DOI: 10.24996/ijs.2020.61.6.32

ISSN: 0067-2904

SDN-assisted Service Placement for the IoT-based Systems in Multiple Edge Servers Environment

Iman Mudhafar Ali[*], Mustafa Ismael Salman
Department of Computer Engineering, College of Engineering, University of Baghdad, Baghdad, Iraq

Received: 3/12/2019 Accepted: 15/3/2020

Abstract
Edge computing is proved to be an effective solution for the Internet of Things (IoT)-based systems. Bringing the resources closer to the end devices has improved the performance of the networks and reduced the load on the cloud. On the other hand, edge computing has some constraints related to the amount of the resources available on the edge servers, which is considered to be limited as compared with the cloud. In this paper, we propose Software-Defined Networking (SDN)-based resources allocation and service placement system in the multi-edge networks that serve multiple IoT applications. In this system, the resources of the edge servers are monitored using the proposed Edge Server Application (ESA) to determine the state of the edge server and, therefore, the acceptable services by each server. Benefiting from the information gathered by ESA, the service offloading decision would be taken by the proposed SDN Non-core Application (SNA) in a way that ensures an efficient load distribution and better resources utilization for the edge servers. A Weighted Aggregated Sum Product Assessment Method (WASPAS) was used to determine the best edge server. The proposed system was compared with a non-SDN system and showed improvement in the performance and the utilization of resources of the edge servers. Furthermore, the request handling time was considerably reduced and settled in constant rates for a different number of devices.

Keywords: Edge computing, Software Defined Networking, Internet of Things, Resource Allocation, Containerization.

توظيف الخدمات للأنظمة المعتمدة على إنترنت الأشياء في بيئة متعددة خوادم حافة بالإستعانة بالشبكات المعرفة بالبرمجيات

إيمان مظفر علي[*] , مصطفى إسماعيل سلمان
قسم هندسة الحاسبات, كلية الهندسة, جامعة بغداد, بغداد, العراق

الخلاصة
لقد أُثبت بالتجارب أن حوسبة الحافة هي حل فعال يمكن استخدامه مع الأنظمة المعتمدة على إنترنت الأشياء (IoT) لغرض تحسين أداء الشبكات وتقليل الحمل على السحابة، وذلك عن طريق تقريب الموارد من الأجهزة الطرفية، ولكن هناك بعض القيود التي تواجهها حوسبة الحافة تتعلق بكمِ الموارد المتاحة في خوادم الحافة، والتي تعتبر محدودة بالمقارنة مع السحابة. اقترحنا في هذا البحث نظاماً لتخصيص الموارد وتوظيف الخدمات في بيئة متعددة خوادم حافة بالإستعانة بالشبكات المعرفة بالبرمجيات (SDN)، والتي تخدم تطبيقات

*Email: mustafa.i.s@coeng.uobaghdad.edu.iq

متعددة معتمدةً على إنترنت الأشياء. من خلال هذا النظام يمكن مراقبة موارد خوادم الحافة بإستخدام تطبيق خادم الحافة (ESA) المقترح لتحديد حالاتها، وبالتالي تحديد الخدمات الملائمة والمقبولة لكل خادم حافة. بالإستفادة من المعلومات التي يتم جمعها من ESA، يقوم تطبيق SDN غير الأساسي (SNA) بإتخاذ القرار المناسب لترحيل الخدمات الى الوجهة الملائمة بطريقة تضمن توزيع الحمل بشكل فعال بين خوادم الحافة وإستغلال أفضل للموارد. في هذا التطبيق يتم استخدام طريقة التقدير التجميعي المرجح لحاصل الجمع والضرب (WASPAS) لتحديد أفضل خادم حافة. تمت مقارنة نتائج النظام المقترح مع النظام بدون وجود الـSDN وأظهرت النتائج تحسناً في أداء واستغلال موارد خوادم الحافة. علاوةً على ذلك، فإن النظام المقترح ساعد على تقليل وقت معالجة طلبات الخدمات واستقراره بمعدلات ثابتة لأعداد مختلفة من الأجهزة الطرفية.

1. Introduction

The recent years have witnessed vast growth in the information and communication technologies. The Internet of Things (IoT) technology is considered as the base stone for the development of smart cities, smart grid, smart factory, smart health care, etc. By using IoT technology, different devices can connect and share information [1, 2]. Every day, a huge number of IoT devices generate a massive amount of data. These data need to be stored, processed, and analyzed. As the resources' capabilities of these devices are very limited, cloud computing provided the needed resources for these data to be processed and stored [3]. However, cloud computing has some limitations related to the transmission latency and allocation of resources [4]. The location of the cloud Data Centers (DCs) and the conditions of the network connection might affect the transmission latency, power consumption, and bandwidth utilization, and therefore affect the Quality of Service (QoS) and user experience [5]. As a solution for these problems, Edge Computing has been introduced. Edge Computing technology has supported cloud computing by bringing the resources closer to the network edge [6]. Instead of sending all the IoT devices' data to far located cloud DCs, some of the data can be processed in local distributed edge servers, fog, or cloudlets. This can reduce the load on the cloud DCs and ensures better latency, link utilization, and more efficient energy consumption [7].

For further improvement in the performance of such networks, the Software Defined Network (SDN) is used. SDN is a technology that decouples the control plane from the data plane and centralizes the network intelligence in a single component which is called the SDN controller. The SDN controller is responsible for specifying the flows in the network. Hence, it can improve traffic distribution in the edge network and reduce the load on the cloud DC. Furthermore, the SDN controller can take the responsibility of service discovery and service placement in the edge by discovering the suitable edge servers. This can carry some of the burdens on the edge servers and the end devices and, therefore, improves the performance of the network and increases the QoS [8].

As expected, edge servers have limited hardware capabilities as compared with cloud DC. Using virtual machines (VMs) to run microservices or services with limited resources needs may lead to inefficient resources' utilization. For this reason, Virtualization might not be the best mechanism to use. Containerization, which is an OS-level Virtualization, can be the proposer alternative. With Containerization, microservices/services can run in containers and consume only the required amount of resources, hence improving the overall hardware utilization of edge servers [9].

This paper presents an SDN-based resource allocation and service placement mechanism in an Edge-Cloud environment. In the proposed system, the SDN controller is responsible for offloading services to an edge server or the cloud. The suitable destination to offload the service will be determined according to the priority of the service, the resources' usage of the edge servers, and the load on the edge servers. By leveraging the Containerization mechanism, edge servers will run services/microservices inside containers. Docker containers are used as a Containerization platform. The main contributions of this work are:

- Reducing the total time for handling IoT service's requests by presenting an efficient SDN assisted offloading mechanism to determine the convenient edge server for each service/microservice using multiple-criteria decision-making (MCDM) algorithms.
- Specifying the servers state, which would determine the acceptable and suspended services for the edge server by monitoring the resources' usage of the edge servers.

- Ensuring an efficient service distribution and load balancing between edge servers to improve the performance of the services/microservice and to grant a better utilization of the edge servers' resources.

The rest of the paper will be organized as follows. Section 2 displays a summary of the related works. Section 3 presents the proposed system architecture and the tools and technologies used in the system. A detailed description of the system implementation and methodology is presented in section 4. Results and discussion are presented in section 5, and the conclusion is drawn in section 6.

2. Related Works

The literature is rich with resources that have studied the Edge Computing technology and the positive aspects of using it with Cloud Computing. It is proved in almost all researches, that the use of Edge Computing has improved the performance of IoT-Cloud platforms. Authors in an earlier work [10] presented a fog-based IoT-healthcare system. The results showed an improvement in network delay and energy consumption. Other authors [11, 12] introduced smart home systems based on Fog-Cloud environment. In another study [13], the authors suggested a smart campus system to enhance real-time service provisioning and application management. With Edge-Cloud computing, problems of resource allocation and service placement in the edge network have appeared. Many resources have discussed these subjects. For example, Minh *et al.* presented a service placement approach in Fog-Cloud environment in which services could be processed in the Cloud, Fog, or IoT devices, depending on their requirements. The proposed approach showed a reduction in latency, energy consumption, and network load [1]. On the other hand, Lui *et al* proposed a multi-objective Mixed Integer Linear Programming (MILP) model to select an optimal Cloudlet from multiple Cloudlets. In the proposed model, the nearest Cloudlet with the highest mean reward and the lowest latency is preferred. The measured results were based on storage and bandwidth [14]. In another investigation [15], Xu *et al* suggested a model that breaks services to multiple subsets according to the request start time and detects spare space in the computing nodes. Nodes with the lowest and enough spare space are selected. Furthermore, they proposed a load balancing scheme in which workloads can be migrated from computing nodes with high resources' usage to others with lower resources' usage. In another work [16], the authors proposed a VM scheduling method in Fog-Cloud environment with VM live migration for load balancing. An application-aware workload allocation scheme was also proposed [17]. The applications handled by VMs are optimally allocated in the closest suitable cloudlets. The results showed improvement in the response time. Zhao *et al* presented edge resources allocation algorithms for multiple applications to minimize average service response time. The results were measured and compared on three algorithms, where the Clustering-Based Heuristic Edge Resource Allocation (CHERA) was preferred due to its higher computational efficiency [18].

Other studies suggested the use of SDN to reduce network congestion and delay. Aujla *et al* [19] presented a workload slicing scheme and an energy-aware inter-DC migration control scheme using SDN with the Stackelberg game to provide an optimal inter-DC migration. The results were evaluated depending on energy consumption, delay, Service-Level Agreement (SLA) violation, and migration rate. Other authors [7] proposed a healthcare system using SDN for forward/reverse data offloading and flow management across multi-region edge DC. The results were measured depending on delay, complexity, and number of handovers.

3. System Architecture

The system architecture is composed of three layers, as shown in Figure-1. The first layer contains various types of IoT devices. The second layer contains edge servers with different hardware capabilities distributed in a local region close to the IoT devices. Furthermore, it contains an SDN controller that is responsible for the flow management and service offloading among the end devices, the edge servers, and the cloud. The third layer contains the cloud DC which is assumed to have huge resources.

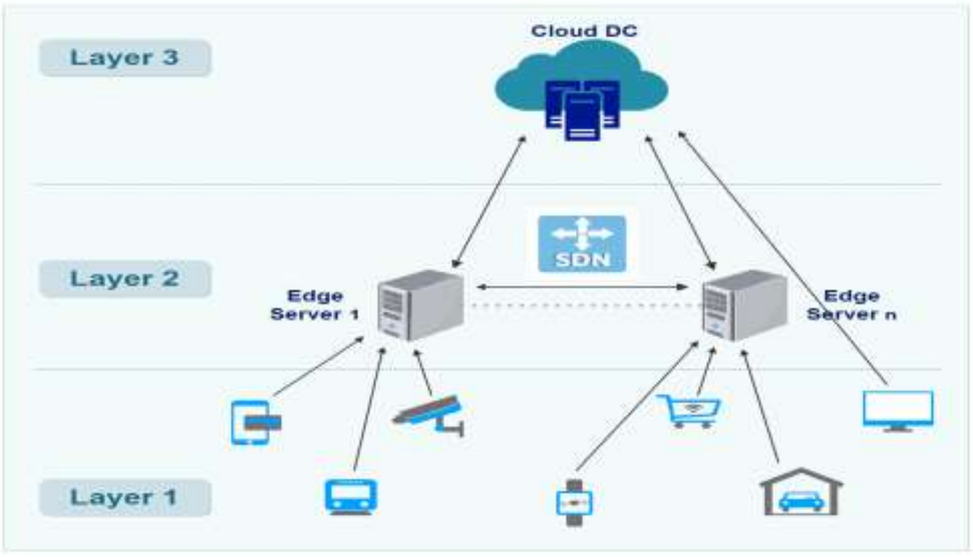

Figure 1- The layered architecture of the proposed system.

3.1. Edge Servers with Containerization Mechanism

The recent years have shown a transformation in the style of application development, from monolithic stand-alone applications to microservices. Microservice is an architectural style that splits a single application to small parts, each of which runs as an independent process [20, 21]. Every microservice is responsible for a specific task. Microservices can be executed in multiple machines and communicate with each other using specific APIs [22]. For such cases, Containerization is considered as the best solution for running microservices/services with limited needs. Containers are faster and more lightweight than VMs, which ensures better resources' utilization, less overhead, better versioning control, and an improved overall system and network performance [23]. In the proposed system, the all edge server supports Docker Containerization. For each microservices/services, there would be a Docker image. These Docker images would be available in the edge servers that provide the service. Using Docker images, an edge server would run a container for each required service, as shown in Figure-2.

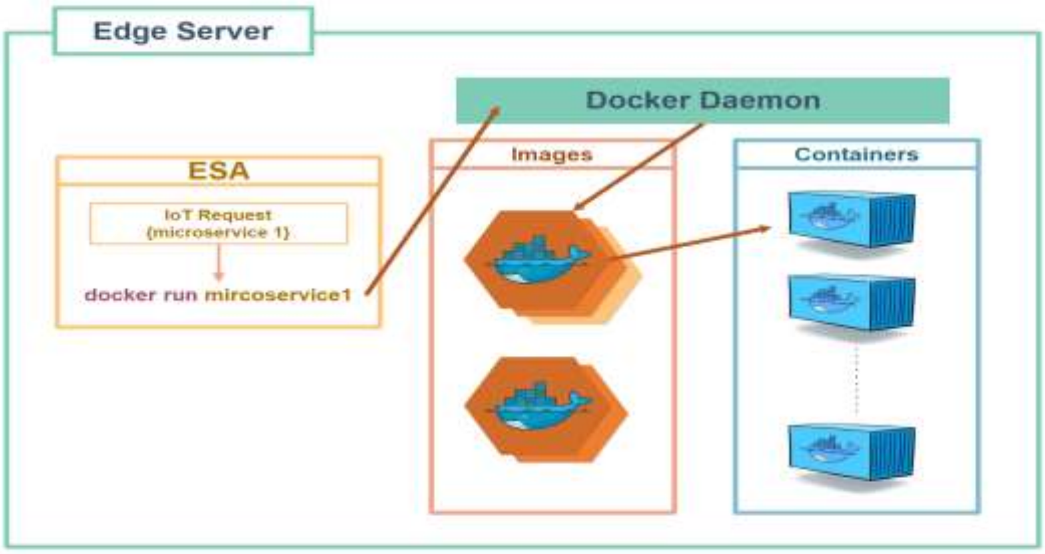

Figure 2- Microservices deployment with Docker containerization.

3.2. SDN-based Service Management Model

In the Edge-Cloud environment that supports various types of applications with enormous services/microservices and a large number of IoT devices, the use of SDN will have positive effects not only on the flow management but also on the service discovery and placement. Considering an IoT device requests a service that requires a quick response time. With multiple edge servers in the proximity of the device, it is a big challenge to know which is the best edge server that provides the service and responds with a minimum delay. In such a case, the IoT device should send the request to all reachable edge servers and wait for a response. The device would not be able to know which is the least overloaded edge server or edge server with the least link delay. Using SDN, service discovery would be easier. The SDN controller would have complete information about the edge servers which includes the server state and the amount of the available resources, while it can also route traffic into the least congested paths. Furthermore, it can monitor the edge servers' resources state and balance the load between the edge servers [8].

4. The proposed System: Methods and Algorithms

By considering the issues and solutions discussed in the previous section, this work proposes a system that facilitates service discovery between IoT devices and edge servers in the edge network and ensures load balancing between edge servers. The following subsections present an extensive description of the proposed methods and the implementation of the system.

4.1. SDN Assisted Service Placement in Multi-Edge Environment (SASPME)

The proposed system has two parts, shown in Figure-3, which are the Edge Server Application (ESA) and the SDN Non-core Application (SNA). The following subsections describe each part of the system.

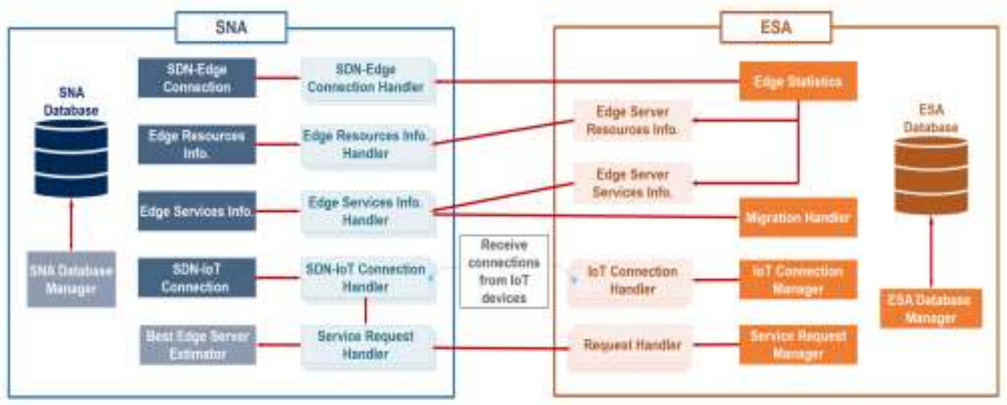

Figure 3- The proposed SASPME (the software design of SNA and ESA).

A. The Edge Server Application (ESA)

ESA would run inside each edge server. It has connections with the SNA and the IoT devices. The main parts of ESA shown in Figure-3 are described below.

1) *Edge Statistics:* monitors the state of the edge server and the utilization of resources and gathers information about services. It has three sockets, as described below.

•SDN-Edge client socket: it registers the edge server in the SNA and sends the total resources information and the available services in the edge server.

•Edge Resources Information client socket: it sends information about the edge resources' utilization (CPU percentage, available memory, free disk space), the number of running services, and the server state (overloaded or normal). Furthermore, it sends a list of suspended services. When the resources' utilization of the edge server exceeds a specific limit, ESA would check the amount of the resources utilized by each service. If a service usage of a specific resource would increase the usage of this resource above the acceptable limit, then this service would be added to the suspended services list.

•Edge Services Information client socket: it would send information about running and exited services (containers). The information includes the container name, IoT IP, minimum and maximum CPU usage, minimum and maximum memory usage, the container start time, minimum and maximum execution time (for exited containers), and the container state. The container state will determine whether a service (container) is running for a long time.

2) *Service Request Manager:* it is responsible for managing IoT requests received from SNA on the Service Request server socket, which receives the IoT request forwarded from the SNA and generates a token for each IoT device-service request.

3) *IoT Connection Manager:* it manages the connection with IoT devices, it has the IoT connection server socket that receives connections from the IoT devices, and runs containers for services after checking the IoT token.
4) *Migration Manager:* it has a server socket which would receive notification from SNA to reject an IoT device-service, when SNA decides to migrate (horizontally offload) a service from this edge server to another edge or the cloud.
5) *ESA Database Manager:* is responsible for ESA database management (insertion, deletion, and adaptation of the data).

B. The SDN Non-core Application (SNA)

SNA would be responsible for the service placement in the edge-cloud network. SNA has connections with the ESA in the edge servers and with the IoT devices, as shown in Figure-3. The details of each part are described below.
1) SDN-Edge Connection: it creates a thread that receives connections from edge servers and gets information about the total resources of the edge server and the available services.
2) Edge Resources Information: it creates a thread that receives information about the edge server resources' utilization, the number of running services, suspended services list, and server state. If the server is overloaded (CPU/memory usage exceeds 80%), SNA would not offload new services to this edge server, and it would decide whether to migrate (horizontally offload) some services to another edge or to the cloud.
3) Edge Services Information: it creates a thread that collects information about services in the edge servers to manage services migration. It has two sockets.
•Services Information server socket: it receives information about running and exited services from each edge server.
•Service Migration client socket: it sends a notification to the edge server to reject an IoT device connected to a specific service.
4) SDN-IoT Connection: it receives connections from the IoT devices. According to the priority of the service, SNA would decide whether to offload it to an edge server or the cloud. Moreover, it would choose the best edge server according to the resources' utilization, number of running services, and distance between each edge and the IoT devices.
5) Service Request Handler: it sends the IoT device request to the best destination (edge or cloud), receives a token from the edge server, and forwards it back to the IoT device.
6) SNA Database Manager: is responsible for SNA database management (insertion, deletion, and adaptation of the data).

4.2. SASPME Offloading Schemes

The proposed system has two schemes for service offloading, namely the vertical offloading and horizontal offloading. Both are described below.

A. Vertical offloading

When an IoT device requests service for the first time, it would send a request to SNA which would estimate the best destination to offload the service. The following steps describe the vertical offloading scheme shown in Figure-4.
1. SNA should have information about the total resources of each edge server connected to the network, and it should receive periodic updates from edge servers about server state and resources' utilization.
2. IoT devices would send service requests to SNA. The request should include the name and the priority of the service. Priority will determine whether a service is delay-sensitive or delay-tolerant. According to the priority of the service, SNA will decide whether to offload it to an edge server or the cloud. Delay-sensitive services would be offloaded to an edge server, while delay-tolerant services would be offloaded to the cloud.
3. For delay-sensitive services, SNA should offload the service to the best edge server. An MCDM model is used to estimate the best edge server.
4. After estimating the best edge server, SNA would forward the IoT request to that server. When the edge server receives the request, it would generate a token for this request (IoT device-service) and send it back to SNA.
5. SNA would forward the response received from the edge server to the end device. The IoT device can then start the connection with the edge server.

6. The IoT device would connect to the edge server. While receiving the IoT connection, the edge server would check the token. If tokens match, the edge server would start the container of the service, and the IoT device would be able to send its data to be processed in that container.

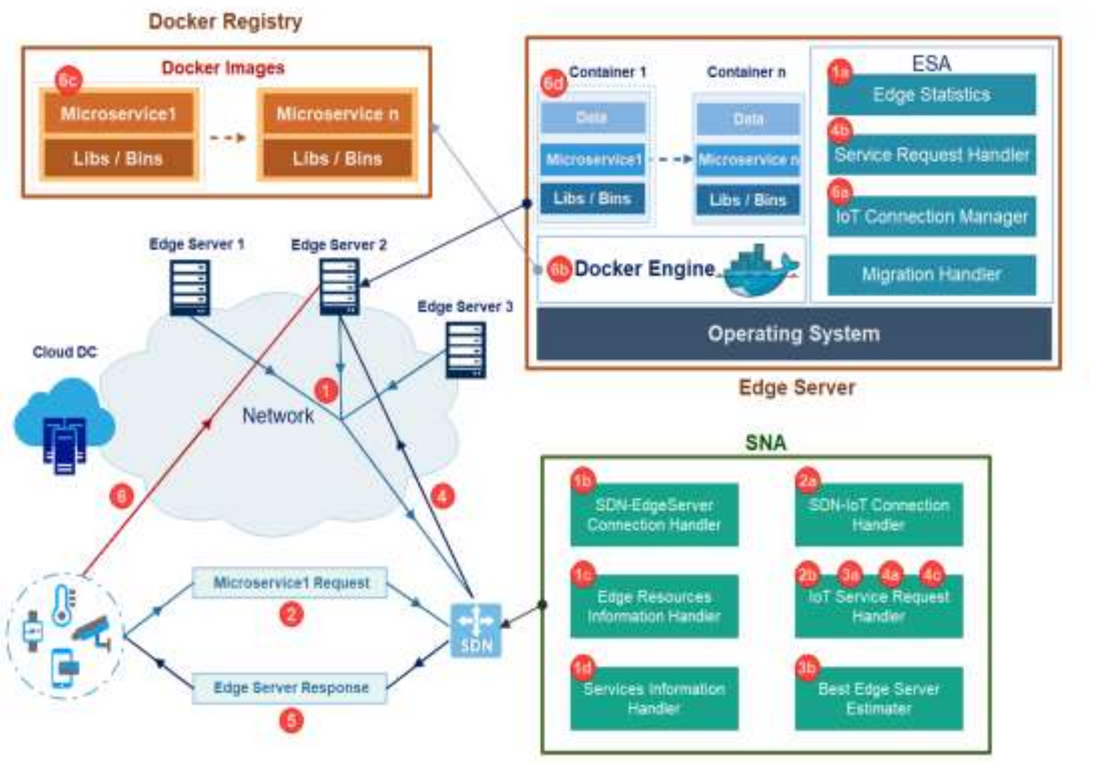

Figure 4- Vertical offloading (actions taken from receiving a service request until service launching).

B. Horizontal offloading (migration)

In this case, services would be migrated (horizontally offloaded) from one edge server to another or from one edge server to the cloud. The following steps describe the horizontal offloading scheme that is shown in Figure-5.

1. When the state of an edge server is changed to overloaded, SNA will check the currently running services in that edge server. For services that are running for a long time, if they are exploiting high resources, they would be migrated (horizontally offloaded) to the cloud or, if they do not need high resources, they would be migrated to another edge server. Otherwise, no services would be offloaded to that edge server, until the state of the edge server is changed to normal.
2. After the decision has been taken to migrate a service, SNA would inform the edge server by sending the service name and the IP of the IoT device which had requested the service. The edge server would reject any future connection from this IoT device to that specific service.
3. If the edge server has rejected the IoT device connection, the IoT would request the service again from SNA. SNA would offload the service to a new edge server or the cloud according to the previous decision.

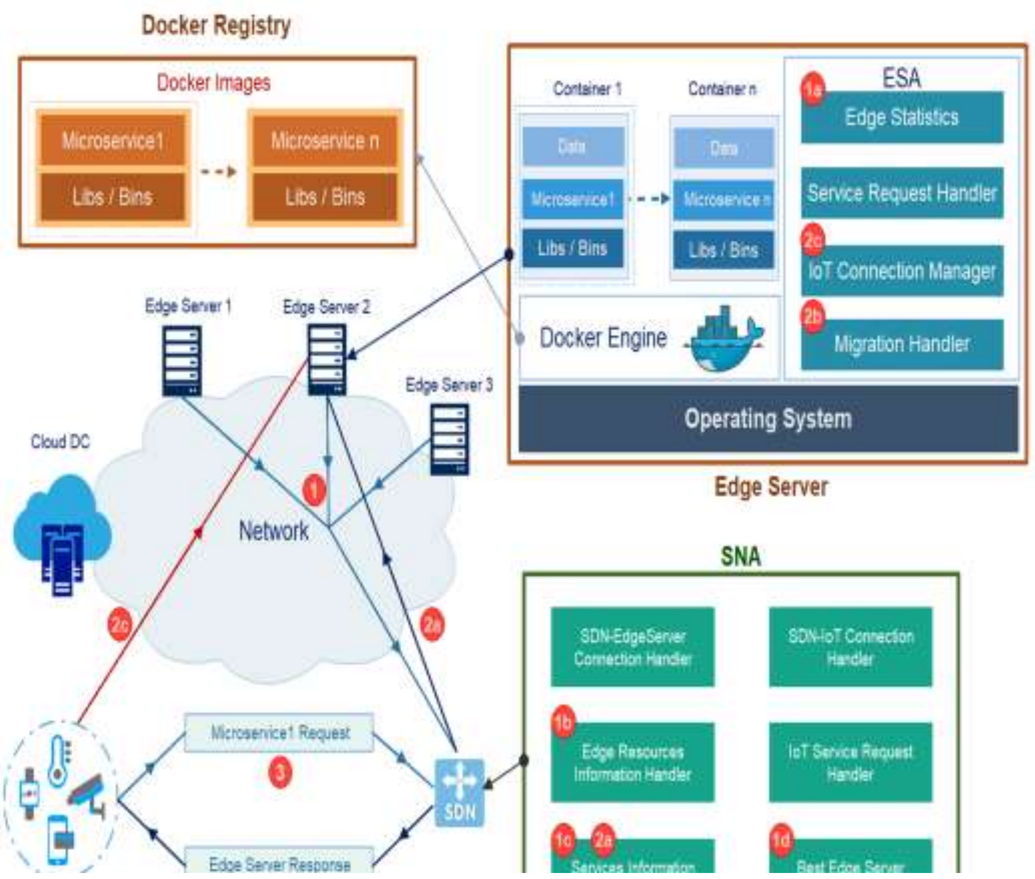

Figure 5- Horizontal offloading (actions taken when an edge server reaches an overloaded state).

Edge Servers' State Determination and Suspended Services Selection

Exhausting the resources of an edge server may lead to degradation in the performance of all the running services. Therefore, monitoring the resources in edge servers will help SNA to determine the state of the edge servers and balance the load between them. Considering multiple edge server environment, let *E* be a set of the edge servers connected to the network, where e_i is an element in the set and *n* is the number of the elements in *E*. Let *S* be a set of services provided by each edge server in *E*, where s_i is an element in the set and *m* is the number of elements in *S*. The memory and CPU usage of every e_i in *E* is denoted by Me_i and Ce_i. It is important to ensure that Me_i and Ce_i do not reach high limits. ESA should periodically check these values. As presented in Algorithm-1, when Me_i and Ce_i exceed 70% of their limits, ESA would determine which services to be suspended depending on their needs for resources. The decision is taken according to js_i, ks_i, ls_i, and ms_i, which represent maximum memory, minimum memory, maximum CPU, and minimum CPU utilization, respectively. Services that are added to suspended list *SL*, would be sent to SNA. *MF* and *CF* are flags that indicate the state of the memory and CPU in edge servers. When Me_i or Ce_i exceed the limits specified in Algorithm-1, *MF*/*CF* would be set to *True* and the state of the edge server would be "overloaded". Otherwise, if both of them is *False*, the state of the edge server would be "normal".

Algorithm 1: edge server's state estimation and suspended services selection

```
Requires: $E, S, Me_i, Ce_i, TMe_i, js_i, ks_i, ls_i, ms_i$        ► ($TMe_i$ : total memory of $e_i$)
Ensures: $SL, MF, CF$
1: $SL \leftarrow None, MF \leftarrow False, CF \leftarrow False$
2: if $Me_i > 70\%$ then
3:    for $s_i \in S$ do
4:        if $((js_i > (TMe_i - Me_i))$ and $(ks_i > (TMe_i - Me_i)))$ or $((ks_i + js_i)/2 > (TMe_i - Me_i))$ then
5:            $SL \leftarrow s_i$
6:        end if
7:    end for
8: else if $Me_i > 90\%$ then
9:    for $s_i \in S$ do
10:       if $(js_i > (TMe_i - Me_i))$ or $(ks_i > (TMe_i - Me_i))$ then
11:           $SL \leftarrow s_i$
12:           $MF \leftarrow True$
13:       end if
14:   end for
15: end if
16: if $Ce_i > 70\%$ then
17:   for $s_i \in S$ do
18:       if $(((ls_i + Ce_i) > 75\%)$ and $((ms_i + Ce_i) > 75\%))$ or $(((ls_i + ms_i)/2 + Ce_i) > 75\%)$ then
19:           $SL \leftarrow s_i$
20:       end if
21:   end for
22: else if $Ce_i > 80\%$ then
23:   for $s_i \in S$ do
24:       if $((ls_i + Ce_i) > 83\%)$ or $((ms_i + Ce_i) > 83\%)$ then
25:           $SL \leftarrow s_i$
26:           $CF \leftarrow True$
27:       end if
28:   end for
29: end if
30: return $SL, MF, CF$
```

4.3. Estimation of The Best Edge Server

In SASPME, SNA would periodically receive information about resources' utilization from all edge servers. To estimate the most suitable edge server for an IoT service request, SNA would decide according to the available resources, the load on the edge server, and the distance between the edge servers and the IoT device. To compare such divergent types of data, MCDM algorithms are used. With MCDM, multiple alternatives are evaluated and ranked depending on multiple criteria [24, 25]. In the proposed system, edge servers (alternatives) would be ranked according to the number of running services, CPU usage, available memory free disk space, and the distance between the edge servers and the IoT devices to estimate the most suitable edge server. The inputs of the decision matrix are shown in Table-1.

Table 1- The inputs of the decision matrix.

	No. of Running Services	CPU Percentage	Available Memory	Available Storage	Distance
Edge server 1	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
Edge server 2	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}
Edge server 3	X_{31}	X_{32}	X_{33}	X_{34}	X_{35}

Because the data are in different types, the matrix should be normalized first to be comparable. The normalization procedure used in this work is the same as previously described [24].
For beneficial criteria, *i.e.* available memory and available storage, where higher values are desired, we have

$$X`_{ij} = \frac{X_{ij}}{MAX_{i}(X_{ij})} \quad (1)$$

For non-beneficial criteria, i.e. the number of running services, CPU percentage, and distance, where lower values are desired, we have

$$X`_{ij} = \frac{MIN_{i}(X_{ij})}{X_{ij}} \quad (2)$$

In this work, the weighted aggregated sum product assessment (WASPAS) method is used. This method is a combination of the Weighted Sum Method (WSM) and the Weighted Product Method (WPM). These methods are detailed below.

A. Weighted Sum Method (WSM)

It is a simple method in which each criterion has a specific weight w_j. The sum of all the weights should be equal to one. It is calculated according to the following equation.

$$Qi_{(1)} = \sum_{j=1} X`_{ij} w_j \quad (3)$$

The results are sorted in a descending order, and the highest result would represent the best choice.

B. Weighted Product Method (WPM)

It is similar to WSM but with some differences. Multiplication is used instead of addition, and the criterion is raised to the power of the weight, as shown in the equation below.

$$Qi_{(2)} = \prod_{j=1} (X`_{ij})^{w_j} \quad (4)$$

After sorting the results in a descending order, the highest result would represent the best choice.

C. Weighted Aggregated Sum Product Assessment Method (WASPAS)

This method is a combination of the WSM and WPM. The following equation presents a joint generalized criterion of weighted aggregation of additive and multiplicative methods.

$$Q = 0.5\, Qi_{(1)} + 0.5\, Qi_{(2)} = 0.5 \sum_{j=1} X`_{ij} w_j + 0.5 \prod_{j=1} (X`_{ij})^{w_j} \quad (5)$$

Same as the previous methods, results are sorted in a descending order, and the alternative (edge server) with the highest result would be the best choice.

5. Results and Discussion

To test the effectiveness of the proposed system, experiments were implemented for two cases: 1) without using the SDN controller; 2) using the SDN controller. The first case includes ESA only. IoT devices would send requests to all reachable edge servers using broadcast messages. The IoT device would choose the edge server with a quick response time without considering the distance and the load on that edge server. In the second case, the proposed SASPME (SNA and ESA) would be implemented. The experiments were executed using two physical machines. The first machine is used to run the SDN controller (Onos) and three VMs, with a VM for each edge server. The second physical machine is used to run two VMs, one is for the Cloud and the other is for IoT devices. The specifications of the physical machines and VMs are presented in Table-2. All VMs in the system are connected to a Mininet network that has an Openflow enabled switch. These networks are connected through Generic Routing Encapsulation (GRE) tunnels. Figure-6 shows the system setup. The results were measured for a different number of devices (10, 20, 30, 40, and 50) in both cases. Each device requests a single service. To explain the impact of running various types of services on the resources' utilization of the edge servers, three types of services were implemented. An edge detection service that implements edge detection algorithms, RSA (Rivest–Shamir–Adleman), and SHA-3 (Secure Hash Algorithm 3) cryptography algorithms were used. The time between a device request and another was randomly selected between 7ms and 20ms. Each device will reconnect to the edge server at a random time between 1 to 3 minutes. The next subsections present the results collected from both cases.

Table 2- Specifications of physical machines and VMs.

Physical Machines	VMs in Each Physical Machine
Processor – Intel Core i7-8550u CPU @1.80 GHz ×8 ***Memory*** – 16 GB ***Disk*** – 1TB ***Operating system*** – Ubuntu 18.04.3 LTS (64-bit)	***Processor*** – Intel Core i7-8550u CPU @1.80 GHz ×2 ***Memory*** – 2 GB ***Disk*** – 70GB ***Operating system*** – Ubuntu 18.04.3 LTS (64-bit)
Processor – Intel Core i5-4210u CPU @1.70 GHz 2.40 ***Memory*** – 8 GB ***Disk*** – 512GB ***Operating system*** – Windows 10 Pro (64-bit)	***Processor*** – Intel Core i5-4210u CPU @1.70 GHz ×2 ***Memory*** – 2 GB ***Disk*** – 70GB ***Operating system*** – Ubuntu 18.04.3 LTS (64-bit)

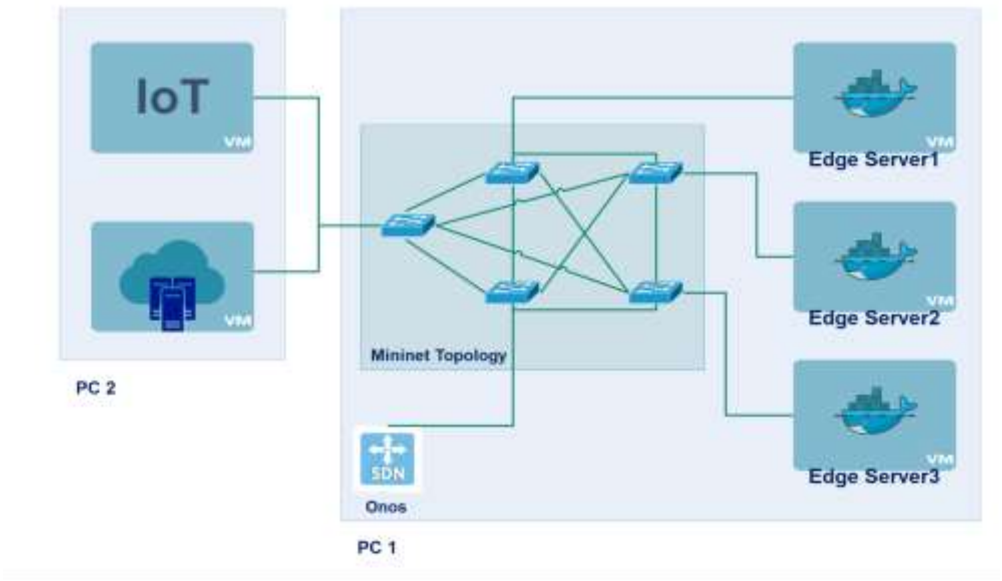

Figure 6- System Setup (connections between the physical and the virtual machines).

The CPU and memory utilization were measured for both cases. In the first case, the type of service for IoT device was chosen randomly and requests were sent directly to edge servers. The IoT devices offloaded their data to the edge server with the quickest response time. The results in Figure-7 show an inequitable load distribution between edge servers. Edge server 1 was in an overloaded state for a long time, while edge servers 2 and 3 remained in a normal state where the CPU and Memory usage were mostly in low rates. Overloaded and normal states are represented by 0 and 1, respectively. Furthermore, the services running in an overloaded edge server are prone to failures or can negatively affect the performance of these services. In the second case, when SNA and ESA are used, the resources of edge servers were utilized more efficiently and the edge servers were in a normal state all the time. Resources' utilization of edge servers was measured and sent to SNA every 30 Seconds. The resources' utilization measurement, collected from the edge servers, have improved future offloading decisions and assured an efficient services distribution. The resources' utilization and the state of edge servers, along with the number of running devices in all edge servers for 10 devices in both cases are shown in Figures-(7 and 8), respectively.

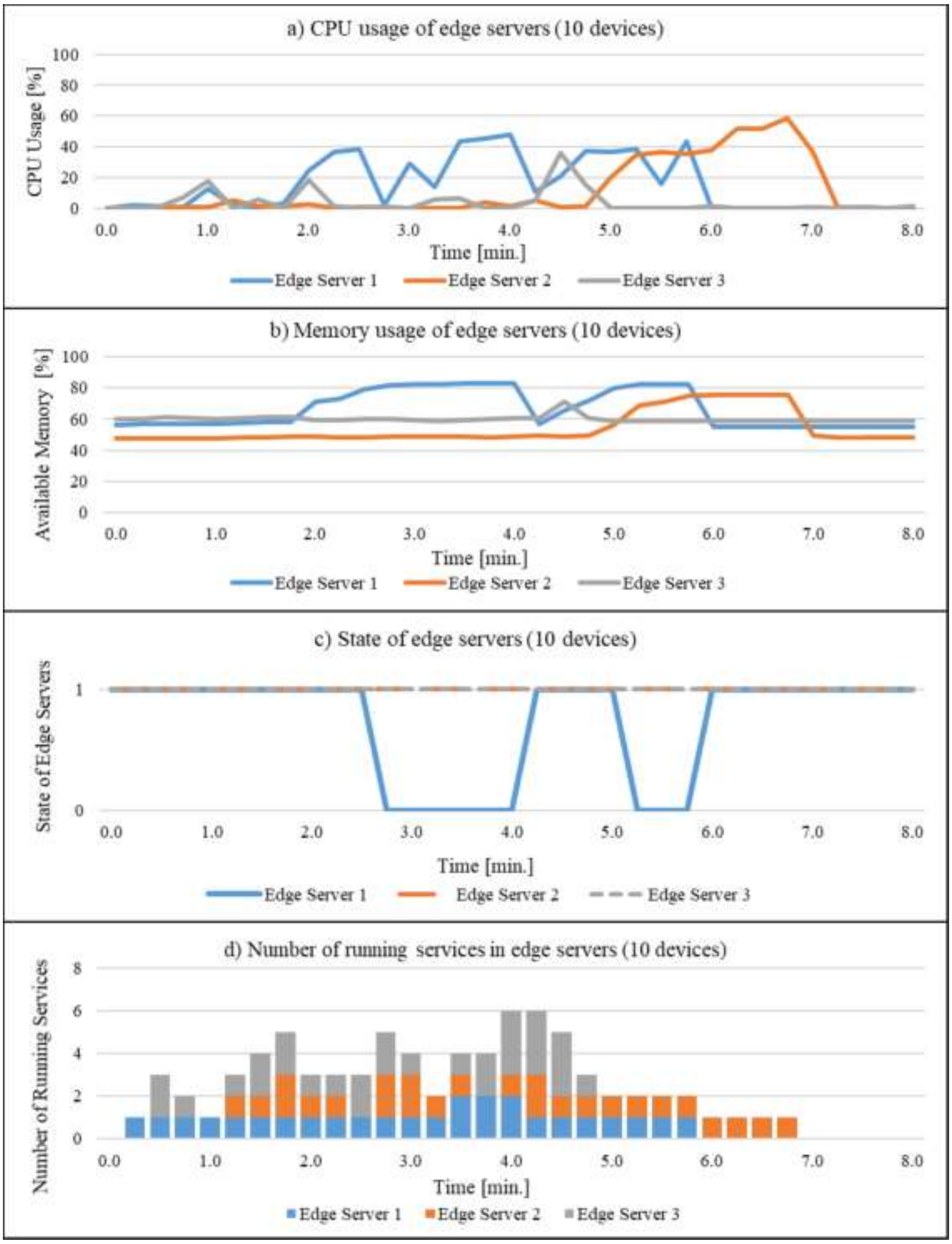

Figure 7- Resources usage of the edge servers using ESA.

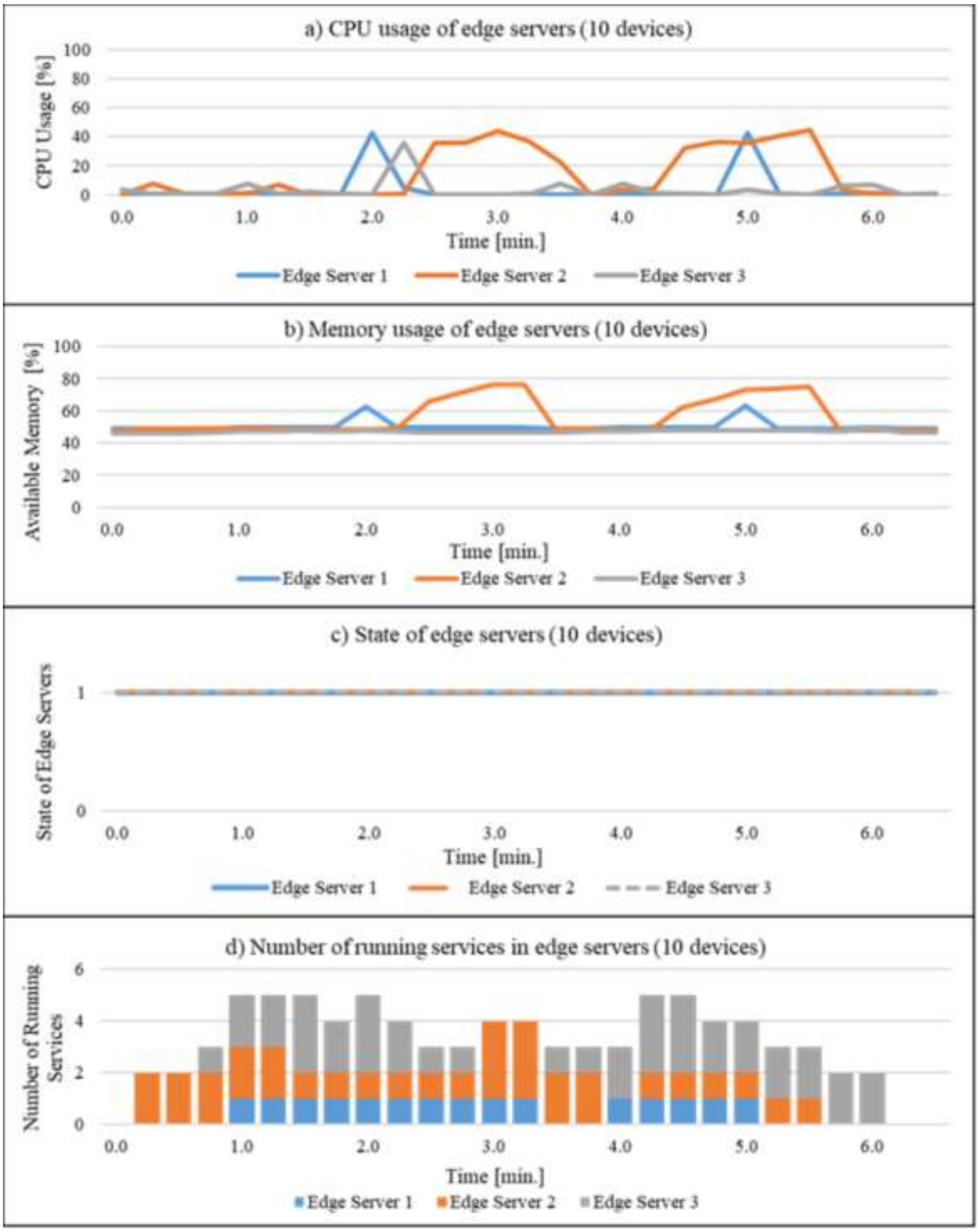

Figure 8- Resources usage of the edge servers using ESA and SNA.

The request handling time was measured for five experiments with a different number of devices. In the first case, the total request handling time was measured by the period in which an IoT device sends a service request and receives a response from an edge server. Figure-9, which presents the average request handling time for different devices using ESA only, shows that the average request handling time rose in varying rates by increasing the number of IoT devices. In this case, the IoT requests is broadcasted to all the edge servers and they should all respond to each received request, even though it may not be completed. This has increased the overhead on the edge servers, and therefore, it has increased the request handling time.

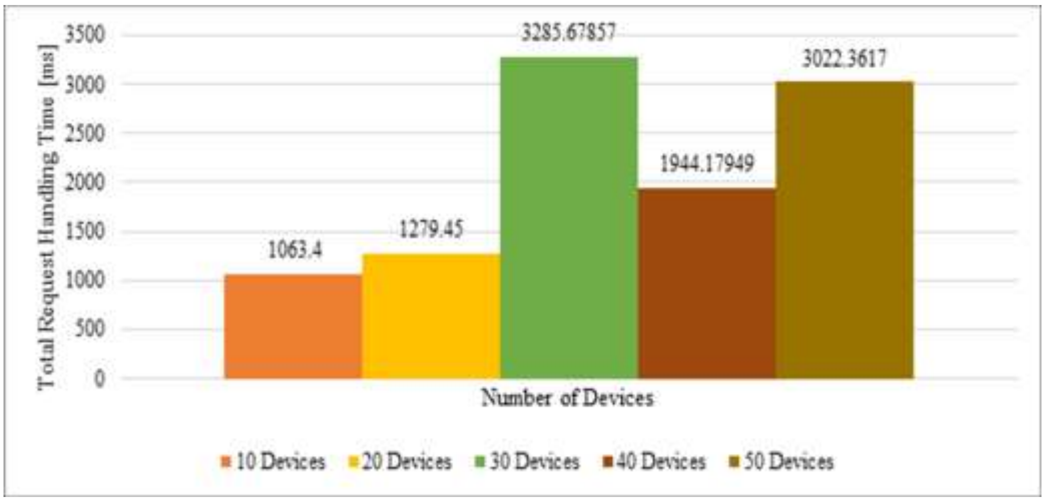

Figure 9- Average request handling time using ESA.

In the second case, the offloading destination would be determined by the priority of the requested service. Therefore, the time to handle requests received from IoT devices would depend on the destination estimation time and the connection time between SNA and ESA. For delay-sensitive services, the request handling time would include the time to choose the best edge server and the time to forward the request to that edge server. As shown in Figure-10, the time to estimate the best edge server was relatively close in all cases. Figure-11 shows the average time to forward the request to the edge server.

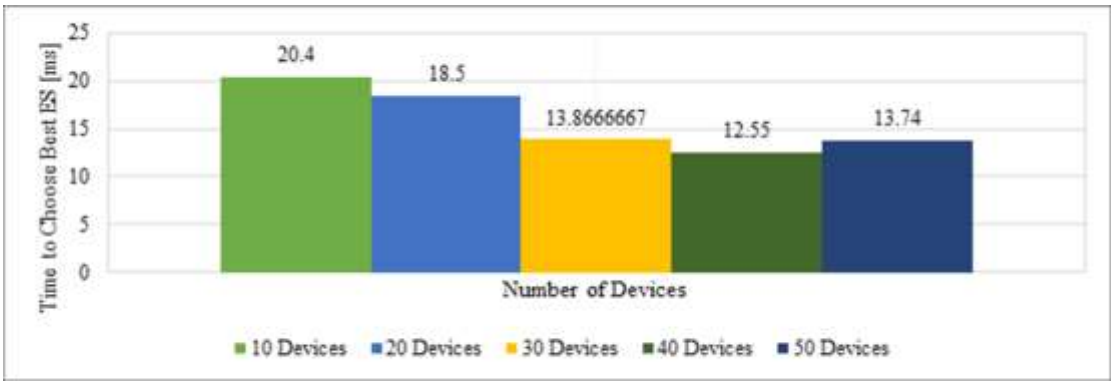

Figure 10-The time to estimate the best edge server.

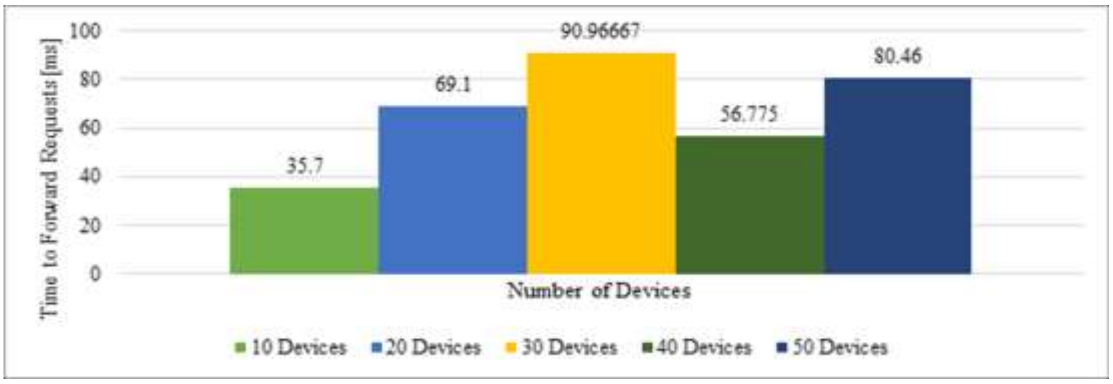

Figure 11- The average time to forward requests to edge servers.

Requests for delay-tolerant services would be directly forwarded to the Cloud. In this case, the request handling time would depend only on the time to forward the request to the Cloud. As shown in Figure-12, the average request handling time using ESA and SNA shows a considerable improvement as compared with the previous case. Also, by increasing the number of devices, the average request handling time remained in convergent rates. In this case, the number of requests sent to each edge server was notably reduced as compared with the previous case. Hence, the average request handling time is minimised.

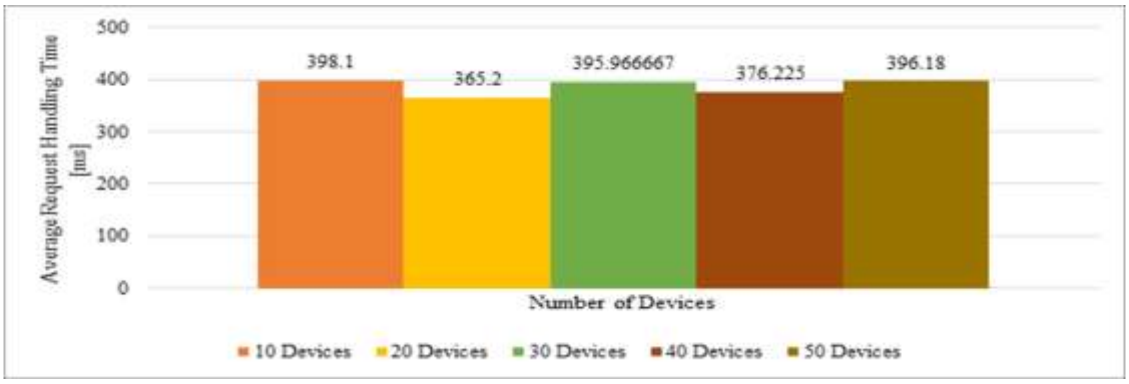

Figure 12- The average request handling time using ESA and SNA.

6. Conclusions

In this paper, we have focused on resources allocation for IoT services in the edge networks. The proposed SASPME system aims to improve the performance of the IoT-based applications by allocating computational resources for the delay-sensitive services in the edge servers. In SASPME, the SDN application is proposed to reduce the overhead on the edge servers by taking the responsibility of handling the IoT requests and taking the offloading decision to the best destination. The experiments showed that gathering information related to the available computational resources from the edge servers in short periods of time can improve the decision making and, hence, ensure a more balanced load distribution between the edge servers, although it may increase the load on the link. The WASPAS method, used in the SDN application, is an effective solution to estimate the best edge server depending on different criteria. Furthermore, containerization is used in the edge servers to ensure efficient utilization of resources. SASPME is compared with a non-SDN system. We conclude that, for a different number of devices, the SASPME can reduce the total time for handling requests, improve the resources' utilization, and distribute the load more efficiently.

References

1. Minh, Q.T., Nguyen, D.T., Le, A.V., Nguyen, H.D. and Truong, A. **2017**. Toward Service Placement on Fog Computing Landscape, 4th NAFOSTED Conference on Information and Computer Science, November 24-25, Hanoi, Vietnam. Doi:10.1109/NAFOSTED.2017.8108080
2. He, Y., Yu, F.R., Zhao, N., Leung, V.C.M. and Yin, H. **2017**. Software-Defined Networks with Mobile Edge Computing and Caching for Smart Cities: A Big Data Deep Reinforcement Learning Approach, *IEEE Communications Magazine*, **55**(12): 31-37. DOI: 10.1109/MCOM.2017.1700246
3. Abbas M. N., Attea B. A., and Kadhim N. J. **2018**. Evolutionary Based Set Covers Algorithm with Local Refinement for Power Aware Wireless Sensor Networks Design, *Iraqi Journal of Science*, **59**(4A): 1959-1966. Doi:10.24996/ijs.2018.59.4A.20
4. Aazam, M., Khan, I., Alsaffar, A.A. and Huh, E. **2014**. Cloud of Things: Integrating Internet of Things with Cloud Computing and the Issues Involved, International Bhurban Conference on Applied Sciences & Technology, January 14-18, Islamabad, Pakistan. Doi: 10.1109/IBCAST .2014.6778179
5. Ghani R. F., and Ajrash A. S. **2018**. Quality of Experience Metric of Streaming Video: A survey, *Iraqi Journal of Science*, **59**(3B): 1531-1537. Doi:10.24996/ijs.2018.59.3B.19
6. Li, S., Zhang, N., Lin, S., Kong, L., Katangur, A., Khan, M.K., Ni, M. and Zhu, G. **2018**. Joint Admission Control and Resource Allocation in Edge Computing for Internet of Things, *IEEE Network*, **32**(1): 72-79. Doi: 10.1109/MNET.2018.1700163
7. Aujla, G.S., Chaudhary, R., Kaur, K., Garg, S., Kumar, N. and Ranjan, R. **2019**. SAFE: SDN Assisted Framework for Edge-Cloud Interplay in Secure Healthcare Ecosystem, *IEEE Transactions on Industrial Informatics*, **15**(1): 469-480. Doi: 10.1109/TII.2018.2866917
8. Baktir, A.C., Ozgovde, A. and Ersoy, C. **2017**. How Can Edge Computing Benefit from Software-

Defined Networking: A Survey, Use Cases & Future Directions, *IEEE Communications Surveys & Tutorials*, **19**(4): 2359-2391. Doi: 10.1109/COMST.2017.2717482

9. Muñoz, R., Vilalta, R., Yoshikane, N., Casellas, R., Martínez, R., Tsuritani, T. and Morita, I. **2018**. Integration of IoT, Transport SDN, and Edge/Cloud Computing for Dynamic Distribution of IoT Analytics and Efficient Use of Network Resources, *Journal of Lightwave Technology*, **36**(7): 1420-1428. Doi: 10.1109/JLT.2018.2800660

10. Mahmud, R., Koch, F.L. and Buyya, R. **2018**. Cloud-Fog Interoperability in IoT-enabled Healthcare Solutions, 19th International Conference on Distributed Computing and Networking, January 4-7, Varanasi, India. Doi: 10.1145/3154273.3154347

11. Yassine, A., Singh, S., Hossain, M.S. and Muhammad, G. **2019**. IoT Big Data Analytics for Smart Homes with Fog and Cloud Computing, *Future Generation Computer Systems*, **91**: 563-573. Doi: 10.1016/j.future.2018.08.040

12. Javaid, N., Butt, A.A., Latif, K. and Rehman, A. **2019**. Cloud and Fog based Integrated Environment for Load Balancing using Cuckoo Levy Distribution and Flower Pollination for Smart Homes, International Conference on Computer and Information Sciences (ICCIS), April 3-4, Sakaka, Saudi Arabia. Doi: 10.1109/ICCISci.2019.8716467

13. Tang, C., Xia, S., Liu, C., Wei, X., Bao, Y. and Chen, W. **2019**. Fog-Enabled Smart Campus: Architecture and Challenges, International Conference on Security and Privacy in New Computing Environments, April 13-14, Tianjin, China. Doi: 10.1007/978-3-030-21373-2_50

14. Liu, L. and Fan, Q. **2018**. Resource Allocation Optimization based on Mixed Integer Linear Programming in the Multi-cloudlet Environment, *IEEE Access*, **6**: 24533-24542. Doi: 10.1109/ ACCESS.2018.2830639

15. Xu, X., Fu, S., Cai, Q., Tian, W., Liu, W., Dou, W., Sun, X. and Liu, A. X. **2018**. Dynamic Resource Allocation for Load Balancing in Fog Environment, *Wireless Communications and Mobile Computing*, **2018**: 6421607. Doi: 10.1155/2018/6421607

16. Xu, X., Liu, Q., Qi, L., Yuan, Y., Dou, W. and Liu, A.X. **2018**. A Heuristic Virtual Machine Scheduling Method for Load Balancing in Fog-Cloud Computing, IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS), May 3-5, Omaha, NE, USA. Doi: 10.1109/BDS /HPSC /IDS18.2018.00030

17. Fan, Q. and Ansari, N. **2018**. Application Aware Workload Allocation for Edge Computing based IoT, *IEEE Internet of Things Journal*, **5**(3): 2146-2153. Doi: 10.1109/JIOT.2018.2826006

18. Zhao, L., Wang, J., Liu, J. and Kato, N. **2019**. Optimal Edge Resource Allocation in IoT-Based Smart Cities, *IEEE Network*, **33**(2): 30-35. Doi: 10.1109/MNET.2019.1800221

19. Aujla, G.S., Kumar, N., Zomaya, A.Y. and Ranjan, R. **2018**. Optimal Decision Making for Big Data Processing at Edge-Cloud Environment: An SDN Perspective, *IEEE Transactions on Industrial Informatics*, **14**(2): 778-789. Doi: 10.1109/TII.2017.2738841

20. Taherizadeh, S., Stankovski, V. and Grobelnik, M. **2018**. A Capillary Computing Architecture for Dynamic Internet of Things: Orchestration of Microservices from Edge Devices to Fog and Cloud Providers, *Sensors*, **18**(9): 2938. Doi: 10.3390/s18092938

21. Buzato, F.H.L., Goldman, A. and Batista, D. **2018**. Efficient Resources Utilization by Different Microservices Deployment Models, 17th International Symposium on Network Computing and Applications (NCA), November 1-3, Cambridge, MA, USA. Doi: 10.1109/NCA.2018.8548346

22. Fernandez, J., Vidal, I. and Valera, F. **2019**. Enabling the Orchestration of IoT Slices through Edge and Cloud Microservice Platforms, *Sensors*, **19**(13): 2980. Doi: 10.3390/s19132980

23. Alam, M., Rufino, J., Ferreira, J., Ahmed, S.H., Shah, N. and Chen, Y. **2018**. Orchestration of Microservices for IoT Using Docker and Edge Computing, *IEEE Communications Magazine*, **56**(9): 118-123. Doi: 10.1109/MCOM.2018.1701233

24. Karande, P., Zavadskas, E.K. and Chakraborty, S. **2016**. A study on the ranking performance of some MCDM methods for industrial robot selection problems, *International Journal of Industrial Engineering Computations*, **7**(3): 399-422. Doi: 10.5267/j.ijiec.2016.1.001

25. Nermend K., Piwowarski M., and Borawski M. **2020**. Decision Making Methods in Comparative Studies of Complex Economic Processes Management, *Iraqi Journal of Science*, **61**(3): 652-664. DOI: 10.24996/ijs.2020.61.3.22.