Research Article

Task Scheduling for Edge Computing with Agile VNFs On-Demand Service Model toward 5G and Beyond

Chia-Wei Tseng¹, **Fan-Hsun Tseng**², **Yao-Tsung Yang**¹,
Chien-Chang Liu¹, and **Li-Der Chou**¹

¹Department of Computer Science and Information Engineering, National Central University, Taoyuan 32001, Taiwan

²Department of Technology Application and Human Resource Development, National Taiwan Normal University, Taipei 10610, Taiwan

Correspondence should be addressed to Li-Der Chou; cld@csie.ncu.edu.tw

Received 16 April 2018; Accepted 28 June 2018; Published 11 July 2018

Academic Editor: Chun-Hsian Huang

Copyright © 2018 Chia-Wei Tseng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The demand for satisfying service requests, effectively allocating computing resources, and providing service on-demand application continuously increases along with the rapid development of the Internet. Edge computing is used to satisfy the low latency, network connection, and local data processing requirements and to alleviate the workload in the cloud. This paper proposes a gateway-based edge computing service model to reduce the latency of data transmission and the network bandwidth from and to the cloud. An on-demand computing resource allocation can be achieved by adjusting the task schedule of the edge gateway via the lightweight virtualization technology, Docker. The edge gateway can also process the service requests in the local network. The proposed edge computing service model not only eliminates the computation burden of the traditional cloud service model but also improves the operation efficiency of the edge computing nodes. This model can also be used for various innovation applications in the cloud-edge computing environment for 5G and beyond.

1. Introduction

The rapid development of cloud computing, mobile broadband networks, and Internet of Things (IoT) has been accompanied by a considerable demand for network resources allocation, data processing, and service management and has inevitably changed the traditional network infrastructure. Both software-defined network (SDN) and network function virtualization (NFV) technologies not only transform the network infrastructure from complicate physical entities into virtual and programmable nodes but also introduce significant changes to the development of the information and communications technology (ICT) [1, 2].

NFV represents a core structural change in how a telecommunication infrastructure is deployed [3]. As shown in Figure 1, NFV aims to solve the problems related to the distribution of network device resources. Specifically, NFV uses virtualization technology to integrate different types of devices that are located on the data center, network nodes, and customer-premises equipment (CPE), including

routers, switches, firewalls, and intrusion-detection systems, into standard or general hardware equipment. In this way, NFV can efficiently reduce the purchase cost of network physical devices and the related network maintenance costs. As the logical outcome of NFV, virtual network functions (VNF) are virtualized tasks that are formerly performed by proprietary, dedicated hardware. VNFs move individual network functions from dedicated hardware devices to a software that runs on commodity hardware. Meanwhile, NFV decouples the network functions from the proprietary hardware appliances to run them by using a software and to subsequently accelerate service innovation and provisioning, especially within service provider environments. The emergence of NFV benefits the software-operated networks instead of those operated by physical devices and enhances the flexibility and rapid deployment of each network service function. SDN is another technology that is used to solve the traditional network rigidity problem and to satisfy the requirements of diversified and dynamic network services [4]. The main purpose of SDN is to solve the limitations of the

FIGURE 1: Concept of NFV.

FIGURE 2: Concept of edge computing.

traditional network architecture where the network is divided into control and data planes. The software rearranges the network architecture by granting the network administration authority to the software controller on the control plane through a centralized control approach. In this way, this control mode, which is withdrawn from the existing network architecture and transformed into a programmable network, effectively addresses the limitations of the traditional network design. A centralized control and flexible operation of the network can be achieved by using a single logic point to control the entire network. SDN and NFV are networking technologies that can work with each other to lead the transformation, deployment, and management of the network design under the environment of data transmission and telecommunication infrastructure. The integration of these technologies also presents a direction for the future development of the Internet, especially after the emergence of the 5G technology [5, 6].

As the network becomes increasingly flexible, software defined, and virtualized, several standards organizations are working to introduce the SDN/NFV technology into

mobile networks to satisfy the low latency requirement of the 5G/IoT network for data processing and information transmission and to indirectly drive the development of the edge computing technology [7–9]. Edge computing is a distributed computing architecture that moves computing power for applications, data, and services from a cloud data center server to the CPE that is located close to the edge of the user network to be processed [10–12]. Gateway is a type of CPE and common-edge computing device that possesses basic capabilities in computing, analyzing, and preprocessing the data that are collected near hosts and other IoT devices to accelerate the data processing and to reduce the transmission latency [13, 14]. In the gateway-based edge network architecture, the data are computed and processed near their sources, thereby making this architecture suitable for processing immediate service requests.

The concept of edge computing is shown in Figure 2. The edge network is located between IoT devices and the cloud [15]. Edge computing extends the existing cloud computing paradigm to the network edge in order to satisfy the requirements of latency-critical and computation-intensive

IoT applications. The distributed architecture of edge computing nodes satisfies the computing requirements for several applications, data, and services from the cloud data center to the CPE or the micro data center. Based on different fields of development, the edge computing network can be divided into cloudlets, mobile edge computing (MEC), and fog computing [16–18]. Cloudlets are communication tools that provide microservices to the network surrounding the user and virtualize and shrink the computing resource to deploy the computing resource near the end of the user. The application services and technologies of cloudlets are currently being promoted by Akamai and Microsoft. Introduced by the European Telecommunications Standards Institute (ETSI), MEC is highly applicable in the field of mobile communication and is operated and managed by communication companies [19]. This technology mainly aims to reduce the increasing pressure on network equipment and to help mobile companies create a unique mobile service model. Introduced by Cisco and currently promoted by the OpenFog Consortium Alliance, fog computing is a concept of extended cloud computing that focuses on the data processing function of the local network [20]. By emphasizing near-field data communication, fog computing can be used in different network equipment, for personal or business management, and for providing relevant IoT services in specific areas. The OpenFog Consortium aims to promote the OpenFog computing technology by developing an open architecture that can identify certain core technologies and capabilities, such as distributed computing, network, and storage, which provide intelligent support at the edge of the IoT. Keeping in line with the current trends in technology integration, OpenFog has also cooperated with the ETSI Industrial Specification Group to establish an application framework and development technology for the specification and interoperability of ICT and for expanding the application scope of edge computing.

The emergence of the 5G technology has increased the requirements for data movement, data storage, data processing, and data analysis for the traditional centralized cloud computing architecture, thereby increasing the required number of hardware devices to improve the load capacity of the core network and the servers. Given the large amount of data generated from IoT, the existing cloud service model cannot satisfy the large and dispersed requests of users. This model has been used for data computing, analysis, and processing. However, the demand for low latency and immediacy has gradually increased, thereby rendering this model slightly insufficient. The introduction of edge computing helps process data in advance and reduce data traffic and transmission time; this technology also allows the transfer of computing power to the CPE or terminal nodes to enhance the immediate feedback of the edge operations to the environment [21]. The Cisco Internet Business Solutions Group [22] predicts that the IoT will have more than 50 billion terminals and devices by 2020, while the International Data Corporation estimates that more than 40% of the data in the future need to be analyzed, processed, and stored at the edge of the network [23]. To effectively respond to the challenges brought upon by the aforementioned changes, the

service model in the edge of the network must be redefined to optimize the network operation and service processing efficiency, to reduce the workload in the cloud, and to satisfy the latency requirements of 5G IoT.

The rest of this paper is organized as follows. Section 2 presents the research background and the related works. Section 3 describes the design of the edge computing service model. Section 4 presents the experiment results. Section 5 concludes the paper.

2. Background and Related Works

Network operators must deploy various types of network equipment and software platforms to support increasingly diverse Internet requirements. However, the current network design is based on hardware concept, various hardware devices are often incompatible or unable to communicate with one another, and network maintenance management has a high complexity and cost. NFV has been recently proposed as a solution to the difficulties being faced by network operators. Through NFV, network and communication operators can decrease the related costs, flexibly and efficiently process the customer network deployment requirements, reduce the waste of hardware resources, and increase the resource utilization efficiency. The current NFV related standards are mainly driven by ETSI. Given the impact and benefits brought upon by the virtualization of network functions, operators—particularly network service providers and telecommunication operators—are considered the most active participants in the development of NFV standards.

As shown in Figure 3, hypervisor and container are the two main frameworks for VNFs [24]. Hypervisor is the most common virtualization technology that uses a group of virtualization hardware devices to execute the host operating system in a virtual manner and to abstract the physical hardware of the host operating system. The virtual machine (VM) is the operating system deployed on the virtual management program, while the Linux container (LXC) is an operating-system-level virtualization technology that mainly packages the application service system into a software container that stores the codes of the application service software, required operating system core, and library. As its most attractive feature, LXC has a simplified software structure. Therefore, the image file of the container only includes a small part of those operating system components that are smaller than the general VM, which is a complete operating system. Given its small size and lightweight features, the container has a much faster deployment or movement speed compared with VM. Accordingly, most industry practitioners prefer the container over the VM. Docker is an open source project based on LXC technology [25] that is operated on the Linux operating system. This technology creates an additional software abstraction layer for virtualizing the service application in response to the development of microservices [26]. Aside from being a famous product of the Git Hub community, Docker has also been favored by Google and Red Hat and is currently the fastest growing NFV virtualization technology. Many studies have also been conducted to examine the

FIGURE 3: Common approach for the hosting of VMs versus containers.

implementation of Docker as an edge computing platform [27, 28].

The development of virtualization technologies has driven several innovations in the IoT architecture. Edge computing has also emerged as a new concept that satisfies the requirements of IoT for distributed data processing and number of devices. CPE devices, such as the edge gateway, use virtualization technology to provide local operations that are particularly suitable for IoT applications with low latency and immediacy. Edge computing can completely disassemble large-scale services that are originally handled by cloud data centers, cut these services into small and easy-to-manage parts, and distribute them among the edge network nodes or micro data centers to accelerate their resolution and to reduce the workload in the cloud. In this case, the service requests of users are processed on the gateway of the edge network, while those requests that exceed the computing capabilities or cannot be handled by the edge network are forwarded to the cloud for processing. Edge computing can also be used to reduce the amount of data being sent back to the cloud. These data are initially processed at the edge node and are then preprocessed or filtered before they are transmitted back to the cloud to ensure that only useful data will be transmitted. Edge computing not only reduces network connection delays and meets the demand of 5G/IoT for improving delays but also promotes the convergence of ICT industry technologies. Using lightweight virtual technology also helps network operators install applications and control software on edge gateways, provide localization services, and create innovative service models. Reference [29] proposes a SDN-based edge computing architecture called software-defined infrastructure (SDI), which uses OpenFlow and OpenStack to virtualize service resources for building smart applications on virtual infrastructure that can flexibly schedule network resources. To introduce the concept of resource sharing, [30] proposes a cloud-edge framework where each edge node can use the local processing platform to share computing resources, to reduce the computational burden on the cloud that processes

the data, and to improve the operation of the edge network. To meet the low latency and fast data processing requirements of mobile networks, [12] examines mobile edge operations, identifies the applicable situations and reference scenarios for different MECs, and proposes marginalities for MEC offloading decisions, computing resource allocation, and mobility management. The computational needs of the Internet can also be used as reference for examining edge network service management and application innovation.

Given the resource constraints (e.g., CPU, memory, and network-intensive service requests) of the local gateway hardware, the task scheduling mechanism for handling the service requests of users plays an important role in improving the service capabilities of edge computing. Several algorithms for task scheduling are available in a computer system [31]. In-depth studies on several task scheduling algorithms for cloud computing have also been conducted [32, 33]. Reference [34] proposes a task scheduling approach called HealthEdge, which sets various processing priorities for different tasks based on the collected human health data and determines whether a task must be operated on a local device or a remote cloud in order to reduce the total processing time as much as possible. Reference [35] adopts a Markov decision process approach to solve the stochastic optimization problem in the MEC system. In this approach, the computation tasks are scheduled based on the queuing state of the task buffer, the execution state of the local processing unit, and the state of the transmission unit. Reference [36] proposes a greedy best availability (GBA) mechanism to identify the idealized task scheduling policy and to reduce the queuing time of services by scheduling the tasks according to their work completion time. Reference [37] analyzes the performance of the round-robin (RR) algorithm in the cloud computing environment and reveals that RR scheduling fairly allocates computing resources among tasks of the same priority by using the time slicing approach. The results show that the RR algorithm demonstrates a better response time and load balancing compared with the other algorithms.

FIGURE 4: Network function scheduling problem in CPE.

3. Problem Definition

NFV is currently the most valued solution to the problems associated with the operation cost and efficiency of network services. Accordingly, many ICT companies have begun to examine virtualization technology. Despite the high expectations toward this technology, satisfying service requests, effectively allocating virtual computing resources, and providing service on-demand application still challenge the deployment of NFV and play key roles in the future development of 5G/IoT services. NFV also allows the establishment of multiple independent, heterogeneous virtual networks based on common underlying network resources, thereby enabling service providers to provide customized services according to the demands of users. Virtual network embedding is a process of mapping the virtual network to the underlying network (substrate network) through the mapping algorithm and according to the current resource situation of the infrastructure provider. This process is an NP-Hard problem and has been investigated in NFV resource allocation research [38, 39]. Previous studies [40, 41] have also systematically discussed the problem of virtual network mapping and provided good references for research.

This paper aims to optimize task scheduling and resource allocation by using the proposed edge computing service model. Scheduling tasks in edge computing are more complex than that in cloud computing. An edge computing operation is typically spread over the device of the client, the edge gateways, and occasionally a broker of the cloud network. Therefore, deciding where to schedule computational tasks remains a key problem in edge computing. Given that the gateway-based IoT architecture is currently the mainstream, the task scheduling mechanism discussed in this paper focuses on the edge gateway. Task scheduling and resource allocation are the main problems to be solved in this gateway given the limited amount of available computing resources. Lightweight NFV technology plays an indispensable role in the rapid deployment of the edge gateway. The relationship between task scheduling and resource allocation for host service requests is plotted in Figure 4. As each service request arrives at a different time, the required VNF service and processing time also differ. The edge gateway can adjust the task scheduling and resource configuration according to the

differences among the service requests. As the basic idea for the service on-demand model, only one VNF is dedicated to a single service request. This service model problem can be analyzed by using queuing theory, which regards the request and react processing as a waiting-line system, the input of a waiting-line system as the service request, the service counter as the gateway scheduling function, and the output as the requested VNF resource. Although many queuing models may be used in operations management [42], this paper primarily focuses on the task scheduling approach that allows a certain edge gateway to process multiple VNFs from a queue one after another. Given its limited capacity, the edge gateway can only schedule a limited number of service requests, while the subsequent service requests need to be forwarded to the cloud for processing. With these considerations, the research problem is formulated as follows. Assume that the service request is Poisson's ratio within one edge gateway and that the service time is an exponential distribution. The service request set can be denoted as $R = \{R_1, R_2, R_3, \dots, R_n\}$, where n denotes the number of service requests in the system. Equations (1) and (2) are used to determine the probability of n service requests in the system. In these equations, N denotes the maximum number of service requests that can be scheduled in the gateway, λ is the average number of incoming user requests in one unit of time, μ is the service efficiency (the ability of the service counter), $\rho = \lambda/\mu$ is the ratio that the request can be met within one unit of time, P_0 denotes the initial condition, L denotes the total number of service requests arriving in the system within a planned time, and L_q denotes the total number of server requests queued in the system. L and L_q can be computed by using (3) and (4), respectively. Equation (5) computes the waiting and service time of a service in the system, while (6) computes W_q or the average waiting time for a user request.

$$P_0 = \begin{cases} \frac{1-\rho}{1-\rho^{N+1}}, & \rho \neq 1 \\ \frac{1}{N+1}, & \rho = 1 \end{cases} \quad (1)$$

$$P_n = \begin{cases} \rho^n P_0 = \rho^n \frac{1-\rho}{1-\rho^{N+1}}, & \rho \neq 1 \\ P_0 = \frac{1}{N+1}, & \rho = 1 \end{cases} \quad (2)$$

FIGURE 5: Service on-demand edge computing model.

$$L = \begin{cases} \frac{\rho}{1-\rho} - \frac{(N+1)\rho^{N+1}}{1-\rho^{N+1}}, & \rho \neq 1 \\ \frac{N}{2}, & \rho = 1 \end{cases} \quad (3)$$

$$L_q = L - (1 - P_0) \quad (4)$$

$$W = \frac{L}{\lambda(1 - P_n)} \quad (5)$$

$$W_q = \frac{L_q}{\lambda} \quad (6)$$

The results reveal that the waiting time for the user request can be reduced in two ways, namely, by reducing the number of service requests queued in the edge gateway and by increasing the processing speed of task scheduling in the edge computing gateway. A large number of tasks must be completed within a short period to achieve an efficient edge computing.

4. Design of the Edge Computing Service Model

How to construct an elastic and cost-effective edge computing service model, improve management efficiency, meet user service requests, achieve centralized management, and develop flexible configuration service models are the current research trends in the development of a 5G SDN/NFV network. This paper proposes a gateway-based edge computing service model (Figure 5) to improve the operational efficiency of the edge computing node, to accelerate the processing of user service requests, and to increase the utilization efficiency of a limited number of computing resources. In this model, when different user requests enter the edge gateway, this gateway determines whether the requested services can be processed or not. If the edge gateway itself lacks the computing capacity or resources, then the controller forwards the service request to the cloud to reduce the data processing latency.

The proposed edge computing service model can be divided into resource estimation, scheduler, and lightweight VNF configuration. Figure 6 presents the flowchart of the operations in the proposed edge computing service model, and these three parts are further described in the following sections.

Resource estimation checks whether the edge gateway has a sufficient amount of computing resources to provide edge computing services. For the user request set R , the resource allocation must abide by the following rules, which are also used by the edge gateway:

- (1) For a single service request R_i , any resource of V_i (e.g., CPU, memory, and disk) is less than the total resources of P . $\forall R_{Ci} < P_C$, $\forall R_{Mi} < P_M$, and $\forall R_{Di} < P_D$, R_{Ci} , R_{Mi} , and R_{Di} denote the CPU, memory, and disk space requests sent to R_i , respectively.
- (2) The sum of computing resources that are allocated to the VNF in the gateway is less than the total number of resources of the physical machine P . $V_{Ci} < P_C$, $V_{Mi} < P_M$, $V_{Di} < P_D$, V_{Ci} , V_{Mi} , and V_{Di} denote the sum of CPU, memory, and disk space resources allocated to V_i , respectively.

As shown in Figure 6, after receiving the user service request, the edge gateway checks whether a sufficient amount of computing resources is available to satisfy such request. The following situations may be encountered in this case:

- (1) If the edge gateway has a sufficient amount of resources, then the user service request is processed through the scheduler and queued in the system based on the result of the task scheduling algorithm.
- (2) If the edge gateway does not have a sufficient amount of available resources, then the user service request is directly transferred to the cloud instead.

Task scheduling aims to increase the operational efficiency of the edge gateway. Given that one service request differs from another, task scheduling examines how the

FIGURE 6: Edge computing service model flowchart.

edge gateway can meet the requirements of different service requests and accomplish task scheduling in the system. As its primary purpose, scheduling attempts to reduce the amount of time spent on dealing with the most demanding service requirements as much as possible. For this purpose, this paper constructs the Greedy Available Fit (GAF) task scheduling mechanism to enhance the operational efficiency of edge computing services.

Assume that each service request R_i configures a VNF virtualization service resource V_i . t_i denotes the processing time of the i th service request, d_i denotes its deadline, and j denotes its completion time in the system. The R_i in the system must be completed at schedule and before the basic time limit. Otherwise, this service request must be forwarded directly to the cloud. As each service request arrives at a different time, the time required for the operation processing and the deadline time also differ. In this case, deadline is selected as a priority parameter for the task scheduling. The deadline can be used to define the processing priority for services; that is, the deadline for those services that require real-time processing may be set according to their processing and precedence requirements on different VNFs. This paper aims to insert as many tasks as possible into the schedule before completing the largest task R_i of the deadline. To accomplish this objective, a task with the maximum deadline is selected as the baseline and the remaining tasks are sequentially inserted into the queue based on their processing

time. $N[i, j]$ denotes the maximum number of j tasks selected from the front i tasks and can be formulated as

$$N[i, j] = \max \begin{cases} N[i-1, j] \\ N[i-1, j-t_i] + 1 & \text{if } j \leq d_i \\ N[i-1, j-t_i] & \text{if } j > d_i \end{cases} \quad (7)$$

Equation (8) shows the initial condition of $N[i, j]$.

$$N[1, j] = \begin{cases} -\infty & \text{if } j \neq t_1 \\ 1 & \text{if } j = t_1 \leq d_1 \\ 0 & \text{if } j = t_1 > d_1 \end{cases} \quad (8)$$

Assume that there is no time gap between R_i and R_{i+1} , that R_1 starts from time 0, and that $N = 1$. If $t_1 \leq d_1$, then $N = 0$ because the deadline is exceeded.

The whole decision process is summarized in Algorithm 1, which starts with an empty schedule and inserts the available tasks into this schedule in three different cases. In case 1, if the current time step j exceeds the deadline of R_i , then $N[i, j] = N[i-1, j-t_i]$. In case (2), if there is not enough time to finish request R_i by the current time j , then $N[i, j] = N[i-1, j]$. In case (3), if time j does not exceed the deadline of R_i and there is enough time to finish request R_i by time j , then $N[i, j] = \max(N[i-1, j], N[i-1, j-t_i] + 1)$.

```

Input:  $R = \{R_1, R_2, \dots, R_n\}$ ,  $t_i$ ,  $d_i$ ,  $j$ 
Output:  $N[i, j]$ 
(1) Start
(2) Set the largest task  $d_i$  as the baseline of the scheduler
(3) for  $i \leftarrow 1$  to  $n$  //initialize the maximum  $N$  at time 0 to be zero for each service request in queue
(4)  $N[i, 0] = 0$ 
(5) for  $j \leftarrow 1$  to  $d_n$  //determine the maximum  $N$  obtained by  $R_1$  at each time step
(6) if  $((j == t_i) \text{ and } (j \leq d_i))$ 
(7)  $N[1, j] = 1$ 
(8) else
(9)  $N[1, j] = N[1, j - 1]$ 
(10) for  $i \leftarrow 2$  to  $n$ 
(11) for  $j \leftarrow 1$  to  $d_i$ 
(12) if  $(j > d_i)$  // case (1): the current time step  $j$  already exceeds  $R_i$ 's deadline
(13) then  $N[i, j] = N[i - 1, j - t_i]$ 
(14) else if  $(j < t_i)$  // case (2): there is not enough time to finish request  $R_i$  by the current time  $j$ 
(15) then  $N[i, j] = N[i - 1, j]$ 
(16) else // case (3): time  $j$  does not exceed  $R_i$ 's deadline, and there is enough time to finish request  $R_i$  by time  $j$ 
(17) then  $N[i, j] = \max(N[i - 1, j], N[i - 1, j - t_i] + 1)$ 
(18) Schedule Completed
(19) End

```

ALGORITHM 1: Greedy Available Fit algorithm.

Virtualization technology can be applied on the CPE in many ways, such as by using the VM, container, or VM integration of container. However, a traditional VM consumes many system resources and cannot meet the requirements for light weight and service on-demand deployment. In this case, this research adopts container technology instead. A container handles only one service request at a time and stops its operation after completing the delivery of a service. The VNF manager on the gateway is responsible for configuring and allocating each VNF. The VNF template image that is required by different services is placed in the VNF resource pool (with Docker Hub as the default; other public or private registrations can also be specified). The edge gateway not only controls the amount of gateway resources used by each container but also manages several resources, such as CPU and RAM, to ensure that the container can obtain the required resources without affecting the performance of the other executing containers on the edge gateway.

Linux containers adopt a hierarchical structure to rapidly deploy VNFs and to manage NFV flexible scheduling. The underlying structure uses the file archiving mechanism of Docker, namely, the advanced multilayered unification file system, to incorporate many different VNF images. The layers are stacked up, and when some VNF service functions need to be accessed, the container retrieves the VNF images through the VNF manager and uses these images directly. To integrate the open resources of Docker Hub and to expand the private warehouse of the VNF service manager, the system can store the image files obtained from the public warehouse in a private warehouse for future use. The judgment mechanism of the VNF service manager is described as follows:

- (1) If the image requested by the host is already in the local container, then the VNF image can be taken out directly by the Docker daemon of the VNF service manager. No complex configurations are required.
- (2) If the VNF image to be used by the host is not in the local container, then the VNF manager connects to the common warehouse Docker Hub on the Internet to find and store the desired service image file in a private warehouse via push/pull. Afterward, the acquired VNF image is controlled by the Docker daemon.

By using the OS kernel for resource isolation, the container technology does not need to rely on a virtual software layer and does not require the VM to install a guest OS. Therefore, the capacity of the image file is much larger than that of the virtual machine. The image file is small and can be rapidly deployed through network transmission, thereby saving network resources. By using Linux container technology, we can configure various VNF images to be executed on the same edge gateway, thereby replacing the virtual machine and achieving the goal of lightweight virtualization.

5. Experiment

An experiment is conducted to evaluate the proposed task scheduling algorithm and to test its performance in deploying lightweight VNF. The experiment environment is shown in Figure 7. A simulation is initially performed to evaluate the task scheduling performance of the algorithm by changing the number of service requirements. The service request is

FIGURE 7: The experiment environment.

characterized as a Poisson process. There we showed that the average number of service requests in the system is given by λ/μ where λ is the average arrival rate and μ is the average service rate. The input parameter λ/μ is 0.45. First come first service (FCFS), priority task scheduling [34], RR [37], and GBA [36] are then compared with the proposed GAF mechanism.

The effect of the service requests on the average waiting time, average response time, and task scheduling of different scheduling mechanisms is evaluated in the simulation. FCFS is based on the service requests that enter the edge gateway queuing system and schedules the tasks according to the sequence of these requests. Priority task scheduling is an unfair scheduling algorithm where the service requests are sorted according to their priority and where the high-priority tasks are performed first. Those tasks having the same priority are sorted by using the FIFO scheduling mechanism. The GBA sorts the tasks based on the completion time of the service requests in queuing system. The RR algorithm is based on the conventional RR scheduling performed in the process scheduling. RR scheduling fairly allocates the computing resources to those tasks with the same priority by using the time slicing approach (time quantum).

Figure 8 shows the experiment result for average waiting time, where the x-axis refers to the number of service requests and the y main axis refers to the average waiting time; that is, the time a user spends to complete the service scheduling and to determine the resource allocation after receiving a service request and a successful reply. If the demand reverts to an error/timeout or if the edge gateway does not have enough computing resources to support such demand, then the sample is excluded from the calculation of the average waiting time. As can be seen in the experimental result, the RR algorithm obtains the longest average waiting time because of its application of time slice rules in order for each task to be processed within a fixed amount of time. The time quantum affects the overall performance of the operation.

FIGURE 8: The experiment result for average waiting time.

Having a very long time quantum leads to a very long waiting time, while having a very short time quantum results in task schedule conversion, a poor execution efficiency, and an extended waiting time. For FCFS, given that the time spent differs across each task, the waiting time for the next task must be determined based on the schedule of the previous task. Therefore, if the last task schedule is too long, the system cannot quickly process the subsequent task schedules, thereby affecting the overall task scheduling efficiency. Given that its average waiting time is relatively shorter than that of the FCFS mechanism, priority task scheduling can satisfy the task requirements and flexibly perform the scheduling. However, prioritizing a high-priority program will delay all of the low-priority requests, thereby creating an indefinite situation in the low-priority program and extending the waiting time. GBA is based on the earliest completion time of the task. When the service demand is low, the average waiting time of the GBA algorithm is similar to that of priority task scheduling. However, when the service demand increases, the average waiting time of the GBA mechanism becomes longer than that of priority task scheduling. Compared with FCFS,

FIGURE 9: The experiment result for average response time.

FIGURE 10: The experiment result for task scheduling efficiency.

priority task scheduling, RR, and GBA, the proposed GAF algorithm selects the largest deadline task as the baseline and then sequentially inserts the remaining tasks into the queue based on their processing time. Therefore, the GAF algorithm obtains the shortest average waiting time.

Figure 9 shows the results for average response time. As expected, the RR algorithm outperforms all other approaches in terms of average response time because this algorithm is especially designed for time sharing. Using a fair strategy to allocate VNF resources ensures that each service request has a fixed time. If the service processing time exceeds the time quantum allocated by the system, then the sample is excluded from the calculation of the average response time. Given that FCFS is processed in sequence, the average response time will be affected by the time spent by the previous R_{i-1} task in the scheduler. For example, if the previous service request has a very long task schedule, then the average response time increases. Given that the proposed GAF algorithm sequentially inserts tasks into the queue, its average response time is slightly longer than that of FCFS and GBA. Although the priority algorithm can sort the tasks according to the priorities of the service requests, the low-priority tasks are easily delayed, thereby increasing the average response time.

Figure 10 shows the task scheduling efficiency of the compared algorithms. If the service request number is less than 60, then each algorithm can complete the task scheduling. However, when this number is exceeded, the number

of tasks that can be scheduled varies across each algorithm. GAF outperforms all the other algorithms as it ranks the largest deadline task at the end and then sequentially sorts the remaining tasks according to their processing time. In this way, this algorithm ensures that most tasks will be completed within a minimum processing time. Those service requests that cannot be scheduled will bypass the gateway and will be directly forwarded to the cloud data center. When the service request number is 80, GAF can complete 68% of the task scheduling. Meanwhile, GBA performs the scheduling based on the earliest task completion time. Although GAF and GBA can schedule the same number of tasks in the experimental environment, the latter has a longer average waiting time compared with the former. GAF also outperforms FCFS and priority task scheduling by 9.6% and 6.2% in terms of task scheduling efficiency, respectively, because this approach ranks the largest deadline task at the end and then sorts the remaining tasks according to their required processing time. GAF also outperforms the RR mechanism by 70% because the latter applies the time-sharing rule to minimize the number of scheduled tasks. Based on these results, the GAF algorithm can schedule more tasks in the edge gateway within a short average waiting time and improve the operational efficiency of edge computing services.

In addition to task scheduling, the VNF configuration is another main factor that affects the operational efficiency of the edge computing service model. This paper utilizes the Docker technology to achieve an agile deployment of VNFs. Such technology can provide a centralized management and flexible configuration of VNF services. Depending on the service requests of each user, the deployment of different VNFs can meet the application requirements of services on-demand. The VNF image of Docker can be mainly divided into four categories, namely, system images (e.g., Ubuntu, CentOS, and Debian), tool images (e.g., Golang, Flask, and Tomcat), service images (e.g., MySQL, MongoDB, and RabbitMQ), and application images (e.g., WordPress and DRUPAL). To test the practical deployment efficiency of lightweight VNF for the edge gateway, a pull performance test is performed for each type of Docker VNF image. The edge gateway is simulated by using a standalone PC with 3.4 GHz dual-core CPU, 16GB memory, and 5400 rpm HD 500G. The experimental network bandwidth is set to 100 Mbps. To compare the pull performance with the local edge gateway, a test VM in the Google Cloud platform is used to simulate the cloud network broker. Table 1 presents the result of the pull performance test.

As shown in Table 1, each VNF image size has different physical gateway and cloud network broker costs due to the limitations in the network bandwidth. A larger VNF image takes a longer time to be downloaded from Docker Hub. As expected, the cloud network broker has a better computing architecture than the local edge gateway because the location of the edge gateway affects the performance of the VNF deployment. Although the edge computing architecture of the cloud network broker deploys VNF more efficiently than the local gateway, the user service requests must still be sent to the Internet for processing, and this method makes it difficult to demonstrate the technical advantages of edge computing.

TABLE 1: Pull performance test results for different VNFs.

Image Types	VNFs	Image Size (MB)	Physical Gateway Cost (seconds)	Cloud Network Broker Cost (seconds)
System images	ubuntu	112	22.095	7.918
	centos	207	24.319	8.926
	debian	207	21.635	5.646
Tool images	golang	779	86.793	18.51
	flask	458	62.22	16.33
	prometheus	112	24.14	4.737
	httpd	177	20.753	1.031
Service images	nginx	109	13.216	4.778
	bind	337	84.38	15.105
	squid	215	150.031	9.973
	mysql	374	24.852	10.449
	mongodb	503	61.347	17.116
Application images	wordpress	407	32.866	13.213
	drupal	452	81.39	14.405

FIGURE 11: VNF boot time test for the VM and container.

FIGURE 12: Average VNF boot time test for the VM and container.

In the VNF boot time test, we set the image to Ubuntu-16-04-x64. VM and Docker are used to perform the boot time test on the edge gateway. We test the time of opening five VMs and Docker containers. VMware is used as the VM hypervisor. As shown in Figure 11, the five VMs are opened in 159.3 seconds, while the Docker containers are opened in 11.095 seconds. In sum, the VMs have a much longer deployment time than Docker. More time can be saved by using Docker in the edge gateway that requires a large amount of VNFs.

To test the average VNF boot time, we boot a VM, wait for its activation, shut it down, and repeat the same steps for 14 other VMs. The test results are shown in Figure 12. When using VMware, the average time to boot the VMs is about 31.86 seconds. However, when using Docker, these VMs are booted in only 2.2 seconds. In sum, the Docker container performs approximately 15 times faster than the VM.

The results of the VNF deployment test highlight the significant advantages of lightweight containers over traditional virtualization methods. These containers can be started

within a few seconds because they adopt a common host OS, thereby making it unnecessary to execute the guest OS in each container. The container also does not need to wait for the OS to start up, thereby saving a few seconds. Fast start is far faster than traditional VMs. Meanwhile, given the design of lightweight VNF images and its highly efficient virtualization, the application-centric virtualization technology Docker can automatically obtain container images from the Docker Hub for flexible deployment and management and can meet the application requirements of the edge network for the service on-demand of VNF deployment.

6. Conclusions

This paper proposed a gateway-based edge computing service model and a GAF task scheduling mechanism that allows the edge gateway to schedule more tasks within a short average waiting time. The resource estimation and lightweight VNF configuration technologies are used to improve the operational efficiency of edge computing services, to increase

resource utilization, to achieve a rapid deployment of VNF, and to meet service on-demand requests. The combination of resource estimation, task scheduling, and lightweight VNF configuration design provides an integrated solution that can satisfy the service on-demand requests for 5G networks. The simulation results showed that the proposed GAF mechanism outperforms the other scheduling algorithms. Meanwhile, the comparison revealed that using the lightweight virtualization technology in edge gateways is more efficient and competitive than using traditional VMs.

In the future, we plan to study the multiqueue task scheduling problem for edge computing, the possible cooperation between edge gateway and cloud servers, and the use of SD-WAN technology to achieve a seamless operation of the cloud-edge network.

Data Availability

The data used to support the findings of this study are included within the article.

Disclosure

The funder had no role in the study design, data collection and analysis, decision to publish, or preparation of this manuscript.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

Acknowledgments

The work described in this paper was supported in part by the Ministry of Science and Technology of the Republic of China (Grants nos. 104-2221-E-008-039, 105-2221-E-008-071, 107-2623-E-008-002, and 107-2636-E-003-001).

References

- [1] J. Matias, J. Garay, N. Toledo, J. Unzilla, and E. Jacob, "Toward an SDN-enabled NFV architecture," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 187–193, 2015.
- [2] Q. Duan, N. Ansari, and M. Toy, "Software-defined network virtualization: An architectural framework for integrating SDN and NFV for service provisioning in future networks," *IEEE Network*, vol. 30, no. 5, pp. 10–16, 2016.
- [3] TSI, "Network Functions Virtualization (NFV), architectural framework," [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf.
- [4] ONF, "SDN Architecture," [Online]. Available: http://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-SDN_ARCH.1.0_06062014.pdf.
- [5] A. Basta, A. Blenk, K. Hoffmann, H. J. Morper, M. Hoffmann, and W. Kellerer, "Towards a Cost Optimal Design for a 5G Mobile Core Network Based on SDN and NFV," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1061–1075, 2017.
- [6] F. Z. Yousaf, M. Bredel, S. Schaller, and F. Schneider, "NFV and SDN-Key technology enablers for 5G networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2468–2478, 2017.
- [7] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How Can Edge Computing Benefit From Software-Defined Networking: A Survey, Use Cases, and Future Directions," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2359–2391, 2017.
- [8] Guangshun Li, Jiping Wang, Junhua Wu, and Jianrong Song, "Data Processing Delay Optimization in Mobile Edge Computing," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 6897523, 9 pages, 2018.
- [9] Zhimin Wang, Qinglin Zhao, Fangxin Xu, Hongning Dai, and Yujun Zhang, "Detection Performance of Packet Arrival under Downclocking for Mobile Edge Computing," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 9641712, 7 pages, 2018.
- [10] W. Yu, F. Liang, X. He et al., "A Survey on the Edge Computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.
- [11] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.
- [12] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile Edge Computing: A Survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.
- [13] Guangshun Li, Jianrong Song, Junhua Wu, and Jiping Wang, "Method of Resource Estimation Based on QoS in Edge Computing," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 7308913, 9 pages, 2018.
- [14] D. Happ and A. Wolisz, "Towards gateway to Cloud offloading in IoT publish/subscribe systems," in *Proceedings of the 2nd International Conference on Fog and Mobile Edge Computing, FMEC 2017*, pp. 101–106, Valencia, Spain, May 2017.
- [15] G. Premasankar, M. Di Francesco, and T. Taleb, "Edge Computing for the Internet of Things: A Case Study," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275–1284, 2018.
- [16] L. Zhao, W. Sun, Y. Shi, and J. Liu, "Optimal Placement of Cloudlets for Access Delay Minimization in SDN-Based Internet of Things Networks," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1334–1344, 2018.
- [17] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.
- [18] Y. Liu, J. E. Fieldsend, and G. Min, "A Framework of Fog Computing: Architecture, Challenges, and Optimization," *IEEE Access*, vol. 5, pp. 25445–25454, 2017.
- [19] ETSI Multi-access Edge Computing, [Online]. Available: <http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing>.
- [20] OpenFog Consortium, [Online]. Available: <https://www.openfogconsortium.org/>.
- [21] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.
- [22] Cisco, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything white paper," [Online]. Available: https://www.cisco.com/c/dam/en_us/.../IoT_IBSG_0411FINAL.pdf.
- [23] IDC, "FutureScape: Worldwide Internet of Things 2017 Predictions," November 2016, [Online]. Available: <https://www.idc.com/getdoc.jsp?containerId=US41910716>.

- [24] T. Salah, M. J. Zemerly, C. Y. Yeun, M. Al-Qutayri, and Y. Al-Hammadi, "Performance comparison between container-based and VM-based services," in *Proceedings of the 20th Conference on Innovations in Clouds, Internet and Networks, ICIN 2017*, pp. 185–190, Paris, France, March 2017.
- [25] Docker Hub,[Online]. Available: <https://hub.docker.com/>.
- [26] C.-W. Tseng, M.-S. Tsai, Y.-T. Yang, and L.-D. Chou, "A Rapid Auto-Scaling Mechanism in Cloud Environment," in *Proceedings of the The 13th International Conference on Grid, Cloud, and Cluster Computing*, Las Vegas, Nev, USA, 2017.
- [27] B. I. Ismail, E. Mostajeran Goortani, M. B. Ab Karim et al., "Evaluation of Docker as Edge computing platform," in *Proceedings of the 2015 IEEE Conference on Open Systems (ICOS)*, pp. 130–135, Melaka, Malaysia, August 2015.
- [28] R. Morabito and N. Beijar, "Enabling Data Processing at the Network Edge through Lightweight Virtualization Technologies," in *Proceedings of the 2016 IEEE International Conference on Sensing, Communication and Networking, SECON Workshops 2016*, UK, June 2016.
- [29] T. Lin, B. Park, H. Bannazadeh, and A. Leon-Garcia, "Demo Abstract: End-to-end orchestration across sdi smart edges," in *Proceedings of the 1st IEEE/ACM Symposium on Edge Computing, SEC 2016*, pp. 127–128, USA, October 2016.
- [30] A. Amjad, F. Rabby, S. Sadia, M. Patwary, and E. Benkhe-lifa, "Cognitive Edge Computing based resource allocation framework for Internet of Things," in *Proceedings of the 2nd International Conference on Fog and Mobile Edge Computing, FMEC 2017*, pp. 194–200, Spain, May 2017.
- [31] C. Fan, H. Deng, F. Wang, S. Wei, W. Dai, and B. Liang, "A survey on task scheduling method in heterogeneous computing system," in *Proceedings of the 8th International Conference on Intelligent Networks and Intelligent Systems, ICINIS 2015*, pp. 90–93, Tianjin, China, November 2015.
- [32] P. Akilandeswari and H. Srimathi, "Survey and analysis on task scheduling in cloud environment," *Indian Journal of Science and Technology*, vol. 9, no. 37, 2016.
- [33] E. Meriam and N. Tabbane, "A survey on cloud computing scheduling algorithms," in *Proceedings of the 2016 Global Summit on Computer and Information Technology, GSCIT 2016*, pp. 42–47, Sousse, Tunisia, July 2016.
- [34] H. Wang, J. Gong, Y. Zhuang, H. Shen, and J. Lach, "HealthEdge: Task scheduling for edge computing with health emergency and human behavior consideration in smart homes," in *Proceedings of the 2017 IEEE International Conference on Big Data (Big Data)*, pp. 1213–1222, Boston, MA, USA, December 2017.
- [35] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 1451–1455, Barcelona, Spain, July 2016.
- [36] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *Proceedings of the 1st IEEE Conference on Network Softwarization (NetSoft '15)*, pp. 1–9, University College London, London, UK, April 2015.
- [37] P. Samal and P. Mishra, "Analysis of variants in Round Robin Algorithms for load balancing in Cloud Computing," *International Journal of Computer Science and Information Technologies*, vol. 4, no. 3, pp. 416–419, 2013.
- [38] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206–219, 2012.
- [39] M. G. Rabbani, R. P. Esteves, M. Podlesny, G. Simon, L. Z. Granville, and R. Boutaba, "On tackling virtual data center embedding problem," in *Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management, IM 2013*, pp. 177–184, Belgium, May 2013.
- [40] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: a survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.
- [41] J. Gil Herrera and J. F. Botero, "Resource Allocation in NFV: A Comprehensive Survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.
- [42] D. Gross, J. F. Shortle, and J. M. Thompson, *Fundamentals of Queueing Theory*, John Wiley & Sons, New York, NY, USA, 4th edition, 2008.