of devices and the required service life time [59]. The energy efficiency of different parts of IoT use cases could be improved with proper mechanisms. This includes: 1) energy-efficient access technologies and access mechanisms of sensor devices like proper device sleep schedules; 2) energy-efficient communication protocols also exploiting recent technological advances like device-to-device communications [60] [61]; as well as 3) green cloud computing and data center technology for power savings in the edge–cloud. An overview of recent advances on *greening IoT* for IoT and smart city applications is provided in [62]. Shallari and O'Nils [63] investigated the effects of different application partitioning scenarios on edge device energy consumption and application latency in the edge–cloud environment. On the one hand, the results are similar to the modeling results we present in this article showing that different placement strategy yields different power consumption and responding time. To meet design constraints, a smart partitioning strategy is required in the application development process. On the other hand, the work does not consider NFV aspects, which leverages IoT deployment's agility and flexibility.

Particularly, upon provisioning IoT services in edge–cloud environments, the workload might be fragmentarily distributed all over the computing machines without an efficient allocation and consolidation methods. In case of data center, this scenario could draw an energy wastage as high as 60% of the server energy consumption [64]. Moreover, more the number of servers running, more is the energy needed for cooling system, as it can take up as much as 50% of the data center energy consumption. Hence, the study of energy efficiency in cloud-involved models also contributes to the resilient, green development of the Information and Communication Technology (ICT) industry.

As stated in the *Thematic Issue* of the European Commission in 2015 [65], the limitation of the current approaches is the lack of a combined solution for resource and energy efficiency. Regarding the SFC embedding solutions, many previous approaches [40], [44], [66] focused on finding the placement for allocating VNF with the lowest cost. These approaches may draw more energy consumption, as more servers and network switches could be turned on, which results in as much as 45% and 15% of total data center energy consumption, respectively, [67]. Zhou *et al.* [68] also considered minimizing service deployment cost. Cao *et al.* [69] proposed an energy cost model and two resource allocation strategies for interdomain NFV-enabled networks. As energy cost accounts for more than half of the total underlying network cost, it is crucial to minimize the total energy cost while keeping high mapping revenue for resource allocation. They proposed a novel and efficient mapping strategy and labeled it EERID, which can map each virtual resource among interdomain networks within polynomial time.

D. Differentiation and Taxonomy of Related Work

A taxonomy of the related work on the edge–cloud interplay based on SDN and NFV for IoT applications is provided in Table I. In particular, for each paper listed in the summary table, we have mentioned clearly whether offloading and mapping mechanisms are proposed. We have also evaluated if the IoT system is capable of dealing with dynamics of SFC requests. The evaluation considers performance (acceptance rate and utilization), energy efficiency, and VM migrations for varying loads. These features of the taxonomy are represented by the different columns in Table I. It is evident that our paper goes beyond the state of the art. To the best of our knowledge, it is the only work that considers offloading and mapping mechanisms for the edge–cloud IoT system, which is evaluated holistically regarding performance, energy efficiency, and overhead in terms of migrations.

III. Problem Formulation

This section addresses research challenges in edge–cloud use cases that offer IoT services. We have described a traffic monitoring IP camera system in the context of smart city. The system is used as a specific use case and can be considered a typical edge–cloud IoT system, in which energy consumption and resource utilization are two important performance factors. Based on the system, specific technical issues that will be solved using the framework of this research are to be addressed.

A. Challenges in Providing IoT Services in Edge-Cloud Environments

Providing IoT services in edge–cloud environments based on NFV/SDN has attracted attention of the research and industrial community recently [18] owing to the challenges it poses.

1) *Resource Allocation, Energy Efficiency Versus Performance:* An issue that arises when creating an SFC is how to embed the service chain on top of the edge–cloud physical substrate so that the physical resources are efficiently utilized, while keeping the energy consumption of the system as low as possible and the QoS performance of the network service optimal. This issue is related to the distribution of the VNFs over different hosts, which is known as the *VNF placement problem* [70]. Notably, the performance and efficient resource utilization and energy consumption can hardly be achieved at the same time.

TABLE I
TAXONOMY OF THE WORK RELATED TO THE EDGE–CLOUD INTERPLAY BASED ON SDN AND NFV FOR IOT APPLICATIONS

Paper	Edge-Cloud	Offloading	Mapping	Performance	Energy	Dynamics	Migrations	Test-bed	Use Case
Openstack++ [22]	✔	✔	—	✔	—	✔	✔	✔	Interactive mobile apps and services (cloudlets)
Software-Defined IoT [20]	—	—	—	—	—	✔	—	✔	IoT cloud systems
Container-based NFs [24]	—	✔	—	✔	—	—	—	✔	Network functions (NFs)
DeepViNE [52]	—	—	✔	✔	—	✔	—	—	Virtual network embedding
VNF placement [54]	—	✔	—	✔	—	✔	—	—	NFV networks
Virtual network embedding: a survey [33]	—	✔	✔	✔	✔	✔	✔	—	Virtual network embedding
Energy-efficient network embedding [34]	—	✔	✔	✔	✔	✔	—	—	Virtual network embedding
Baseline Offloading [36]	✔	✔	—	✔	—	—	—	—	QoS-aware VNF
SFC placement [41]	✔	✔	✔	✔	—	—	—	—	General cloud services
VNF interference [43]	—	—	—	✔	—	—	✔	✔	QoS-aware VNF consolidation
Interference-aware VNF placement [44]	✔	—	✔	✔	—	—	✔	—	5G network slices
SFCCM Offloading[37]	✔	✔	—	✔	—	—	—	—	edge–cloud services
Geo-distributed Cloud [39]	✔	✔	✔	✔	✔	—	—	—	edge–cloud SFC
Least Loaded Mapping [42]	✔	—	✔	✔	—	✔	—	—	SFC NFV
CORD [25]	✔	—	—	✔	—	—	—	✔	Telco network infrastructure
Software-defined edge Computing [26]	✔	✔	✔	✔	—	—	—	—	IoT cloud systems
IoT PaaS [27]	✔	✔	✔	✔	—	✔	—	✔	NFV networks
NFVdeep [46]	—	—	✔	✔	—	✔	—	—	Network services
Multi-task deep [47]	—	✔	✔	✔	—	✔	—	—	VNF chains
Maritime IoT [38]	✔	✔	—	✔	✔	—	—	—	Maritime
EAR-ADS [45]	—	—	✔	✔	✔	✔	—	—	Delay-sensitive applications
Multi-attribute MEC [48]	✔	—	✔	✔	✔	✔	✔	—	IoT services
Breadth-first search [49]	—	✔	✔	✔	—	✔	—	—	VNF SFCs
DPTO [50]	✔	✔	—	✔	—	—	—	—	
CDFSA-PGA [51]	—	—	✔	✔	—	—	—	—	MEC services
BACON [55]	—	✔	✔	✔	—	✔	—	—	NFV networks
Cost-efficient VNF orchestration [64]	✔	✔	—	✔	✔	✔	—	—	Cross-edge SFCs
EERID [65]	—	—	✔	✔	✔	✔	—	—	Inter-Domain networks
Our proposed solution	✔	✔	✔	✔	✔	✔	✔	✔	**Smart traffic monitoring**

2) *Resource Dynamics:* In reality, SFC requests can arrive in the system and leave dynamically on demand according to the pay-as-you-go service model in edge–cloud. This poses a challenge as to how to develop mechanisms that can optimally and dynamically reallocate resources, while maintaining low complexity of network operations.
3) *Modeling and Evaluation of SFC Embedding in Edge–Cloud:* In order to design and evaluate the performance of SFC embedding algorithms, the behaviors of the edge–cloud infrastructure in terms of QoS, resource utilization, energy consumption, etc., should be known. That is, *measurement* and *modeling* in edge–cloud should be performed. So far, very little is available in the literature in this regard.

B. Motivating Use Case

A smart traffic monitoring IP camera system was used as a case study to model an edge–cloud IoT system. The IP camera system was originally developed in a previous work of The *et al.* [71]. The design of the system stems from a requirement to get traffic information in busy Asian cities such as Hanoi, where traffic congestion usually occurs, especially during the peak hours. The original IP camera system can detect the traffic condition of the road in real time by processing motion JPEG pictures. Instead of sending the pictures to a central server located in the cloud that could require a high-performance server system, as well as high network throughput with high deployment cost, the system makes use of a low cost ARM-based Raspberry Pi (RPi) residing near the camera to process the pictures. It can deliver four traffic levels of service (LOS) over a low-bandwidth transmission line to a central application server, namely: 1) free flow; 2) stable flow; 3) unstable flow; and 4) breakdown flow.

In this research, we have further assumed that this smart traffic detection system can be deployed in a big city for traffic monitoring by installing IP cameras in many intersections. The massive deployment of thousands of IP cameras with smart processing units at the edge is a typical IoT application, where the edge–cloud implementation could be beneficial. On the one hand, as the resources at the edge are limited, a combined resource allocation strategy in both edge and cloud that allows

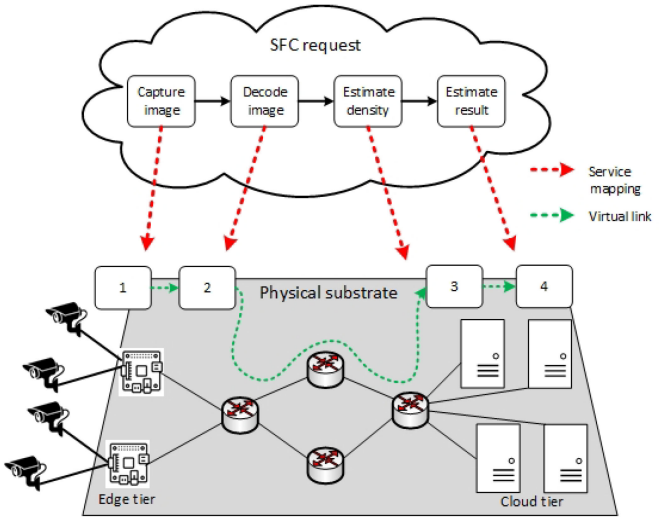

Fig. 1. Traffic monitoring IP camera system as SFC distributed over the edge–cloud architecture.

tasks to be offloaded to the cloud can help increase the overall system utilization. On the other hand, due to the fact that the centralized DCs with computing-intensive applications consume a lot of energy [10], [11], and sending heavy data to the central cloud can be costly, placing some tasks at the edge can reduce the system energy consumption, as well as bandwidth requirement, thereby reducing operational costs. Moreover, we find that energy and resource profiling is essential for developing realistic service chain embedding algorithms. However, resource and energy profiling and modeling of a real system is difficult to find in the literature. Thus, in our research, a system's testbed has been developed to measure, profile, and model resource and energy consumption.

In order to deploy the service in the edge–cloud system, the traffic monitoring software is divided into four different functions, which can be placed in different locations:

1) *Capturing (Cap):* It is connected directly to the camera and gets pictures at six frames per second, with a resolution of 640 × 480. This function solely runs on the embedded computer (EC) and sends captured images to the Decoding block.
2) *Decoding (Dec):* It transforms received images into grayscale images, sets some initial parameters, and sends the initial parameters and gray images to the density estimation block. This function can be placed at the edge or in the cloud.
3) *Density Estimation (Des):* It makes use of the initial parameters and the next gray images to calculate the traffic density and sends the information to the LOS decision block on the server. This function can reside either at the edge or in the cloud.
4) *LOS Decision:* It receives information sent from the density estimation and decides whether the traffic is congested or not. This function solely runs on the central server in the cloud.

These four functions must be performed in order. As depicted in Fig. 1, by modularizing the IoT service, the IoT functions can now be formed as the SFC and be distributed over the edge–cloud infrastructure.

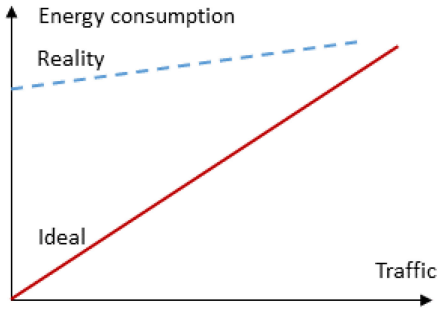

Fig. 2. Ideal energy proportional property.

C. Objectives

Based on the above motivations, the objectives of this work are as the follows.

1) *Resource efficiency* of the physical edge–cloud substrate should be improved in the sense that the overall utilization can be increased. Thus, more SFCs can be served using limited physical resources.
2) *Energy efficiency* must be improved, that is, the overall energy consumption should be proportional to the system utilization. This is called the system's energy proportional property, as shown in Fig 2, that is, the energy consumption in a low utilization scenario should be much lower than in a case of high utilization. In other words, the average energy consumption per SFC should be kept ideally constant, independent of the system's utilization. In this research, the energy consumption should be in proportional to the traffic intensity of vehicles on the roads.
3) *Resource dynamics* is the ability to reallocate resources at the edge or in the cloud dynamically, as SFC requests join and leave the system over time. Thus, the SFC embedding algorithm must be performed periodically to maintain a high system resource utilization. As a consequence, VNFs shall be consolidated in other machines as a result of SFC remapping. VNF relocation might happen often, which poses a challenge in the implementation. Thus, in any approach, the implementation complexity should be taken into account.

IV. System Architecture, Testbed and Measurement

A. System Architecture

Fig. 3 represents the developed architecture of the edge–cloud framework for providing IoT services proposed in this work. The architecture uses two opensource platforms, namely, OpenStack [19] and Kubernetes [20] as its core. In our framework, the infrastructure, including computing, storage, and network resources, are managed by OpenStack's basic modules, namely, Nova, Cinder, and Neutron. The VNF manager (VNFM) and NFV orchestrator (NFVO) are partly composed of Kubernetes controllers and OpenStack

TABLE II
TESTBED CONFIGURATIONS

	Hardware	Software	Description
Data tier	3 x Logitech C170 cameras Resolution: 640 x 480 Frame rate: 6fps	N/A	Capture traffic pictures
Edge tier	Rasberry Pi 2B (RPi): 900MHz quad-core ARM Cortex-A7 CPU, 1GB RAM, 100 Base Ethernet	OS: Raspbian Stretch; Kubernetes v1.12; Container: Docker v8.09.5	One Pi serves up to 3 cameras
Cloud tier & controller	Dell server, 2 x Intel Xenon 5570: 2.93GHz x 4 cores, 96GB memory, 1TB storage, 4x NICs	OS: Ubuntu Server16.04 LTS; OpenStack Ocata; Kubernetes v1.12; Container: Docker v18.09.5	SFC embedding & resource management
Network devices	Cisco Linksys-X2000 router; 4 x 100Mbps NICs	N/A	Transmitting data from edge to cloud
Measurement	MakerHawk USB multimeter UM25; Voltage resolution/accuracy: $0.01V/\pm 0.2\%$; Current resolution/accuracy: $0.001A/\pm 0.8\%$	htop, vnstat, NTP	Measure energy consumption, computing/RAM utilization, latency

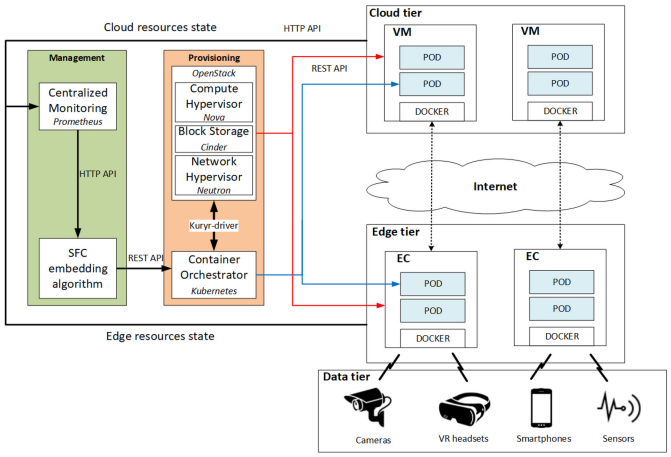

Fig. 3. Overall architecture of the three-tier traffic camera system.

modules' controllers. *Container* and *hypervisor* are the two widely used virtualization technologies in commercial computing systems. While some public clouds, such as Terremark and AWS use either ESXi Hypervisor or XEN Hypervisor, which have gained tremendous popularity. Kernel VM (KVM), another hypervisor technology, is trusted by OpenStack, Microsoft Azure, and its private cloud. On the other hand, Google, IBM/Softlayer, and Jonent are typical examples of successful public cloud platforms using containers. In containerization technology, applications share the same OS, and therefore the deployment sizes and processing time are much less than that of hypervisor. Because of its lightweight deployment, containers can run on devices with limited hardware resources, such as RPi, Beaglebone, or other common embedded platforms. In this research work, to take the advantages of containers, we implemented container-based VNFs in both edge tier and cloud tier. The components of the framework are described in the order from left to right as follows.

1) The *management* block is responsible for monitoring the system states, receiving SFC requests from outside and performing SFC embedding strategy based on the developed SFC embedding algorithms. It contains two components. The centralized monitoring (CM) component manages network resource states and is implemented using the Prometheus [72] project. Prometheus is also installed in the computing units (VMs or embedded computers) for gathering necessary metrics, such as CPU usage, bandwidth usage, and RAM usage. These information are sent to the CM host in the cloud via HTTP API. The other block performs SFC embedding algorithms for VNF provisioning. Different SFC embedding strategies can be integrated and tested via the testbed open API. When an SFC request arrives, locations of the incoming VNFs are decided using SFC embedding algorithms based on the current system state information. A template is then created and sent to the Container Orchestrator in the next block.
2) The *Provisioning* block takes charge of creating VNFs, VNF networks, and the provisioning container-based VNFs to the computing units following the template from the Management block. The block consists of network hypervisor (NH) and container orchestrator (CO). While NH is managed by Neutron, which is responsible for provisioning a unified network among computing units, CO is a Kubernetes controller implemented by Magnum [73] that reads the deployment template from the previous block and deploys VNFs to the Kubernetes nodes. Since Neutron is used as the hypervisor for container networking, Kuryr–Kubernetes [74] is required to make the communication between Neutron and CO block feasible.
3) The *cloud tier* and *edge tier* blocks contain the computing units where VNFs are deployed. At the cloud tier, VMs play the role of computing units and host VNF's containers, whereas physical EC directly host virtual containers for VNFs at the edge tier. ECs also take charge of the communication with data collection devices, such as camera and sensor. The computing units are initiated in the Kubernetes environment and Docker container runtime. Hence, they serve as the Kubernetes worker nodes in the system. VNFs are provisioned on

TABLE III
RESOURCE UTILIZATION IN TERMS OF CPU, RAM, POWER, AND BANDWIDTH OF EACH VNF TYPE RUNNING ON THE TESTBED

VNF type	CPU (%)		RAM (MB)		Power (W)		Bandwidth(Mbps)
	RPi	Server	RPi	Server	RPi	Server	(B_u)
Idle	1.10	0.10	107.00	169.00	1.21	205.10	0.00
Capturing	3.00	N/A	1.50	N/A	0.43*	N/A	47.35
Decoding	8.00	1.50	8.00	26.21	0.09	1.67	16.32
Density Estimation	13.60	6.50	2.50	19.66	0.11	7.23	0.60
LOS Decision	N/A	4.96	N/A	13.10	N/A	5.52	N/A

**Camera power included*

the computing units in the form of Kubernetes pod. The pod networking is transparent and is managed by the Neutron controller because of the Kuryr driver.

It is noted that the proposed architecture can be generalized to be used for other IoT applications and is not necessarily specific to the context of this research.

B. Testbed and Measurement Results

A testbed based on the conceptual architecture shown in Fig. 3 was built for the measurement and modeling of SFC deployment in the edge–cloud infrastructure. Table II shows the testbed configurations. As can be seen, the testbed was set up using an RPi 2B for the edge and Dell high-performance commodity servers for the cloud tier and for the management and provisioning blocks. The energy consumption at edge was measured using a USB multimeter. Besides that, several softwares are also used as measurement and monitoring tools, such as *htop* [75] and *vnstat* [76] for process and bandwidth monitoring, respectively.

The operations of the SFC embedding were investigated using the testbed. VNFs were deployed at both RPi and the server, and they were chained together to form an SFC. Based on the deployment, various tests and measurements were conducted, which were divided into two sets of test cases, namely, *single VNF test cases* and *combination test cases*.

First, in the *single VNF test cases*, the performance parameters of the hardware infrastructure and 4 individual VNFs were conducted, including *CPU*, *RAM*, *power consumption* as well as *bandwidth* and *latency*. We conducted both test cases using the same toolset in the same environment for ten times; each time, we recorded 50 instantaneous measuring samples. The final results are the average number of these values, as shown in Table III and discussed as follows.

1) CPU and RAM Utilization: The CPU utilization in the percentage of the total system computing capacity and the RAM utilization (in MB) were investigated for the embedded computer and the server in the following test cases.

1) *Idle Mode:* It is the state when the computer is turned on but does not run any application, that is, the computer just runs basic services of the OS. In this case, CPU utilization takes up only 1.1% and 0.1% of the total CPU in the embedded computer and the server, respectively. Moreover, RAM utilization is minimal; CPU and RAM utilization in the idle mode are the utilization of the computing hardware infrastructure without any virtual service deployed.
2) *CPU and RAM Utilization of VNFs:* Next, the CPU and RAM utilization was measured when the computer hosted one of the four VNFs in the service chain. In the RPi, a container for the corresponding VNF was instantiated directly in the OS, while for the x86 server platform, the container was embedded into a VM. Note that a VM can host a number of containers. As shown in the table, the resource utilization of the same function is different when placed at the edge or in the core. As the RPi at the edge is less powerful, the CPU utilization at the edge is generally larger than that at the core. As can be seen, a server requires more RAM to run a function than the edge computer. The reason is that edge and cloud platforms have different hardware and OS. The server OS and hardware are designed to best utilize its computing power for generic purposes, while a less powerful edge-embedded system gears toward resource and energy efficiency. Also, the external libraries to make the VNFs compatible with both ARM-based and x86-based processors are different.

2) Power Consumption: The power consumption (in Watt) of the embedded computer and high-performance server is then measured. It is to be noted that in this article, energy and power consumption are used interchangeable.

1) *Baseline Power Consumption:* It is the basic power consumption when the computer is in the idle state. In this state, the computer consumes minimum power, as the clock operates at its minimum frequency. As can be seen in Table III, even in the idle mode, the high-performance server in the core consumes much more power than the RPi.
2) *Power Consumption of VNFs:* With the same configurations mentioned above, we measured the power consumption of the VNFs when running on different platforms. As can be seen, running a function at the core is more power-intensive than the same function running at the edge.

3) Required Bandwidth: As the motion pictures were sent from the edge to the cloud along the service chain in the order of *Capturing – Decoding – Density Estimation – LOS Decision*, we measured the bandwidth required to send the data from a function to the next one. Sending raw pictures from the Capturing function requires the maximum bandwidth, that is, 47.35 Mb/s, while sending the processed data from the Density Estimation to the LOS Decision requires the least, that is, 0.60 Mb/s.

4) Latency: In this work, latency has been defined as the time needed for the SFC to capture images until it can give out the LOS. Latency is highly random, depending on different factors, such as channel utilization, CPU load, number of SFCs embedded in the system, and so forth (Fig. 4).

After conducting the single VNF testcases as discussed above, the second test cases to be conducted were the *combination test*. In the test setups, we composed the VNFs in service chains and instantiated them in different locations in the edge and the core. The purpose of the combination test was to investigate the system behaviors and the impacts of the SFC embedding in terms of resource utilization, power consumption, latency, etc., when multiple SFCs are mapped on the physical substrate.

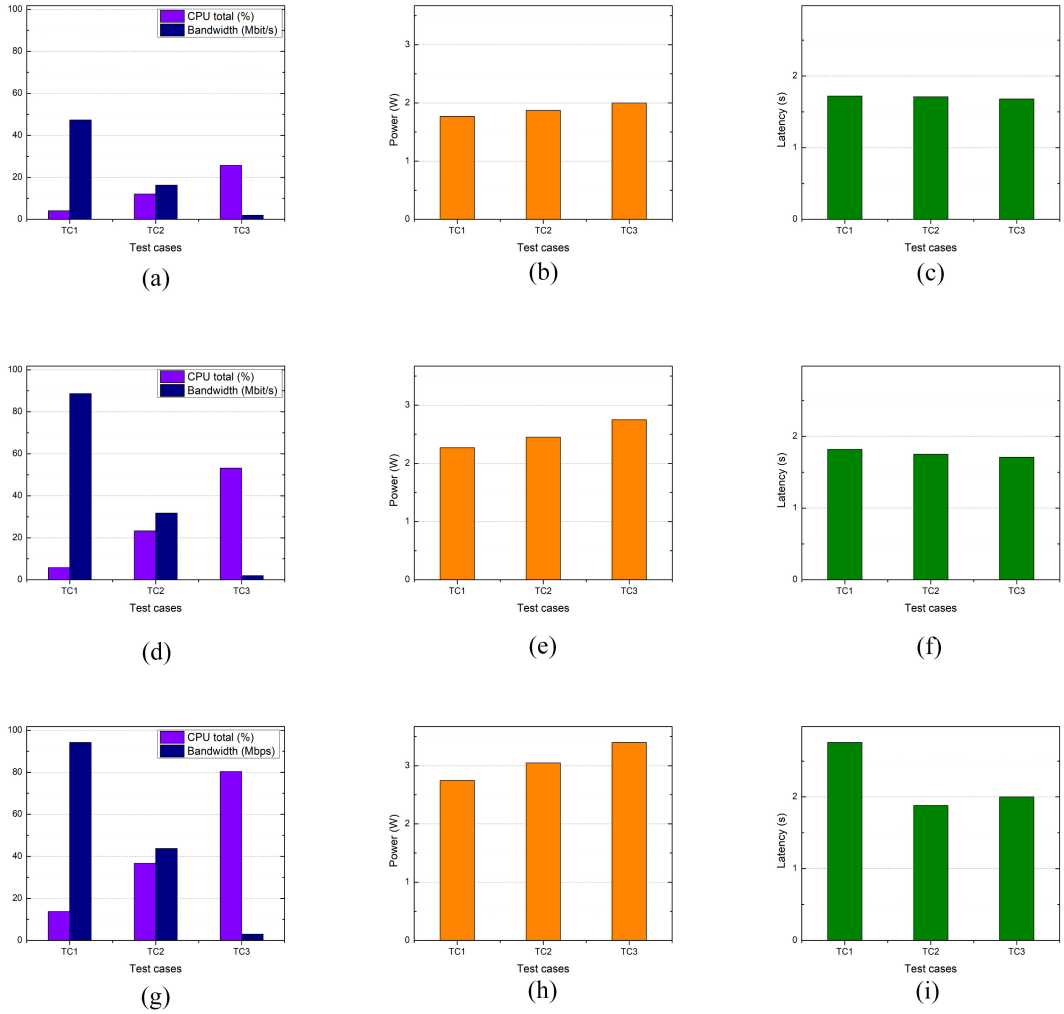

Fig. 4. Experiment result in terms of CPU, BW, latency, and power of the edge computer according to each test case: (TC1) capturing at edge, the rest in cloud; (TC2) capturing and Decoding at edge, the rest in cloud; (TC3) capturing, Decoding and density estimation at edge, LOS decision in cloud. (a) CPU and bandwidth usage 1 SFC. (b) Power consumption 1 SFC. (c) Latency 1 SFC (d) CPU and Bandwidth usage 2 SFCs. (e) Power consumption 2 SFCs. (f) Latency 2 SFCs. (g) CPU & Bandwidth usage 3 SFCs. (h) Power consumption 3 SFCs. (i) Latency 3 SFCs.

When the system receives multiple SFC embedding requests, there are different combinations of resource allocation and VNF placement depending on the SFC embedding strategies. Various scenarios have been investigated on the testbed, but only nine test cases are shown in Fig. 4. As the edge-embedded computer can host up to three SFC requests concurrently, the measurement of one SFC embedding [Fig. 4(a)–(c)], two SFC [Fig. 4(d)–(f)] embedding, and three SFC embedding [Fig. 4(g)–(i)] have been shown. The measurement in each figure is further categorized into: 1) *TC1:* only the Capturing function of an SFC is located at the edge, while the others are in the cloud; 2) *TC2:* the Capturing and Decoding are at the edge, the other two are in the cloud; and 3) *TC3:* only the LOS Decision is in the cloud, while the others are at the edge.

5) Impact of the VNF Locations on Resources and Performance:

1) *Bandwidth Versus CPU Utilization Versus Power Consumption:* Fig. 4(a), (d), and (g) clearly shows the that bandwidth and CPU utilization are inversely proportional to each other. Thus, there should be a tradeoff between the highly loaded edge computer with computing intensive VNFs and the high-bandwidth requirement when offloading these functions to the cloud. Also, it is obvious that placing more functions at the edge causes more power consumption in the embedded computer. However, as the server in the cloud consumes much more power, placing functions at the edge may also save the total power consumption, as shown later on in this article.
2) *Latency:* Fig. 4(c), (f), and (i) shows that latency is lower when capturing, decoding, density estimation are placed at the edge computer. This can be explained by the fact that latency is optimal if the image processing is placed locally near the camera, which requires intensive computing power, as well as high bandwidth between functions. The information sent to the LOS decision needs minimal bandwidth (as shown in Table III) to reduce network delay. The only exception is shown in Fig. 4(i). As can be seen, when three SFCs with three VNFs each are embedded at the RPi, the latency increases. It is

TABLE IV
NOTATIONS USED FOR SYSTEM MODELING

Notation	Description
$G^p = \{N^p, ND^p, L^p\}$	Physical substrate graph
$N^p = \{E^p, S^p\}$	Set of edge computers and cloud servers, respectively
$E^p = \{e_i^p\}; i \in \Omega_e$	Set of edge computers
$S^p = \{s_j^p\}; j \in \Omega_s$	Set of servers in the cloud
$ND^p = \{n_k^p\}; k \in \Omega_n$	Set of physical network nodes, including nodes in the core of the network and nodes in data center networks
$L^p = \{l_m^p\}; m \in \Omega_l$	Set of physical links
$R_j = \{r_j^1, r_j^2, r_j^i, ...\}$	Set of SFC requests originated from the edge computer e_j^p, thus the beginning of the SFC should be from e_j^p
$r_j^i = \{VNF_j^i, L_j^{V_i}, t_j^i, d_j^i\}$	The i^{th} request originated from edge computer e_j^p
$A = \{r_1^1, r_1^2, r_j^i, ...\}$	Set of SFC requests that has been mapped successfully to the system
$vnf_j^{i,k}$	VNF type $k \in \{Cap, Dec, Des, LOS\}$ belonging to the i^{th} request originated from edge computer e_j^p
$VNF_j^i = \{vnf_j^{i,k}\}$	Set of VNFs belonging to the i^{th} request originated from edge computer e_j^p
$L_j^{V_i} = \{l_j^{V_i}(u, v)\}$	Set of virtual links belonging to the i^{th} request originated from edge computer e_j^p, interconnecting $vnf_j^{i,u}$ and $vnf_j^{i,v}$
$l_j^{V_i}(u, v)$	Virtual link belonging to the i^{th} request originated from edge computer e_j^p that connects two consecutive VNF $vnf_j^{i,u}$ and $vnf_j^{i,v}$
t_j^i	Arrival time of request r_j^i
d_j^i	Service time of request r_j^i

because the embedded computer is overloaded with the average CPU utilization reaching 80% [Fig. 4(g)].

Thus, it is observed that placing more functions at the edge can reduce the latency and required transmission bandwidth but may cause the edge computer to be overloaded.

6) Impact of the Number of SFCs on Resource and Performance: It is obvious that the more SFCs with VNFs are placed in the edge computer, the more the computer is overloaded. When placing three SFCs with nine VNFs at the edge, as represented in Fig. 4(g)–(i) (TC3), the average CPU utilization becomes as high as 80%, and the transmission bandwidth becomes saturated at around 94 Mb/s, which is maximum bandwidth of the Fast Ethernet interface in our experiment. This shows that the RPi cannot host more than three SFCs for the smart traffic detection services.

V. SYSTEM MODELING

The measurement results in Section IV have been analyzed and modeled so that they can be applied to a larger-scale simulation of the system. Table IV defines notations used for the formulation.

A. SFC Embedding Problem

In the model, the physical substrate G^p contains three physical elements, namely, N^p, ND^p, and L^p. N^p denotes the set of computing units, including the set of edge devices E^p and servers in the cloud S^p. ND^p and L^p denote the sets of physical network devices and physical links, respectively. The set of SFC requests serving active cameras in an edge device R_j consists of several requests $\{r_j^i\}$, each of which consists of other parameters, as indicated in Table IV.

The mapping of an SFC request r_j^i to the physical substrate G^p can be divided into two actions: mapping of VNF set VNF_j^i with the corresponding CPU and RAM demands to the physical nodes N^p; and mapping of virtual link set $L_j^{V_i}$ to physical links L^p and network devices ND^p. The mapping process can be expressed as follows:

$$f : R_j \longrightarrow G^p \tag{1}$$

which is separated into two subprocesses

$$g : \text{VNF}_j^i \longrightarrow N^P \tag{1a}$$

$$h : L_j^{V_i} \longrightarrow \left(\text{ND}^P, L^P\right). \tag{1b}$$

We further introduced several binary variables to indicate if the mapping was successful

$$\alpha\left(\text{vnf}_j^{i,k} \rightarrow n_i^p\right) = \begin{cases} 1; n_i^p \in N^p, & \text{if vnf}_j^{i,k} \text{ is mapped to } n_i^p \\ & \text{successfully} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$\alpha\left[l_j^{V_i}(u, v) \rightarrow l_m^p\right] = \begin{cases} 1, & \text{if virtual link } l_j^{V_i}(u, v) \text{ is} \\ & \text{mapped to physical } l_p^m \\ & \text{successfully} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$\delta\left(r_j^i \rightarrow G^p\right) = \begin{cases} 1, & \text{if SFC } r_j^i \text{ is mapped successfully} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

B. Resource Utilization

1) CPU Utilization of Physical Nodes: As shown in Section IV-B, the CPU usage increases linearly depending on the number and type of VNF embedded. Therefore, the CPU usage in the percentage of an edge device e_j^p and a cloud server s_k^p can be modeled as the sum of the CPU utilization in the idle mode and the CPU utilization of all VNFs residing in this computer. This can be expressed as follows:

$$C\left(e_j^p\right) = C\left(\emptyset \rightarrow e_j^p\right) + \sum_{i \in \arg(A)} \sum_{v \in \arg\left(r_j^i\right)} \alpha\left(\text{vnf}_j^{i,v} \rightarrow e_j^p\right) \times C\left(\text{vnf}_j^{i,v} \rightarrow e_j^p\right) \tag{5}$$

$$C\left(s_k^p\right) = C\left(\emptyset \rightarrow s_k^p\right) + \sum_{i \in \arg(A)} \sum_{v \in \arg\left(r_j^i\right)} \alpha\left(\text{vnf}_j^{i,v} \rightarrow s_k^p\right) \times C\left(\text{vnf}_j^{i,v} \rightarrow s_k^p\right). \tag{6}$$

In (5) and (6), $C(\emptyset \rightarrow e_j^p)$ and $C(\emptyset \rightarrow s_k^p)$ express the CPU utilization in idle mode of the edge computer and cloud server, respectively. $C(\text{vnf}_j^{i,v} \rightarrow e_j^p)$ and $C(\text{vnf}_j^{i,v} \rightarrow s_k^p)$ express the CPU utilization of the VNF type v, belonging to SFC i originating from edge e_j^p when embedding on the edge e_j^p or server

TABLE V
POWER PROFILE OF THE HP ENTERPRISE SWITCH [79]

Operating speed	Power(W)
P_{static}	39
P_{10}-10Mpbs per port	0.42
P_{100}-100Mpbs per port	0.48
P_{1000}-1Gpbs per port	0.9

s_k^p, respectively. The actual CPU utilization of individual VNF types is shown in Table III. Besides, the CPU capacity of edge device e_j^p and cloud server s_k^p are denoted as $C_{\text{cap}}(e_j^p)$ and $C_{\text{cap}}(s_k^p)$, respectively.

2) Bandwidth Utilization of Physical Link: Function $B(.)$ is used to demonstrate the bandwidth usage of a link. For example, the bandwidth demand of virtual link $l_j^{V_i}(u, v)$ when being mapped to physical link $l_m^p \in L^p$ is defined as $B[l_j^{V_i}(u, v) \rightarrow l_m^p]$. The total bandwidth usage of a physical link l_m^p is the total bandwidth demands of all virtual links successfully embedded on that link, expressed as follows:

$$B\left(l_m^p\right) = \sum_{i \in \arg(A)} \sum_{\arg\left[l_j^{V_i}(u,v)\right] \in \arg\left(L_j^{V_i}\right)} \alpha\left[l_j^{V_i}(u, v) \rightarrow l_m^p\right] \times B\left[l_j^{V_i}(u, v) \rightarrow l_m^p\right]. \tag{7}$$

The bandwidth capacity of a physical link l_m^p is denoted as $B_{\text{cap}}(l_m^p)$. It should be noted that if $\text{vnf}_j^{i,u}$ and $\text{vnf}_j^{i,v}$ reside at the same node, no physical link is required to interconnect these VNFs; thus, the bandwidth demand on the physical link can be denoted as $B[l_j^{V_i}(u, v) \rightarrow \emptyset] = 0$. The bandwidth demand of each type of VNFs has been shown in Table III.

C. Power Consumption

As energy consumption is significantly proportional to the states of the servers and network devices [37], power saving goal can be achieved by reducing the power consumption of servers and network devices. It is assumed that the computing units and network devices are able to change their states depending on actual loads [77], [78], that is, devices are able to turn off or enter sleep/standby mode after consolidation or offloading while maintaining the ability to wake up when necessary. Power consumption of a device can significantly be reduced when inactive, as shown in the previous work [37], [77], [78]. In this work, the state of a physical device is denoted by the binary function `state`, which is equal to 1 when the device is in the `ON_state` and 0 otherwise (`OFF_state`).

1) $\texttt{state}(s_k^p, t)$ and $\texttt{state}(e_j^p, t)$ denote the working state of the server $s_k^p \in S^p$ and edge device $e_j^p \in E^p$ at time t, respectively.
2) $\texttt{state}(nd_k^p, t)$ denotes the working state of the network device $nd_k^p \in nd^p$ at time t.

1) Power Consumption of Computing Units: As observed earlier in Section IV-B, similar to the CPU model, the power consumption of computing unit also follows the linear trend depending on the number and type of hosted VNFs, that is, the power consumption of a computing unit (i.e., edge computer or cloud server) is in proportion to the number of VNFs running inside the computer. Let us denote $P(\emptyset \rightarrow e_j^p)$ as the baseline power of an edge device e_j^p, $P(\text{vnf}_j^{i,v} \rightarrow e_j^p)$ as the power consumption of $\text{vnf}_j^{i,v}$ when it resides in e_j^p, and $\texttt{state}(e_j^p, t)$ as the working state of the edge e_j^p; then the power consumption of an edge device at time t can be defined as the baseline power and the power of all VNFs located in e_j^p

$$P\left(e_j^p, t\right) = \texttt{state}\left(e_j^p, t\right) \times \left[P\left(\emptyset \rightarrow e_j^p\right) + \sum_{i \in \arg(A)} \sum_{v \in \arg(r_j^i)} \alpha\left(\text{vnf}_j^{i,v} \rightarrow e_j^p\right) \times P\left(\text{vnf}_j^{i,v} \rightarrow e_j^p\right)\right]. \tag{8}$$

Similar to the edge device, we have formulated cloud server power consumption as follows:

$$P\left(s_k^p, t\right) = \texttt{state}\left(s_k^p, t\right) \times \left[P\left(\emptyset \rightarrow s_k^p\right) + \sum_{i \in \arg(A)} \sum_{v \in \arg\left(r_j^i\right)} \alpha\left(\text{vnf}_j^{i,v} \rightarrow s_k^p\right) \times P\left(\text{vnf}_j^{i,v} \rightarrow s_k^p\right)\right]. \tag{9}$$

2) Power Consumption of Network Devices: The power consumption of a network device in the corresponding working states was modeled based on the previous analysis [37], [80], [81]. The state of a network element nd_k^p is defined by binary indicator $\texttt{state}(nd_k^p, t)$, which is 1 if the device is turned on and 0 otherwise. For the simulation section, we have chosen the energy-aware commercial 24-port HP Enterprise [79] with the power modeling, as shown in Table V. A switch can adapt the capacity of its network interfaces to the actual load, so that the power consumption can be saved. In the equation below, $P(\emptyset \rightarrow nd_k^p)$ is denoted as the baseline power of the switch, and P_j^k indicates the power consumption of one of the m network interfaces of nd_k^p with regard to its operating speed (Table V)

$$P\left(nd_k^p, t\right) = \texttt{state}\left(nd_k^p, t\right) \times \left[P\left(\emptyset \rightarrow nd_k^p\right) + \sum_{j=1}^{m} P_j^k\right]. \tag{10}$$

D. Objectives and Constraints

Assume that at time t, a request R_j arrives in the edge $e_j^i \in E^p$, and it is expected that the SFC algorithm must maximize the number of accepted SFCs

$$\textbf{maximize } Z = \sum_{i \in \arg(R_j)} \delta\left(r_j^i \rightarrow G^p\right). \tag{11}$$

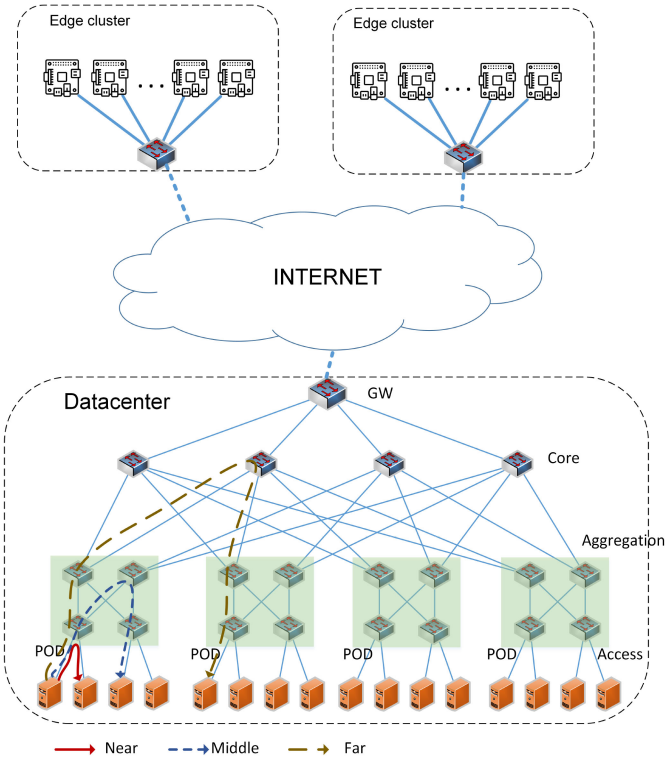

Fig. 5. Overall topology from the edge tier to cloud tier.

The process of mapping R_j into G^p that ensures maximum number of accepted chain may result in multiple solutions as follows:

$$f\left(R_j \longrightarrow G^p\right) = \{x_1, x_2, \ldots, x_n\} = \mathbf{x} \tag{12}$$

where x_i is an individual mapping solution and $\mathbf{x}$ is a vector of solutions. The subobjective is to find the mapping method $x_i \in \mathbf{x}$ that the power consumption equation $P(x_i, t)$ returns the minimum value

$$\text{find } x_i = \arg\min_{x_i} P(x_i, t), \quad x_i \in \mathbf{x} \tag{13a}$$

$$\text{with } P(x_i, t) = \sum_{k \in \arg(E^p)} P\left(e_k^p, t\right) + \sum_{k \in \arg(S^p)} P\left(s_k^p, t\right) + \sum_{k \in \arg(\text{ND}^p)} P\left(nd_k^p, t\right). \tag{13b}$$

The objective functions in (11) and (13) are subject to the following constraints:

$$\alpha\left(\text{vnf}_j^{i,s} \to e_j^p\right) = 1; \ \alpha\left(\text{vnf}_j^{i,d} \to s_k^p\right) = 1 \quad \text{if } \delta\left(r_j^i\right) = 1 \quad \forall r_j^i \in R_j. \tag{14}$$

This constraint ensures that if an SFC r_j^i is mapped to the system, the first VNF $\text{vnf}_j^{i,s}$ (i.e., decoding) must stay at the edge node while the last VNF $\text{vnf}_j^{i,d}$ (i.e., LOS decision) must always stay at the server node

$$\sum_{j \in \arg(E^p)} \alpha\left(\text{vnf}_j^{i,v} \to e_j^p\right) \le 1; \ \sum_{k \in \arg(S^p)} \alpha\left(\text{vnf}_j^{i,v} \to s_k^p\right) \le 1 \quad \text{with } \forall \text{vnf}_j^{i,v} \in r_j^i \quad \forall r_j^i \in R_j. \tag{15}$$

Constraint (15) ensures that every VNF must be regulated such that only one instance of it is mapped to the system at either the edge or cloud

$$\sum_{j \in \arg(E^p)} \sum_{v \in \arg\left(r_j^i\right)} \alpha\left(\text{vnf}_j^{i,v} \to e_j^p\right) - \sum_{v \in \arg\left(r_j^i\right)} \alpha\left(\text{vnf}_j^{i,v} \to e_j^p\right) \le 0 \text{ with } \quad \forall r_j^i \in R_j. \tag{16}$$

Constraint (16) ensures that VNFs belonging to an SFC must stay within an edge node, as mentioned earlier

$$C\left(e_j^p\right) \le C_{\text{cap}}\left(e_j^p\right); \quad C\left(s_k^p\right) \le C_{\text{cap}}\left(s_k^p\right) \quad \text{with} \quad \forall e_j^p \in E^p \quad \forall s_k^p \in S^p. \tag{17}$$

In constraint (17), the CPU usage of edge nodes and server nodes must be less than or equal to their designated CPU capacity

$$B\left(l_m^p\right) \le B_{\text{cap}}\left(l_m^p\right) \quad \text{with } \forall l_m^p \in L^p. \tag{18}$$

Constraint (18) ensures that the total utilization of the physical link must not exceed its capacity $B_{\text{cap}}(l_m^p)$

$$\forall e_k^p, e_k^p \in g\left(\text{VNF}_j^i\right) : \texttt{state}\left(e_k^p, t\right) = 1, \texttt{state}\left(s_k^p, t\right) = 1 \tag{19}$$

$$\forall nd_k^p \in h\left(L_j^{V_i}\right) : \texttt{state}\left(nd_k^p, t\right) = 1. \tag{20}$$

Constraints (19) and (20) require physical nodes that VNFs have been mapped upon, and the network device that physical links are connected to must be turned on, respectively.

The introduced optimization problem is intractable because it requires solving an $\mathcal{NP}$-hard problem. Due to its high complexity, it is not possible to solve the problem directly in a timely manner given the large number of servers and network nodes. Thus, in the next section, RE-SCE—a heuristic SFC embedding strategy, is introduced. The system models in this section are useful for the development of the embedding algorithm, as well as the simulation in the following sections.

VI. Resource and Energy-Aware Service Chain Embedding Strategy

A. Edge–Cloud Network Topology

Fig. 5 shows the edge–cloud topology that is used to deploy the proposed SFC embedding algorithm. The edge clusters are connected to the centralized data center through the Internet. At the data center, we use a k-ary fat-tree topology [82], which consists of the same $(5k^2)/4$ k-port switches and $k^3/4$ servers. As shown in the figure, switches in the fat-tree topology are arranged in three layers, namely, the access, aggregation, and core. Two servers are said to belong to a *near group*, *middle group*, or *far group* if the traffic exchanged between them should go through an access switch only, or aggregation switches or core switch, respectively. The virtual link interconnecting two VNFs in a near group, middle group, and far group are called the *near link*, *middle link*, and *far link*, respectively. It was observed that a near link is shorter, its bandwidth demand is easier to satisfy, and it is more energy-efficient, as the link should traverse less network devices.

B. RE-SCE Strategy

This section describes the RE-SCE strategy proposed in this work that is suitable for the edge–cloud infrastructure. The main objective of RE-SCE is to maximize the number of accepted chains by balancing the resource usage between edge and cloud tier. Then the power consumption of the physical substrate can be minimized by optimizing VNF-placement at DC and the edge.

By combining *offloading* and *mapping* strategies (see Section II-B), the SFC embedding is performed in three steps expressed by three different procedures as follows.

1) *Edge–Cloud VNF Offloader:* This procedure decides if a VNF can stay at the edge or be offloaded to the cloud server based on the resource state of both tiers' devices.
2) *VNFmapping:* After deciding which VNFs reside in the edge or cloud, this procedure is responsible for mapping VNFs to specific physical computing devices. Since the VNFs of an SFC in this work could only be mapped either on the edge device that the SFC originates, or in the data center, this process mainly decided the VNF placement to a specific server at the data center.
3) *Virtual Link Mapping:* Once the locations of the VNFs in the SFC are decided, this procedure takes charge of how virtual links among VNFs are mapped in the physical links.

The entire RE-SCE is a joint resource–energy efficient embedding described in Algorithm 1. The proposed embedding algorithm relies on edge–cloud VNF offloader (E-Coff), *VNFm*, and *VLm* that are presented in Procedures 1–3, respectively.

When a set R_j of SFC requests arrives, RE-SCE first uses *E-Coff* to create an *OffloadingSet* list of feasible placements for VNFs. Subsequently, each mapping feasibility in the list is fetched into the *VNFm* and *VLm* procedure, which are responsible for mapping VNFs and virtual links into physical machines and physical links, respectively. The results of these two procedures are added into *ListPlacement*, which includes all placements that satisfy the constraints. If *ListPlacement* is empty, meaning that there is no satisfied placement due to resource shortage, the algorithm rejects one random r_j^i in the request set R_j in order to reduce the resource demand (lines 10–12). Finally, *RE-SCE* calculates the power consumption of each placement following 13b and sorts all possible placements in the ascending order of power consumption (lines 17 and 18 in Algorithm 1). The first placement with minimum power consumption is chosen as the final solution. `SFCmapping()` (line 21) can do the actual mapping on the system. The detail of the three procedures is illustrated in the following sections.

Algorithm 1 `RE-SCE Algorithm`

1: **Input:** G^P, R_j
2: **Initialization:** *Success* = **false**; *ListPlacement* = $\varnothing$;
3: **while** *Success* = **false**; **do**
4: *OffloadingSet* = `VNFOffloader`(R_j);
5: **for** *feasibility* $\in$ *OffloadingSet* **do**
6: *VNFmResult* = `VNFmapping`(R_j, *feasibility*);
7: *VLmResult* = `VLmapping`(R_j, *VNFmResult*);
8: *ListPlacement* $\leftarrow$ *VLmResult*;
9: **end for**
10: **if** *ListPlacement* = $\varnothing$ **then**
11: r_j^i = `pickRandom`(R_j)
12: $R_j = R_j \setminus r_j^i$;
13: **continue**;
14: **else**
15: *Success* = **true;**
16: **end if**
17: **end while**
18: `calculatePower`(*ListPlacement*);
19: `sorting`(*ListPlacement, PowerAscendingOrder*);
20: *Solution* = `getFirstElement`(*ListPlacement*);
21: G^P = `SFCmapping`(*Solution*);
22: **output:** G^P

C. Edge–Cloud VNF Offloader

When a set of requests R_j arrives in the system, E-Coff finds a feasible placement for each VNF in R_j. It is to be noted that this process solely decides if the VNF placement will take place at the edge device e_j^p and/or in the cloud, while the server location in the cloud is decided by the *VNFm* procedure. The following steps explain the procedure in detail.

1) *EdgeSet* and *CloudSet* keep a possible combination of VNF placements in the edge–cloud for each trial. The algorithm first tries a combination of all $\{\text{vnf}_j^{i,\text{Cap}}, \text{vnf}_j^{i,\text{Dec}}, \text{vnf}_j^{i,\text{Des}}\}$ in the edge, except $\text{vnf}_j^{i,\text{DOS}}$, which is in the cloud.
2) It then tries to gradually move VNFs to the cloud. In each trial, `VNFOffloader()` checks the CPU sufficiency in the cloud. If the condition is satisfied, the procedure moves one additional VNF to the cloud (lines 5–11).
3) All possible combinations are then added in *OffloadingSet*. The combinations are arranged in the order that the one with more VNFs in the edge has higher priority. The `VNFmapping()` is then invoked to find the specific locations of VNFs in the edge–cloud.

Note that this algorithm can be applied not only for the smart traffic monitoring IP camera system but also for any linear service chain.

D. VNF Mapping

Considering the VNF placement feasibility in *OffloadSet* from `VNFOffloader()`, the VNFm algorithm represented in Procedure 2 first finds possible servers that can host VNFs by solving the bin-packing problem related to VNF CPU demands. Like First Fit [43], `serverMapping()` used in VNFm tries to group VNFs into a small number of servers to reduce the power consumption. SFCs are mapped one by one in the series of server from left to right in the fat-tree-based data center. VNFs belonging to each SFC are allocated into the first server that has enough available CPU (line 3). However, this method may put VNFs belonging

Procedure 1 VNFOffloader()

1: **Input:** G^P, R_j
2: **Initialization:** $OffloadingSet = \varnothing$;
3: $EdgeSet = \{vnf_j^{i,Cap}, vnf_j^{i,Dec}, vnf_j^{i,Des}\}$;
4: $CloudSet = \{vnf_j^{i,LOS}\}$;
5: **while** $EdgeSet \setminus vnf_j^{i,Cap} \neq \varnothing$; **do**
6: **if** $\left\{C[EdgeSet \rightarrow e_j^p] \leq C_a(e_j^p)\right\}$ $\wedge\left\{C[CloudSet \rightarrow S^p] \leq C_a(S^p)\right\}$ **then**
7: $OffloadingSet \leftarrow \{EdgeSet \cup CloudSet\}$;
8: $CloudSet \equiv CloudSet \cup$ getLastElement$(EdgeSet)$
9: $EdgeSet \equiv EdgeSet \setminus$ getLastElement$(EdgeSet)$
10: **end if**
11: **end while**
12: **Output:** $OffloadingSet$

Procedure 2 VNFmapping()

1: **Input:** G^P , R_j, $OffloadingSet$
2: **Initialization:** $VNFmResult = \varnothing$;
3: $g^p =$ serverMapping($OffloadingSet$, G^p);
4: **if** getGroup$(vnf_j^{i,S}, vnf_j^{i,D}) \neq near$ **then**
5: **if** $C(vnf_j^{i,D} \rightarrow s_k^p) \leq C_a(s_k^p)$ **then**
6: $g^p \leftarrow (vnf_j^{i,D} \rightarrow s_k^p)$;
7: **else**
8: $indVNFnumber =$
9: ceil$[C(vnf_j^{i,D} \rightarrow s_k^p)/C(vnf_m^{n,ind} \rightarrow s_h^p)]$;
10: **if** getNumberVNF$(vnf_m^{n,ind}) \geq indVNFnumber$ **then**
11: $indVNFset =$
12: getSetVNF$(vnf_m^{n,ind}, indVNFnumber)$;
13: $g^p =$ serverMapping$(indVNFset, g^p)$;
14: $g^p \leftarrow (vnf_j^{i,D} \rightarrow s_k^p)$;
15: **end if**
16: **end if**
17: **end if**
18: $VNFmResult \longleftarrow g^p$;
19: **Output:** $VNFmResult$

to an SFC at a far distance (i.e., far link). Let us denote $vnf_j^{i,S}$ and $vnf_j^{i,D}$ as two neighbor VNFs of an SFC, where traffic goes from $vnf_j^{i,S}$ to $vnf_j^{i,D}$. It is possible that the serverMapping() does not place $vnf_j^{i,S}$ and $vnf_j^{i,D}$ in the same near group (line 5), which causes an increased bandwidth usage in the physical links. In order to improve the network utilization, a migration strategy is introduced (from lines 5–17) as follows.

1) If the server s_k^p hosting $vnf_j^{i,S}$ has sufficient CPU to serve $vnf_j^{i,D}$, then $vnf_j^{i,D}$ is migrated to s_k^p.
2) Otherwise, let us call an *independent* VNF as an only LOS VNF of an SFC that resides in the cloud side, while the other VNFs are at the edge. The algorithm finds a set *indVNFset* of independent $vnf_m^{n,ind} \in s_k^p$ that can be migrated to other locations to free up CPU capacity for $vnf_j^{i,D}$.
3) Moving $vnf_m^{n,ind}$ by using serverMapping() (lines 11–13), and $vnf_j^{i,D}$ is to be migrated to s_k^p. Placing $vnf_j^{i,ind}$ at low priority will not affect the link utilization of the system as it does not have connection between VNFs inside the data center.

Moreover, in order to improve the system resource utilization, VNF consolidation for all active SFCs in the system A^v is performed whenever an SFC r_j^i departs from the system. To reduce the consequences of VNF-interference mentioned in Section II-B2, the algorithm tries to consolidate VNFs belonging to an SFC into one physical machine. This practically reduces the bandwidth consumption caused by traffic between VNFs and, therefore, also reduces the impact of VNF-interference.

E. Virtual Link Mapping

Virtual link mapping (VLm) takes charge of creating connections inside and outside the data center. In our link-mapping algorithm, we call the *external traffic* as the traffic from the edge to the cloud and *internal traffic* as the traffic between VNFs inside the data center.

In our work, the *Elastic Tree* [83] based on the fat-tree topology is applied for energy efficiency in the data center. The elastic tree maintains a minimal number of devices in the ON_State. In a large-scaled data center, putting devices into inactive state would save a significant amount of energy. For the core network, the BFS [84] algorithm is used to find the shortest path between the edge and the data center. The *VLm* procedure is described in the following steps.

1) *Step 1:* For each virtual link (line 4), $l_j^{V_i}(u, v)$ connects $vnf_j^{i,u}$ with $vnf_j^{i,v}$ that exists *external traffic* (line 5), *VLm*() finds the path through the core network by using *BFS*() (line 6), which prioritizes the shortest path; the result returns true if a feasible path is found.
2) *Step 2:* Subsequently, for the *internal traffic*, *Vlm*() identifies the list of ON_STATE switch according to VNFs location *VNFmResult* (line 8). Then, the set of physical link to connect these switches is constructed, which consumes the least available bandwidth and least power (lines 10–12).
3) *Step 3:* Finally, *VLm*() checks if all elements in the list of link have enough bandwidth for mapping (lines 13–17). If both *external traffic* and *internal traffic* are satisfied (line 22), the SFC is accepted (line 23).

VII. Performance Evaluation

A. Simulation Scenarios

To evaluate the performance of the proposed algorithm on a large IoT system, a simulation was conducted. In the field of edge–cloud, there are several popular simulators, such as IFogSim [85] and FogTorch [86]. However, as IFogSim only allows creating the tree-based topology [87] and other tools are deprecated, we developed *BK-EdgeCloud*, a flow level event-based Java simulator that simulates the entire system from the edge to the cloud environment. BK-EdgeCloud allows evaluating different key performance indicators in detail, such as energy consumption, acceptance ratio, and utilization.

Procedure 3 `VLmapping()`

Input: G^P, *VNFmResult*
Initialization: *VLmResult* = $\varnothing$;
for $r_j^i \in R_j$ **do**
  for $l_j^{V_i}(u, v) \in L_j^{V_i}$ **do**
    if $\exists$ *ExternalLink* **then**
      ExLinkResult = `BFS`[$l_j^{V_i}(u, v)$];
    end if
    listSwitchIn = `VLim`[*VNFmResult*, $l_j^{V_i}(u, v)$];
    //Construct link
    ListLinkIn = `constructLink`(*listSwitchIn*);
    ListLinkIn = `sort`(*listSwitchIn, key = cap, order = asc*);
    for *sLink* $\in$ *listLinkIn* **do**
      if *sLink* = *satisfied* **then**
        result $\leftarrow$ `map`(*link, sLink*);
      end if
    end for
  end for
  if $|result| = |L_j^{V_i}|$ **then**
    InLinkResult = **true**;
  end if
  if *ExLinkResult* $\wedge$ *InLinkResult* = **true then**
    VLmResult $\leftarrow$ *result*;
  end if
end for
Output: *VLmResult*

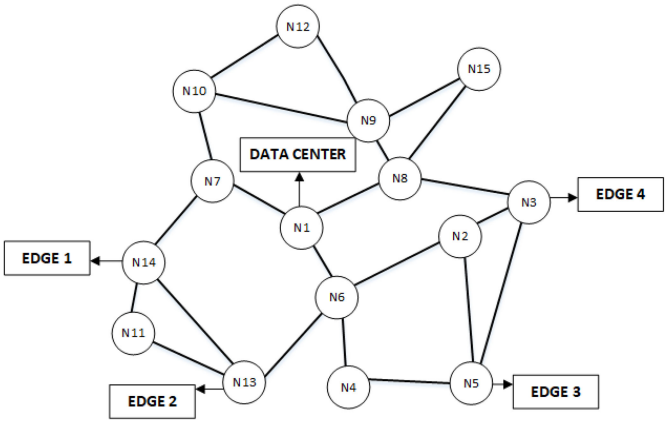

Fig. 6. Atlanta city network [88].

1) Topology: We selected the topology of Atlanta city provided by IBM Watson Research, New York, shown in Fig. 6, as the core network. All nodes from $N1$ to $N15$ represent the backbone switches. The data center was connected to node $N1$, four edge clusters *EDGE*1, *EDGE*2, *EDGE*3, and *EDGE*4 are connected to $N14$, $N13$, $N5$, and $N3$, respectively. At the data center, a k-ary Fat-tree topology with $k = 10$, which can serve 250 servers, was used. We assumed the bandwidth capacity of the physical link from an edge node to the edge gateway to be 100 and 20 Gb/s of the core network, and 1 Gb/s inside DC. The power profile of switches is taken from [79], as shown in Table V, while the power profile of the servers is modeled as

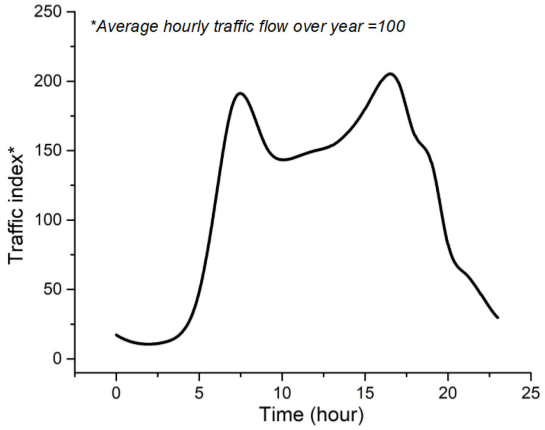

Fig. 7. Vehicle unit distribution recorded in a weekday, U.K., 2018 [89].

previously described (see Section V-C1). *EDGE*1 and *EDGE*2 have 100 Rpis each, while *EDGE*3 and *EDGE*4 have 50 Rpis each. Therefore, the number of edge computers account for 300 units. At each edge node, the Pis are connected by a star topology to a central gateway, as previously shown in Fig. 5.

2) Camera Requests: In simulation scenarios, it is assumed that each Rpi can host up to seven cameras, which means that a total number of 2100 cameras are deployed massively in the streets of the city. They are activated depending on the current traffic situation. As shown in Fig. 7, we considered a one-day average traffic density in the U.K. in 2018 as reported by the British government [89]. The traffic index is the normalized traffic volume in terms of the number of vehicles relative to the mean value of 100. It is further assumed that the number of active cameras is proportional to the road traffic density and follows the Poisson distribution, that is, the number of camera activation requests or the equivalent number of SFC requests is Poisson-distributed and in proportion with the traffic density. The maximum arrival rate (λ) is 2100, while the minimum λ is 120. The average duration of an SFC request is 2 h and exponentially distributed.

3) Performance Criteria: The three following criteria are used to evaluate the performance of the proposed approach.

1) *Resource efficiency*, which shows how limited resources are utilized to serve a maximized number of SFC requests.
2) *Energy efficiency*, which expresses how much energy is consumed to serve a number of SFCs. The less energy is consumed for an SFC, the more energy efficient the system is.
3) *Complexity*, which is the cost when performing measures to improve the resource and energy efficiency, such as VNF offloading or migration.

As addressed in Section VI, in order to meet the requirements on resource and energy efficiency, the proposed RE-SCE combines an *offloading* strategy that balances resource utilization in the edge–cloud (E-Coff) with the resource and energy-efficient VNFm and virtual link *mapping* (VLm) algorithms. The performance of RE-SCE, including E-Coff and VNFm, is compared with two existing offloading algorithms,

TABLE VI
ALGORITHMS SELECTED FOR COMPARISON WITH RE-SCE

Offloading \ Mapping	Least-Loaded [42]	First-Fit [40]	**VNFm**
Baseline [36]	BL-LL	BL-FF	BL-VNFm
SFCCM [37]	SFCCM-LL	SFCCM-FF	SFCCM-VNFm
E-Coff	ECoff-LL	ECoff-FF	**RE-SCE**

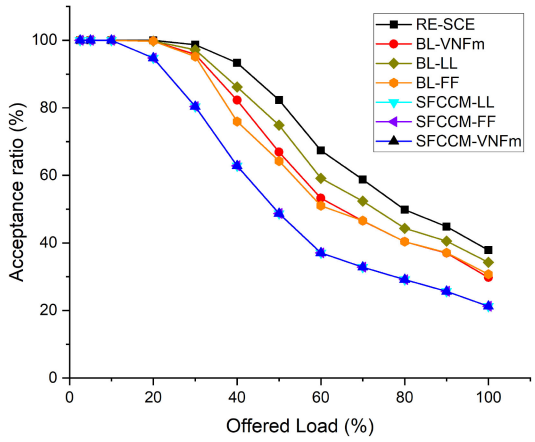

Fig. 8. Acceptance ratio.

namely, SFCCM [40] and Baseline [39], and two existing mapping algorithms, namely, Least-Loaded [45] and First-Fit [43], as shown in Table VI.

B. Simulation Setups

A simulation was performed from scratch in Java. All entities of the network topology in Fig. 6 include edge devices, servers, network switches, and physical links that were covered in the simulation. The simulation was performed on the SFC-level so that factors, such as the packet collision and packet drop rate were neglected. A discrete event, that is, an SFC request, arrived in the system containing several attributes of its demands, such as CPU and bandwidth. Incoming SFC requests were handled in a first-in–first-out manner. The simulations were repeated in several runs and the average over the simulation runs have been reported in the figures and tables in accordance with the *replicate-delete method* [90]. An implementation of the simulator can be found at our GitHub repository [91].

C. Simulation Results

This section shows results of the proposed RE-SCE in comparison with the aforementioned mapping and offloading strategies.

1) Resource Efficiency: Fig. 8 shows the acceptance ratio of the SFC embedding algorithms, which is the ratio between the number of accepted incoming SFCs and the total SFC requests. Fig. 9 presents the system utilization. These two metrics were investigated under varied offered load. The two figures show that as the offered load increase, the acceptance ratio decreased and the system utilization increased accordingly but not linearly with the offered load. This implies that

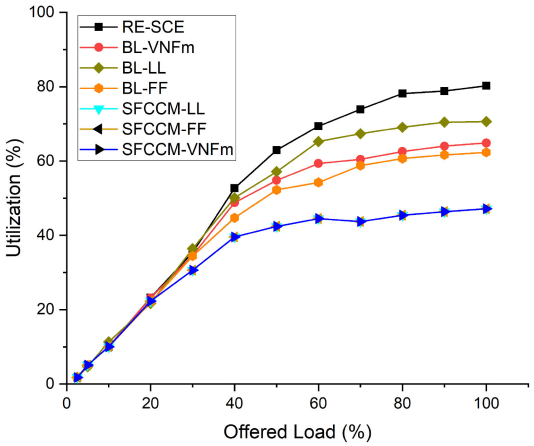

Fig. 9. System utilization under varied load situations.

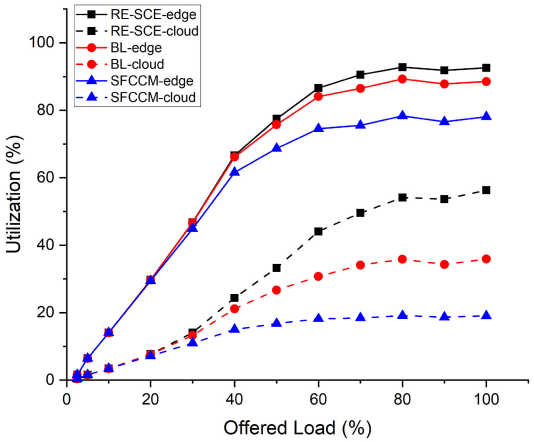

Fig. 10. Comparison of resource utilization in the edge and among algorithms.

physical resources cannot be fully utilized. SFC requests were rejected even if some resources are available (i.e., utilization is less than 100%) in the physical substrate due to the imperfection of the SFC embedding algorithms.

Overall, RE-SCE utilized the edge and cloud resources better than the other two BL and SFCCM algorithms owing to its offloading strategy. The results show that a higher number of requests are accepted in RE-SCE when the offered load increases following more demands for turning on cameras in the rush hour. On the other hand, SFCCM did not perform well, regardless of the mapping algorithms used. Even though BL and SFFCM were combined with VNFm, their results could not match those of RE-SCE due to the strategy of preferring "low" cost resources, putting the edge resources into exhaustion, as well as the lack of migration process between the edge–cloud. As shown in Fig. 9, the system utilization under RE-SCE can reach up to 80% while that of SFCCM and Baseline with Least-Loaded can only be around 40% and 70%, respectively. In Fig. 10, the edge and cloud utilization of these algorithms are shown separately. It is to be noted that BL and SFCCM are combined with VNFm to have a fair comparison with RE-SCE in this scenario. Thanks to the balance between the edge–cloud resources, as well as the migration strategy, RE-SCE has a higher utilization in the cloud, resulting in

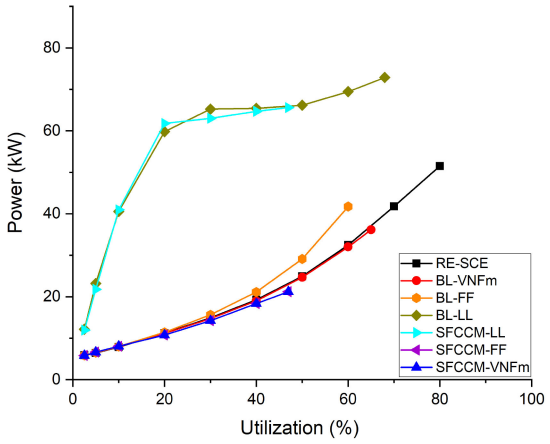

Fig. 11. System power consumption.

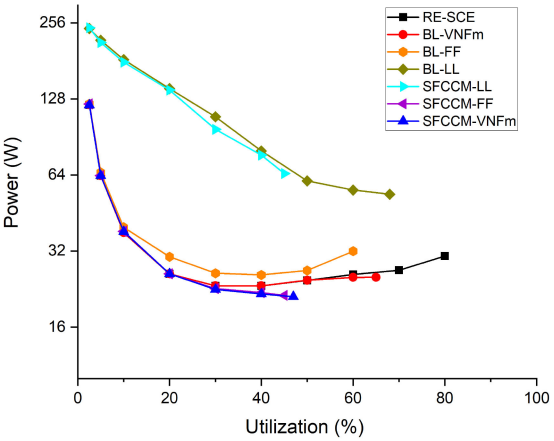

Fig. 12. Average power consumption of an SFC.

higher overall utilization and a higher acceptance ratio. As can be seen, the utilization of RE-SCE in the cloud can be as high as 55%, while the utilization of BL and SFCCM in the cloud is remarkably lower, which accounts for 30% and 20%, respectively. One important observation here is that the offloading strategies have an impact on the resource efficiency of the physical edge–cloud system.

2) Energy Efficiency: Fig. 11 shows the total system power consumption for different SFC embedding strategies under various utilization scenarios. The results show that mapping strategies have an impact on the system power consumption, that is, applying different mapping algorithms with the same offloading strategy could produce a different energy consumption profile. For instance, as can be seen in Fig. 11, Baseline is the most power-consuming algorithm when combined with Least Loaded (i.e., up to 72 kW), while its power consumption is only a half when combined with First Fit or VNFm (40 kW). Since the Least Loaded algorithm maps VNF requests in the least loaded servers, the load tends to be distributed evenly among all servers in the cloud, leading to a high number of active servers in the data center with the consequence of high power consumption. In contrast, First Fit and VNFm try to map VNF requests in a minimum number of active servers, so that idle servers can be put in the standby mode. Thus, the

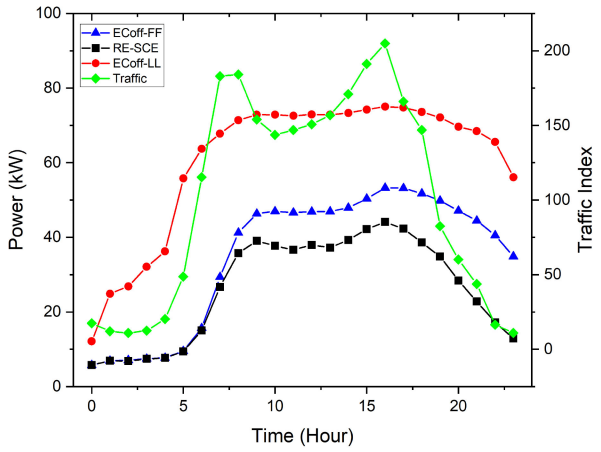

Fig. 13. System power consumption varied with traffic intensity.

strategy applied in VNFm and First Fit is clearly more energy efficient.

The above argument is further elaborated when calculating the power consumption per SFC, as shown in Fig. 12. The results presented in the figure indicate that under the same offloading strategy (i.e., BL, SFCCM, or RE-SCE), VNFm is the most energy-efficient mapping algorithm. For instance, under the same utilization of 47%, the average power consumption of an SFC with SFCCM-LL can be up to 66 W, while that of SFCCM-VNFm is only 22 W. It is also observed that the power consumption of an SFC in low-loaded situation is higher than that of higher loaded. As the utilization increases, the SFC power consumption gradually decreases, especially in the case of BL-LL and SFCCM-LL. In contrast, the SFC power consumption of RE-SCE remains nearly the same as the system utilization is higher than 25%, which implies that when deploying RE-SCE the power consumption of the physical edge–cloud substrate stays nearly linear with the number of accepted SFCs, following the energy proportional property. On the other hand, when utilization reaches 55% and above, the power consumption slightly increases. In highly loaded situation, in order to maintain high utilization, VNFs are gradually offloaded from the edge to cloud, resulting in higher power consumption as more VNFs are located in the cloud.

We further investigated the dependency of power consumption on the road traffic intensity under different mapping strategies, namely, VNFm, First Fit, and Least Loaded, where First Fit and Least Loaded make use of the newly proposed offloading E-Coff algorithm. The green curve in Fig. 13 presents the traffic density in the U.K. in one day (also shown in Fig. 7). As can be seen, the system power consumption varies with traffic density, which also reflects the energy proportional property of the mapping algorithms. RE-SCE is the most energy-efficient strategy, as it consumes the least power under any load situation. Also, the power consumption can adapt very well to the actual traffic situation. In contrast, the Least Loaded is the worst energy-efficient strategy.

3) Complexity: We further investigated the complexity of RE-SCE. As addressed in [37], as requests continuously join and leave the system, resource fragmentation may occur, which consequently leads to temporary degradation of resource

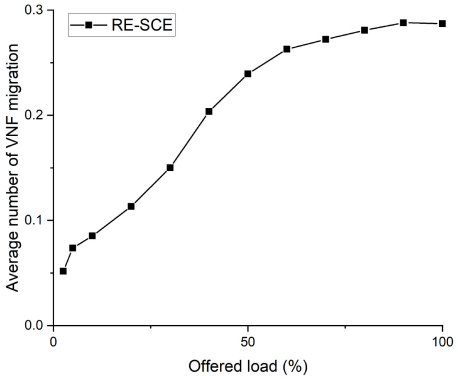

Fig. 14. Average number of migrations per VNF.

efficiency. In order to cope with this resource dynamic problem, consolidations of VNF's containers are regularly performed by RE-SCE to transfer accepted VNFs to new, more efficient locations. This mechanism helps rearrange available resources in physical machines to increase the acceptance ratio and system utilization. However, the relocation of VNFs also increases the system complexity. Fig. 14 shows the average number of required migrations per VNF of an SFC request depending on the offered load. In a low load situation, each VNF is moved to a new location 0.1 times on the average. As the load increases, it is more difficult to find an appropriate place to map a new VNF request, thus more migrations are required to optimize available resource.

D. Discussions

The aforementioned evaluation shows that RE-SCE is the most efficient strategy. RE-SCE performs very well in terms of energy efficiency, as shown in Figs. 11 and 12. While some other strategies, such as BL-VNFm or SFCCM-VNFm might be a little better in terms of energy performance, RE-SCE outperforms the others in terms of resource efficiency (Figs. 8 and 9) owing to the tradeoff between offloading and mapping methods deployed in E-Coff and VNFm. In general, RE-SCE is the best strategy that can accept a maximum number of SFC requests while keeping the power consumption low. In other words, it can balance energy efficiency and system utilization very well, in the sense that the energy consumption is one of the lowest, while the system utilization is the highest. On the other hand, RE-SCE increases the system complexity by performing VM migration as the cost to improve resource and energy efficiency (Fig. 14).

In general, it is observed that the offloading algorithms are important to balance resource utilization between the edge and the cloud and improve resource utilization. VNFm algorithms, on the other hand, are considered as a fine-grained algorithm to decide the precise location of the VNF in a server. The mapping policy has an impact on energy efficiency, as well as the resource utilization. By combining offloading and mapping, a good tradeoff between resource and energy efficiency can be reached.

VIII. Conclusion and Future Work

In this article, we have first addressed some issues in modeling resource and energy utilization for IoT applications in edge–cloud paradigms. A smart traffic monitoring IP camera system was deployed as a use case for a realistic modeling of a service chain. The system was implemented in our testbed, which was designed and developed specifically to model and investigate the resource and energy utilization of SFC embedding strategies. A dynamic energy-aware SFC strategy in the edge–cloud paradigm for IoT applications was then proposed. Results showed that our strategy outperforms some of the existing approaches in terms of resource and energy efficiency.

The evaluation in this work is applicable to homogeneous, linear SFCs. As the current and future work, it would be interesting to develop flexible and cost-efficient SFC embedding strategies for dynamic, general SFC requests with any topology to support a wide range of IoT services and applications. Under this circumstance, it is required that the SFC embedding algorithm be more adaptive to the edge–cloud system condition and have a higher level of automation to better accommodate the current and upcoming demands. For IoT applications with predictable load patterns in a time period, ML approaches may provide load and resource prediction. Then, the SFC embedding algorithm can benefit from such a prediction. Appropriate ML approaches are a good solution to reduce complexity and improve efficiency of the ML-based SFC embedding algorithm. A comparison of our proposed solution in conjunction with ML for load prediction can be considered for future research work.

References

[1] F. Shrouf, J. Ordieres, and G. Miragliotta, "Smart factories in industry 4.0: A review of the concept and of energy management approached in production based on the Internet of Things paradigm," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manag.*, 2014, pp. 697–701.

[2] W. Elghazel, J. Bahi, C. Guyeux, M. Hakem, K. Medjaher, and N. Zerhouni, "Dependability of wireless sensor networks for industrial prognostics and health management," *Comput. Ind.*, vol. 68, pp. 1–15, Apr. 2015.

[3] E. Hajrizi and X. Krasniqi, "Use of IoT technology to drive the automotive industry from connected to full autonomous vehicles," *IFAC-PapersOnLine*, vol. 49, no. 29, pp. 269–274, 2016.

[4] J.-C. Zhao, J.-F. Zhang, Y. Feng, and J.-X. Guo, "The study and application of the IoT technology in agriculture," in *Proc. 3rd Int. Conf. Comput. Sci. Inf. Technol.*, vol. 2, 2010, pp. 462–465.

[5] S. Fang *et al.*, "An integrated system for regional environmental monitoring and management based on Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1596–1605, May 2014.

[6] L. Catarinucci *et al.*, "An IoT-aware architecture for smart healthcare systems," *IEEE Internet Things J.*, vol. 2, no. 6, pp. 515–526, Dec. 2015.

[7] C. Mahapatra, A. K. Moharana, and V. C. M. Leung, "Energy management in smart cities based on Internet of Things: Peak demand reduction and energy savings," *Sensors*, vol. 17, no. 12, p. 2812, 2017.

[8] M. M. Rana and W. Xiang, "IoT communications network for wireless power transfer system state estimation and stabilization," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 4142–4150, Oct. 2018.

[9] K. Kaur, S. Garg, G. S. Aujla, N. Kumar, J. J. P. C. Rodrigues, and M. Guizani, "Edge computing in the industrial Internet of Things environment: Software-defined-networks-based edge-cloud interplay," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 44–51, Feb. 2018.

[10] T. N. Huu *et al.*, "Modeling and experimenting combined smart sleep and power scaling algorithms in energy-aware data center networks," *Simulat. Model. Pract. Theory*, vol. 39, pp. 20–40, Dec. 2013.

[11] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future Internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 2, pp. 223–244, 2nd Quart., 2011.

[12] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[13] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct.–Dec. 2009.

[14] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," Sophia Antipolis, France, Eur. Telecommun. Stand. Inst., White Paper, 2015.

[15] NFV-ISG, "Network functions virtualisation (NFV): Network operator perspectives on industry progress," Sophia Antipolis, France, ETSI, White Paper, 2013. [Online]. Available: http://portal.etsi.org/NFV/NFV WhitePaper2.pdf

[16] W. John *et al.*, "Research directions in network service chaining," in *Proc. IEEE SDN Future Netw. Services (SDN4FNS)*, 2013, pp. 1–7.

[17] J. Halpern and C. Pignataro, "Service function chaining (SFC) architecture," Internet Eng. Task Force, Fremont, CA, USA, RFC 7665, 2015.

[18] J. Pan and J. McElhannon, "Future edge cloud and edge computing for Internet of things applications," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 439–449, Feb. 2018.

[19] OpenStack. *OpenStack: Open Source Cloud Computing Infrastructure*. Accessed: Jan. 19, 2021. [Online]. Available: http://www.openstack.org/

[20] Kubernetes. *Kubernetes: Production-Grade Container Orchestration*. Accessed: Jan. 19, 2021. [Online]. Available: https://kubernetes.io

[21] J. Guo, B. Song, S. Chen, F. R. Yu, X. Du, and M. Guizani, "Context-aware object detection for vehicular networks based on edge-cloud cooperation," *IEEE Internet Things J.*, Vol. 7, no. 7, pp. 5783–5791, Jul. 2020.

[22] S. Garg, S. Guo, V. Piuri, K.-K. R. Choo, and B. Raman, "Guest editorial special issue on edge-cloud interplay based on SDN and NFV for next-generation IoT applications," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5690–5694, Jul. 2020.

[23] S. Nastic, S. Sehic, D.-H. Le, H.-L. Truong, and S. Dustdar, "Provisioning software-defined IoT cloud systems," in *Proc. Int. Conf. Future Internet Things Cloud*, 2014, pp. 288–295.

[24] CloudStack. *Apache CloudStack: Open Source Cloud Computing*. Accessed: Jan. 19, 2021. [Online]. Available: http://cloudstack.apache.org/

[25] K. Ha and M. Satyanarayanan, "Openstack++ for cloudlet deployment," Sch. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Rep. CMU-CS-15-123, 2015. [Online]. Available: http://elijah.cs.cmu.edu/DOCS/CMU-CS-15–123.pdf

[26] K. Kritikos and P. Skrzypek, "A review of serverless frameworks," in *Proc. IEEE/ACM Int. Conf. Utility Cloud Comput. Companion (UCC Companion)*, 2018, pp. 161–168.

[27] R. Cziva, S. Jouet, K. J. White, and D. P. Pezaros, "Container-based network function virtualization for software-defined networks," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, 2015, pp. 415–420.

[28] L. Peterson *et al.*, "Central office re-architected as a data center," *IEEE Commun. Mag.*, vol. 54, no. 10, pp. 96–101, Oct. 2016.

[29] P. Hu, W. Chen, C. He, Y. Li, and H. Ning, "Software-defined edge computing (SDEC): Principle, open IoT system architecture, applications and challenges," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5934–5945, Jul. 2020.

[30] C. Mouradian *et al.*, "An IoT platform-as-a-service for NFV based-hybrid cloud/fog systems," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6102–6115, Jul. 2020.

[31] Q. Li, H. Yao, T. Mai, C. Jiang, and Y. Zhang, "Reinforcement-learning and belief-learning-based double auction mechanism for edge computing resource allocation," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5976–5985, Jul. 2020.

[32] OpenMANO. *Open Source Management and Orchestration (MANO)*. Accessed: Jan. 19, 2021. [Online]. Available: https://osm.etsi.org/

[33] OPNFV. *Open Platform for NFV (OPNFV)*. Accessed: Jan. 19, 2021. [Online]. Available: https://www.opnfv.org/

[34] ONOS. *Open Network Operating System (ONOS) SDN Controller for SDN/NFV Solutions*. Accessed: Jan. 19, 2021. [Online]. Available: https://opennetworking.org/onos/

[35] S. Wang, M. Zafer, and K. K. Leung, "Online placement of multi-component applications in edge computing environments," *IEEE Access*, vol. 5, pp. 2514–2533, 2017.

[36] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 4th Quart., 2013.

[37] T. M. Nam, N. H. Thanh, H. T. Hieu, N. T. Manh, N. Van Huynh, and H. D. Tuan, "Joint network embedding and server consolidation for energy–efficient dynamic data center virtualization," *Comput. Netw.*, vol. 125, pp. 76–89, Oct. 2017.

[38] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2015, pp. 1346–1354.

[39] F. B. Jemaa, G. Pujolle, and M. Pariente, "QoS-aware VNF placement optimization in edge-central carrier cloud architecture," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2016, pp. 1–7.

[40] D. Zhao, D. Liao, G. Sun, and S. Xu, "Towards resource-efficient service function chain deployment in cloud-fog computing," *IEEE Access*, vol. 6, pp. 66754–66766, 2018.

[41] T. Yang *et al.*, "Two-stage offloading optimization for energy–latency tradeoff with mobile edge. computing in maritime Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5954–5963, Jul. 2020.

[42] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 10, pp. 2179–2192, Oct. 2019.

[43] D. S. Johnson, "Near-optimal bin packing algorithms," Ph.D. dissertation, Dept. Math., Massachusetts Inst. Technol., Cambridge, MA, USA, 1973.

[44] L. Dinh-Xuan, M. Seufert, F. Wamser, P. Tran-Gia, C. Vassilakis, and A. Zafeiropoulos, "Performance evaluation of service functions chain placement algorithms in edge cloud," in *Proc. 30th Int. Teletraffic Congr. (ITC)*, vol. 1, 2018, pp. 227–235.

[45] V. Eramo, A. Tosti, and E. Miucci, "Server resource dimensioning and routing of service function chain in NFV network architectures," *J. Electr. Comput. Eng.*, vol. 2016, no. 9, 2016, Art. no. 7139852. [Online]. Available: https://www.hindawi.com/journals/jece/2016/7139852/

[46] C. Zeng, F. Liu, S. Chen, W. Jiang, and M. Li, "Demystifying the performance interference of co-located virtual network functions," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2018, pp. 765–773.

[47] Q. Zhang, F. Liu, and C. Zeng, "Adaptive interference-aware VNF placement for service-customized 5G network slices," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2019, pp. 2449–2457.

[48] G. Sun, R. Zhou, J. Sun, H. Yu, and A. V. Vasilakos, "Energy-efficient provisioning for service function chains to support delay-sensitive applications in network function virtualization," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6116–6131, Jul. 2020.

[49] Y. Xiao *et al.*, "NFVdeep: Adaptive online service function chain deployment with deep reinforcement learning," in *Proc. Int. Symp. Qual. Service*, 2019, pp. 1–10.

[50] J. Zhou, P. Hong, J. Pei, and D. Li, "Multi-task deep learning based dynamic service function chains routing in SDN/NFV-enabled networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–6.

[51] Z. Lv and W. Xiu, "Interaction of edge-cloud computing based on SDN and NFV for next generation IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5706–5712, Jul. 2020.

[52] G. Sun, Z. Xu, H. Yu, X. Chen, V. Chang, and A. V. Vasilakos, "Low-latency and resource-efficient service function chaining orchestration in network function virtualization," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5760–5772, Jul. 2020.

[53] M. Adhikari, M. Mukherjee, and S. N. Srirama, "DPTO: A deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5773–5782, Jul. 2020.

[54] P. Jin, X. Fei, Q. Zhang, F. Liu, and B. Li, "Latency-aware VNF chain deployment with efficient resource reuse at network edge," in *Proc. IEEE INFOCOM*, 2020, pp. 267–276.

[55] M. Dolati, S. B. Hassanpour, M. Ghaderi, and A. Khonsari, "DeepViNE: Virtual network embedding with deep reinforcement learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2019, pp. 879–885.

[56] P. T. A. Quang, Y. Hadjadj-Aoul, and A. Outtagarts, "A deep reinforcement learning approach for VNF forwarding graph embedding," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1318–1331, Dec. 2019.

[57] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou, "Optimal VNF placement via deep reinforcement learning in SDN/NFV-enabled networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 263–278, Feb. 2020.

[58] H. Hawilo, M. Jammal, and A. Shami, "Network function virtualization-aware orchestrator for service function chaining placement in the cloud," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 643–655, Mar. 2019.

[59] U. Singh and I. Chana, "Enhancing energy efficiency in IoT (Internet of Thing) based application," in *Proc. 2nd Int. Conf. Emission Electron. (ICEE)*, 2014, pp. 1–4.

[60] A. Orsino, G. Araniti, L. Militano, J. Alonso-Zarate, A. Molinaro, and A. Iera, "Energy efficient IoT data collection in smart cities exploiting D2D communications," *Sensors*, vol. 16, no. 6, p. 836, 2016.

[61] W. Li *et al.*, "Complexity and algorithms for superposed data uploading problem in networks with smart devices," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5882–5891, Jul. 2020.
[62] S. H. Alsamhi, O. Ma, M. S. Ansari, and Q. Meng, "Greening Internet of things for smart everythings with a green-environment life: A survey and future prospects," *Telecommun. Syst.*, vol. 72, no. 4, pp. 609–632, 2019.
[63] I. Shallari and M. O'Nils, "From the sensor to the cloud: Intelligence partitioning for smart camera applications," *Sensors*, vol. 19, no. 23, p. 5162, 2019.
[64] P. Corcoran and A. Andrae, "Emerging trends in electricity consumption for consumer ICT," College Eng. Inf., Nat. Univ. Ireland, Galway, Ireland, Rep., 2013.
[65] Science for Environment Policy, *Exploring the Links Between Energy Efficiency and Resource Efficiency*, Eur. Comm., DG Environ. Sci. Commun. Unit, Univ. West England, Bristol, U.K., 2015. [Online]. Available: https://data.europa.eu/doi/10.2779/068285
[66] H. Zhu and C. Huang, "EdgePlace: Availability-aware placement for chained mobile edge applications," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 11, 2018, Art. no. e3504.
[67] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, 2008.
[68] Z. Zhou, Q. Wu, and X. Chen, "Online orchestration of cross-edge service function chaining for cost-efficient edge computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1866–1880, Aug. 2019.
[69] H. Cao *et al.*, "An efficient energy cost and mapping revenue strategy for interdomain NFV-enabled networks," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5723–5736, Jul. 2020.
[70] A. Leivadeas, M. Falkner, I. Lambadaris, and G. Kesidis, "Optimal virtualized network function allocation for an SDN-enabled cloud," *Comput. Stand. Interfaces*, vol. 54, pp. 266–278, Nov. 2017.
[71] P. N. The *et al.*, "Real time ARM-based traffic level of service classification system," in *Proc. 13th Int. Conf. Electr. Eng./Electron. Comput. Telecommun. Inf. Technol. (ECTI-CON)*, 2016, pp. 1–5.
[72] Prometheus. *Prometheus—Monitoring System & Time Series Database*. Accessed: Jan. 19, 2021. [Online]. Available: https://prometheus.io/
[73] Magnum. *OpenStack Magnum Project: Container Orchestration Engine*. Accessed: Jan. 19, 2021. [Online]. Available: https://docs.openstack.org/magnum/
[74] Kuryr. *OpenStack Kuryr Project: Kubernetes Integration With OpenStack Networking*. Accessed: Jan. 19, 2021. [Online]. Available: https://docs.openstack.org/kuryr-kubernetes
[75] Htop. *Htop: Interactive Process Viewer*. Accessed: Jan. 19, 2021. [Online]. Available: https://linux.die.net/man/1/htop
[76] Vnstat. *Vnstat: A Console-Based Network Traffic Monitor*. Accessed: Jan. 19, 2021. [Online]. Available: https://linux.die.net/man/1/vnstat
[77] R. Bolla, R. Bruschi, A. Cianfrani, and M. Listanti, "Enabling backbone networks to sleep," *IEEE Netw.*, vol. 25, no. 2, pp. 26–31, Mar./Apr. 2011.
[78] R. Basmadjian, "Flexibility-based energy and demand management in data centers: A case study for cloud computing," *Energies*, vol. 12, no. 17, p. 3301, 2019.
[79] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "Energy aware network operations," in *Proc. IEEE INFOCOM Workshops*, 2009, pp. 1–6.
[80] N. P. Ngoc *et al.*, "A new power profiling method and power scaling mechanism for energy-aware NetFPGA gigabit router," *Comput. Netw.*, vol. 78, pp. 4–25, Feb. 2015.
[81] N. H. Thanh *et al.*, "ECODANE: A customizable hybrid testbed for green data center networks," in *Proc. Int. Conf. Adv. Technol. Commun. (ATC)*, 2013, pp. 312–317.
[82] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.
[83] B. Heller *et al.*, "ElasticTree: Saving energy in data center networks," in *Proc. NSDI*, vol. 10, 2010, pp. 249–264.
[84] S. S. Skiena, *The Algorithm Design Manual*. London, U.K.: Springer, 2008.
[85] H. Gupta, A. Dastjerdi, S. Ghosh, and R. Buyya, "iFogSim: A Toolkit for modeling and simulation of resource management techniques in Internet of things, edge and fog computing environments," *Softw. Pract. Exp.*, vol. 47, no. 9, pp. 1275–1296, 2017.
[86] A. Brogi and S. Forti, "QoS-aware deployment of IoT applications through the fog," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1185–1192, Oct. 2017.
[87] S. Margariti, V. Dimakopoulos, and G. Tsoumanis, "Modeling and simulation tools for fog computing—A comprehensive survey from a cost perspective," *Future Internet*, vol. 12, no. 5, p. 89, 2020. 2020.
[88] *SNDlib.* Accessed: Jan. 19, 2021. [Online]. Available: http://sndlib.zib.de/
[89] GOV.UK. *Report TRA0307: Traffic Distribution on All Roads by Time of Day and Day of the Week in Great Britain*. Accessed: Jan. 19, 2021. [Online]. Available: https://www.gov.uk/government/statistical-data-sets/road-traffic-statistics-tra
[90] A. M. Law, W. D. Kelton, and W. D. Kelton, *Simulation Modeling and Analysis*, vol. 3. New York, NY, USA: McGraw-Hill, 2000,
[91] NFVteam-HUST. *Edge-Cloud Simulation for Traffic Camera Application*. Accessed: Jan. 19, 2021. [Online]. Available: https://github.com/FILHUST/BKedgecloud