Towards Elastic Edge Computing Environments: An Investigation of Adaptive Approaches

Abdullah Fawaz Aljulayfi[1, 2] and Karim Djemame[1]
[1]School of Computing
[1]University of Leeds
[1]Leeds, United Kingdom
[2]Prince Sattam Bin Abdulaziz University, Kingdom of Saudi Arabia
e-mail: {ml16afa, K.Djemame}@leeds.ac.uk

***Abstract*— The workload dynamicity of internet of things devices represents a substantial challenge for edge computing environments as it often has limited resources. It requires an efficient elasticity framework that aware of its operational environment in order to adapt in accordance to workload fluctuation which contributes towards efficient resource utilisation, high acceptance rate, and avoids quality of service violation. The edge computing elasticity can be provided through a self-adaptive system that is capable of taking the proper elasticity decisions. This self-adaptive system can be designed using a proactive-, reactive-, or hybrid-adaptation. However, the performance of these adaptation approaches may vary according to the domain, application, and workload. Therefore, this paper designs an edge computing self-adaptive system that can support proactive-, reactive-, and hybrid-adaptation. It also conducts simulation-based investigations on the performance of the adaptation approaches in an edge computing environment under different workloads and application scenarios. The experimental results reveal that the hybrid adaptation performs at least 10% better than other approaches whereas the performance of both proactive and reactive adaptations is application scenarios dependent.**

Keywords- Elasticty; Auto-scaling; Proactive; Reactive; Hybrid.

I. INTRODUCTION

The Internet of Things (IoT) [1] has become a part of everyday life. This technology provides wide-ranging benefits and is extensively used in various domains, such as healthcare and industry, for increased efficiency and productivity [2][3][4]. The massive growth of the IoT devices besides their requirements, such as low latency, location awareness, and mobility, represent a bottleneck for Cloud Computing (CC) [5]. Thus, Edge Computing (EC) considers a promising paradigm to support IoT devices by leveraging the CC resources to the edge of the network addressing their requirements. However, its workload dynamicity represents a main challenge as EC infrastructures often have limited resources [5][6][7][8]. This requires an efficient elastic resource provision framework to cope with the workload dynamicity to support resources scaling up and down in accordance to workload demand. Further, the proper elasticity decision can contribute to avoid both resources over- and under-provisioning.

An agile elasticity framework can be provided via Self-Adaptive System (SAS) which is a promising solution that provides autonomic resource management in such complex systems [9][10][11]. This allows adapting in accordance to the workload dynamicity over time. The SAS can be designed using proactive, reactive, or hybrid adaptation approaches. However, the performance of these approaches vary depending on the domain, application, and workload [12]. Therefore, the investigation of these approaches in the EC environment is an obvious research problem that aims to design an elastic SAS framework using the most appropriate adaptation approach that suites EC requirements.

Our previous work has proposed a Machine Learning Based Context-aware Prediction Framework [13]. This is extended in this paper by conducting thorough empirical investigations and evaluation of the performance of the adaptation approaches in an EC environment. These approaches are evaluated under different IoT devices' workload, application scenarios, and hypotheses.

The main contribution of this paper can be summarised as follows:

- Design an elasticity SAS framework that can support proactive-, reactive-, or hybrid-adaptation where the most proper approach can be selected. The framework itself is supported by four algorithms, namely proactive, reactive, hybrid, and admission control algorithms.
- Profile six IoT applications in a containerised edge environment which help their simulation in EC environments. This profiling considers details about applications' latency requirements, the required resources, and the uploading/ downloading data size, which are specified based on real scenarios.
- Investigate the effectiveness of the elasticity SAS in the EC environment using the adaptation approaches, which shows the suitability of these approaches to the EC. This investigation considers three real workload and two applications scenarios (i.e., mixed applications and single application) and a range of evaluation metrics, e.g., task acceptance rate and servers' utilisation. Additionally, some recommendations are made accordingly about the suitability of the SAS in EC computing environments and its design.

 ISBN: 978-1-61208-854-9

The rest of this paper is organised as follows. Section II discusses the related work and positions the paper. It is followed by Section III which presents the elasticity SAS framework. Section IV illustrates the experimental design. The performance of SAS is evaluated in Section V. The conclusion and future work are presented in Section VI.

II. RELATED WORK

This section presents the related work in relation to the adaptation approaches which are proactive-, reactive-, and hybrid-adaptation. Further, it positions this paper across the literature by highlighting its contributions.

A. Proactive Adaptation

Proactive-based SAS is a SAS that uses the collected historical data to anticipate the future system behaviour or environmental changes [14][15][16]. The future anticipation can be in different folds, such as workload and performance. The main objective of this approach is to act prior to an event occurring which helps to optimise the resource utilisation, avoid Quality of Service (QoS) violation, and support the elasticity [15][17][18].

This approach has been widely used in the literature. For instance, Spatharakis, D. et al. [19] propose a two-layers EC system architecture for location-based services which can be consumed by IoT or mobile devices. It is supported by an offloading decision mechanism and applications' performance requirement profiling mechanism. Furthermore, the Kalman Filtering estimation method is adopted for future request estimation.

In [20], an energy-aware cost prediction framework is presented using Auto Regression Integrated Moving Average (ARIMA) and Linear Regression (LR) as prediction models. The ARIMA is used for predicting Virtual Machines (VM) Central Processing Unit (CPU) utilisation, Random Access Memory-, Disk-write-, and Network-usage. The LR is utilised for predicting the physical machines' CPU utilisation. The work in [20] is extended using the same methods to propose a performance and energy-based cost prediction framework [21]. Support Vector Regression is another Machine Learning (ML) method that is adopted in [16] to introduce an auto-scaling system for web servers. Further, the queuing theory is adopted as a performance model.

An auto-scaling method for containerised micro-services in Fog Computing (FC) environment is proposed in [22]. It is driven by two ML methods, which are Decision Tree Regression and Elastic Net for learning auto-scaling policy and workload forecasting using small and large window size. A thorough evaluation is performed using both synthetic and real workload traces.

B. Reactive Adaptation

The reactive-based SAS is a system that monitors operational environment continuously to trigger a specific event when a condition is satisfied [14]. Although, this kind of adaptation may lead to system instability and late decision, it is the common approach found in the literature.

A multi-agent-based resource provision system architecture for FC is proposed in [23]. It aims to provide a self-adaptive and self-sustainable load-balancing system. It mainly relies on a threshold-based categorisation algorithm that arranges fog nodes based on their workload (i.e., overloaded, underloaded, and balance). The system architecture considers both fog and cloud layers where the cloud layer can be utilised in case there is a need for further processing storage, or the fog layer is fully utilised. The system is evaluated using Poisson distribution synthetic workload that is conducted on iFogSim. In [24], a container-as-service system architecture for task selection and scheduling for real-time data processing in an EC is proposed. This system migrates the containers reactively when over/under-utilisation of the servers is triggered aiming to maintain the Service Level Agreement (SLA) objectives and reduce the energy consumption.

ML methods can also be used in this approach. For example, a resource allocation agent for MEC is developed using deep reinforcement learning [25]. It aims to improve the end-to-end reliability and avoid QoS violation where the decision is made using channel quality, data packet size, and waiting time. The reactive adaptation decisions include changing the scheduling policy and adding/removing tasks.

In [26], a reactive method is introduced for allocating the web-based resource in a CC. It is performed using the user demand targeting the total deployment cost and the QoS. In [27], an energy and SLA-aware self-adaptive resource management scheme for CC is proposed. It uses size of input queue, number of available VMs, and number of provisioned VMs to adjust the number of running physical machines.

C. Hybrid Adaptation

A hybrid SAS uses both proactive and reactive adaptation. The reactive adaptation is used as a back-up for unpredicted occurrences [12]. The benefit of this combination can be seen in uncertain environments [28]. A limited of research is conducted considering this approach as it is complex and requires consistent decisions.

A Monitor, Analyse, Plan, and Execute over shared Knowledge (MAPE-K)-based resource provisioning framework is designed for IoT applications in a FC environment with the possibility to utilise the cloud layer [7]. It uses several statistical workload forecasting methods (e.g., Auto Regression Moving Average (ARMA) and ARIMA). An evaluation is performed to identify the most appropriate forecasting model. Further, the scaling decision of the fog nodes is made based on a Bayesian learning technique. On the reactive side, a threshold-based technique is used to make the cloud layer offloading decisions. The evaluation is conducted on iFogSim using two synthetic workloads (i.e., Smooth and bursty workloads) and New York city taxi trip real workload.

An elastic cloud platform for web applications based on Docker is designed in [29]. In terms of proactive adaptation, it uses second order ARMA for forecasting the number of requests. In terms of reactive adaptation, a resource utilisation threshold is used. The adaptation actions are adding/removing and starting/stopping the containers.

A cloud-assisted EC system architecture is proposed in [30]. It allows monitoring the EC resources by three main

 ISBN: 978-1-61208-854-9

strategies. The first strategy is an elasticity provision which is responsible for forecasting the workload using both ARIMA and neural network. Another strategy is a dynamic replica placement across both cloud and edge layers. The last strategy is the migration of the data from the cloud to the edge once the edge workload decreases with consideration to data reliability, migration time, and cost.

The SAS is not only related to elasticity problems but can also be used to manage the power and energy consumption. For instance, an energy-aware SAS for CC is presented in [31] to minimise the power and energy consumption where adaptation changes are implemented the in the VM level.

D. Related work limitations

Although, a considerable body of research has conducted using different SAS approaches considering either CC, EC, or both environments, several limitations and clear research gaps can be identified. These limitations are related to the hybrid SAS consideration, SAS design, the suitability of SAS to the targeted environment, and the adopted workload. In fact, these limitations become obvious when considering the EC environments.

The literature limitations can be summarized as follows. Firstly, a limited research effort has been conducted using the hybrid SAS which represents a clear research gap, especially for EC environments. Another limitation is the consideration of the theoretical perspective when designing and implementing the SAS using the adaptation approaches where the adaptation approaches performance may vary based on environment, application, and workload. This means a thorough investigation about the adaptation approaches considering the EC environment, IoT applications, and workload is an open research problem and not been tackled yet. Furthermore, the use of real EC workload in the SAS evaluation represents a significant weakness due to the use of either synthetic or real cloud workload to perform the SAS evaluation in EC environments. Therefore, this paper aims to address the mentioned limitations by designing an elasticity SAS framework considering the suitability of the adaptation approach to the EC environments. Further, this design is driven via thorough investigations on the performance of proactive, reactive, and hybrid adaptation in the EC environment considering two applications' scenarios (i.e., single and mixed) and real workload.

III. PROPOSED SELF-ADAPTIVE SYSTEM

This section presents the proposed elasticity SAS framework of the EC environments.

A. MAPE-based Elasticity SAS Framework

The proposed elasticity SAS framework is designed using a MAPE-based control loop. The use of this control loop allows the system to have a full and autonomic management of the resources over time in order to act in accordance to the workload variation. This means the system can instantiate/terminate containers as well as accept/reject requests in an autonomic fashion where each request represents an IoT task that needs to be executed by containerised application. Additionally, the proposed framework is designed to support three adaptation approaches which are proactive-, reactive-, and hybrid-adaptation. The SAS implementation details are provided in Section IV-A.

In this section, the proposed SAS is discussed according to each activity in the MAPE loop, which are monitor, analyse, plan, and execute. The SAS is shown in Figure 1 and divided into regions based on MAPE activities where each activity is highlighted using different colour. These colours are red for Monitor, yellow for Analyse, green for Plan, and blue for Execute. Further, the main contributions of this paper are highlighted in grey.

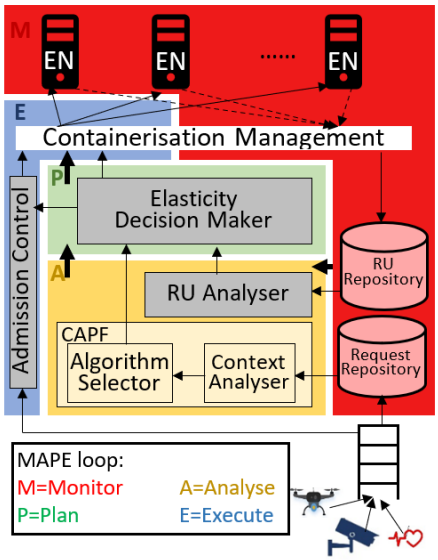


Figure 1. MAPE-based elasticity SAS framework.

1) ***Monitor:*** is responsible for collecting the data of the Edge Node (EN) resources and the IoT devices' workload. The EN resources include the number of containers which is used to instantiate/terminate containers by all adaptation approaches. This data is stored in the Resource Utilisation (RU) repository. The workload history of IoT devices is stored in the Request Repository and utilised to forecast the future IoT workload.

2) ***Analyse:*** is responsible for analysing the stored data by the monitor activity. It consists of two main components. The first component is the Context-aware Prediction Framework (CAPF) that represents the core of proactive adaptation. The CAPF is responsible for forecasting the future workload by utilising the workload history that is received from Request Repository and applying the most appropriate ML algorithm with consideration to the workload context. In this paper, this component is used as a black box as a full paper was published about CAPF in [13]. The second component is RU Analyser which is an important component for all adaptation approaches. In terms of proactive adaptation, it is responsible for analysing the run-time EN resources (i.e., number of idle containers) which can be used in the elasticity decision-making process. In case of the reactive adaptation, it is responsible for triggering the number of containers when it is below the threshold.

 ISBN: 978-1-61208-854-9

3) ***Plan****:* is responsible for making elasticity decision (i.e., instantiate/ terminate) where the action is taken by the Elasticity Decision Maker. In case of the proactive adaptation, this component uses both the forecasting results that are generated by CAPF and the RU analyser to make the elasticity decision. In case of the reactive adaptation, it only utilises the RU analyser results to make the elasticity decision. Lastly, in the hybrid adaptation, both CAPF and RU analyser are important in decision-making process.

4) ***Execute:*** this activity is responsible for performing the decisions that are made by the decision-maker. It has the admission control which is responsible for accepting/ rejecting the requests based on the resources' availability.

B. System Model and Assumptions

This section describes the adopted system model using the layered architecture. It consists of IoT-layer, edge-layer, and cloud-layer. The cloud layer is out of the paper's scope.

The IoT layer (i.e., bottom layer) consists of many IoT devices (e.g., smartphones) and connected to the upper layer (i.e., edge layer) via a 5G cellular network. The IoT devices send a set of requests $R = \{r_i \epsilon R \mid i = 1, 2, 3, \dots, N\}$ to the edge layer where each IoT device demands only one application type and eligible for sending more than one request. Further, each request is associated with application type a_i where a set of containerised applications are available $A = \{a_x \epsilon A \mid x = 1, 2, 3, \dots, 6\}$, RC_x, TL_x, and UP_x where these notations are defined in Table I. The specification of these requirements is made with respect to the implementation environment which will be discussed in Section IV-C.

On the other hand, the edge layer consists of a set of ENs, $EN = \{en_j \epsilon EN \mid j = 1, \dots, 4\}$ that are located in one cluster (i.e., a group of ENs) and connected to a centralised orchestrator that is the brain of the edge layer and hosts the proposed elasticity SAS framework. Further, we assume that the ENs are homogenous virtualised environment that hosts containerised applications A in a bare-metal manner. Further, each en_j is associated with the following capabilities: CPU_j and PS_j, which are defined in Table I.

In case of a request r_i is accepted, the request maps to a container instance based on the application type a_i. Once the request is processed, the results are sent back to the IoT device associated with the DW_i that is defined in Table I.

TABLE I. NOTATIONS

Symbol	Definition
EN	A set of edge nodes at the edge layer, where $en_j \in EN$
CPU_j	# of CPU cores for each en_j
PS_j	Processing speed of en_j
A	A set of applications requested by IoT, where $a_i \in A$
R	A set of requests generated by IoT, where $r_i \in R$
$RC_{i,x}$	Required CPU of r_i
$TL_{i,x}$	Task length of r_i
$UP_{i,x}$	Uploading data size of r_i
$DW_{i,x}$	Downloading data size of r_i
y	# of containers that needs to be instantiated/terminated

C. Proposed Algorithms

There are four algorithms that are developed to support the proposed framework. These algorithms are explained next.

1) Reactive algorithm (see Figure 2): it is a threshold-based algorithm that is responsible for reactively triggering the number of utilised containers and make instantiation decision. This algorithm is used in both reactive adaptation and hybrid adaptation. Once a request is accepted (line 1), it checks the number of stand-by (i.e., up and ready) containers. If the number of stand-by containers is below the threshold, it makes an instantiation decision (line 2 and 3).

Input: # of stand-by containers for each App. $Cont_{SB}^{a_x}$ and Minimum # of containers $Cont_{Min}$
Output: Elasticity decision (instantiate) by y

0: Begin
1: For after each accepted request of a_i **do**
2: **If** ($Cont_{SB}^{a_x} < Cont_{Min}$)
3: Instantiate by y from a_x
4: **End if**
5: End for
6: End

Figure 2. Reactive algorithm (Algorithm 1).

Input: Predicted value P, # of stand-by containers for each App. $Cont_{SB}^{a_x}$, maximum number of allowed containers $Cont_{Max}$, and # of App. A_{len}.
Output: Elasticity decision (instantiate/ terminate) by y

0: Begin
1: For each time interval **do**
2: **Compute** $P^{a_x} \leftarrow \left\lceil \frac{P}{A_{len}} \right\rceil$
3: **If** ($P^{a_x} > Cont_{Max}$)
4: **Set** $P^{a_x} \leftarrow Cont_{Max}$
5: **End if**
6: **For** each application a_i **do**
7: **If** ($P^{a_x} == Cont_{SB}^{a_i}$)
8: No decision
9: **Else if** ($P^{a_x} < Cont_{SB}^{a_i}$)
10: Terminate by $y \leftarrow (Cont_{SB}^{a_x} - P^{a_x})$
11: **Else**
12: Instantiate by $y \leftarrow (P^{a_x} - Cont_{SB}^{a_x})$
13: **End if**
14: **End for**
15: End for
16: End

Figure 3. Proactive algorithm (Algorithm 2).

2) Proactive algorithm (see Figure 3): it is responsible for instantiating and terminating containers proactively using the CAPF outputs and the number of stand-by containers. First, for each time interval, it computes the predicted value for each application type (line 2). Then, it compares this value for each application type with the maximum number of allowed containers in the edge (line 3). If the predicted value is greater than the maximum number of containers, it considers the maximum (line 4). Then, for each application, it calculates the required number of containers by comparing the stand-by containers with the predicted value (line 6- 13).

Based on this calculation the decision is made either no decision, instantiate, and terminate.

3) Hybrid algorithm (see Figure 4): it brings both proactive and reactive algorithms together in the same SAS. The proactive algorithm runs at equal time intervals whereas the reactive is continuously running.

4) Admission control algorithm (see Figure 5): it is responsible for accepting/rejecting the requests based on the container's availability. Once a request is received, it checks if there is any stand-by container (line 2-6). If there is a stand-by container from the same application category, the request will be accepted and executed on the targeted containerised application. Otherwise, the request will be rejected.

Input: # of stand-by containers for each App. $Cont_{SB}^{a_x}$ and Minimum # of containers$Cont_{Min}$, predicted value P, maximum number of allowed containers $Cont_{Max}$, and # of App. A_{len}
Output: Elasticity decision (instantiate/ terminate) by y

0: Begin
1: While true **do**
2: **For** each time interval **do**
3: **Call** proactive algorithm
4: **End for**
5: **Call** reactive algorithm
6: End while
7: End

Figure 4. Hybrid algorithm (Algorithm 3).

Input: # of stand-by containers for each App. $Cont_{SB}^{a_i}$ and the App. type of received request $r_{i,x}$.
Output: Accept or reject decision

0: Begin
1: While true **do**
2: **If** ($Cont_{SB}^{a_i} > 0$)
3: Accept $r_{i,x}$
4: **Else**
5: Reject $r_{i,x}$
6: **End if**
7: End while
8: End

Figure 5. Admission control algorithm (Algorithm 4).

D. Applications Profiling

This section describes the selection of the adopted applications. In fact, selecting the application is a critical decision that must be taken carefully as it plays a major role in any resource management research. Therefore, we select a set of applications that are different in terms of *latency requirements* where the EC paradigm is mainly emerged to support latency-sensitive applications and the *resource requirements* where the EC has limited resources by nature. In other words, both latency- and resource-requirement are important concepts for EC environments as it emerges to support latency-sensitive applications as well as often has limited resources which represents a bottleneck for IoT applications. For these reasons, we classify the applications into latency-sensitive, medium-latency, and latency-tolerant applications. Similarly, the application requirements are classified into high, medium, and low requirements. Thus, the requirements are specified using ranges as these requirements do not exist in the literature with consideration to the implementation environment. For example, the required number of CPU cores ranges between 1-4 cores. Similarly, the required processing speed and servers' utilisation ranges between 500-2000 MIPS and 5%-20%, respectively. In case of the uploading and downloading data size, the assumptions are made according to the scenarios below as limited information is available in the literature. In short, we select 6 different applications, two applications from each category. This variety of both applications and their requirements is critical as it shows the effectiveness of the proposed framework and represents a realistic scenario. The main configuration parameters of adopted applications are shown in Table II. They are set to be as realistic as possible based on the real scenarios below:

TABLE II. APPLICATIONS CONFIGURATIONS

App.	CPU cores	Avg. processing speed (MIPS)	Server's utilisation (%)	Avg. upload size	Avg. download size (KB)
FR	4	2000	20	450 MB	5
ETM	4	2000	20	200 MB	5
AR	2	1000	10	1 MB	50
HM	2	1000	10	200 KB	5
IHM	1	500	5	200 KB	20
IP	1	500	5	4 KB	4

1) Face Recognition (FR): Public safety domain includes a wide range of applications, such as FR and Cars' plates identification [32][33][34]. This kind of applications is important as it helps the authority to track people, find a missing person, and track cars. It can be implemented by consuming video surveillance resources. In the context of this paper, we assume that FR application is used to find or track a person in any incident, thus, it is considered as a latency-sensitive application in the sense that the video frames need to be analysed quickly. Further, due to the type of data (video frames), the FR is categorised as a data-intensive and computational-intensive application [32][35][36].

2) Emergency Traffic Management (ETM): Nowadays, traffic flow management systems are important applications that help improving traffic efficiency, reduce accidents, support emergency services, and manage traffic jams [37]. In this paper, ETM application focuses on supporting emergency services where the application can be requested by emergency services (e.g., ambulance and fire trucks) to perform better traffic management. The ETM can be classified as a latency-sensitive application as it is related to emergency cases. It is also considered data-intensive and computational-intensive in some use cases [38]. However, in our scenario, we assume that the data size is medium as it may require gathering data from different sources (e.g., infrastructure, vehicles, and pedestrians). At the same time, ETM is computational-intensive. Based on this scenario, the application configuration is set as shown in Table II.

3) Augmented Reality (AR): AR is a well-known application that can be used in many domains, such as healthcare, agriculture, and tourism [40][41][42][43]. For the purpose of this research, we assume that it is used to guide the people during their mobility in a city by displaying contextual information about the objects/ directions by analysing the captured figures (i.e., frames). We also assume that it requires medium latency as it is not an urgent application when compared to TFM and FR. It also requires a medium-requirments of resources.

4) Health Monitoring (HM): It is a common application that benefits from EC. In general, it is classified as data-sensitive and latency-sensitive application [41][44][45][46]. However, we assume that the real-time and emergency data analysis is performed locally on the patients' smartphones or wearable devices whereas the edge layer is utilised to perform a future health prediction in a form of requests. Then, the alarm will be sent to the hospital in case of any incident is predicted. This assumption makes this application requires medium latency and resources.

5) Industrial Health Monitoring (IHM): IHM application covers a wide number of scenarios. In this research, we adopted the same scenario that is presented in [3], where the data is collected and sent to the edge for analysis and visualisation. In other words, the application is used to monitor the workers' health and environment. Further, IHM is latency-tolerant and low-requirements application.

6) Intelligent Parking (IP): IP is a smart city application which is used to search for parking slots [37]. It is assumed that the driver submitted a search request to the edge layer which is responsible for retrieving the relevant data about parking slots and suggestions to the user as a list of nearest parking spaces. Based on this scenario, IP is considered as a latency-tolerant and low-requirements application.

IV. EXPERIMENTAL DESIGN

This section presents the experimental design, which includes implementation scenarios, dataset, workload, evaluation metrics, and hypothesis.

A. Implementation Scenarios

The proposed elasticity SAS is implemented considering the proactive, reactive, and hybrid adaptation, separately. For instance, in the implementation of proactive adaptation, the reactive adaptation components are deactivated.

To evaluate the effectiveness of these approaches two application-based scenarios are considered as summarised in Table III. The experiments are conducted considering *1) mixed applications*, *2) single application*. Further, in each scenario, the adaptation approaches are evaluated using three real workloads which will be discussed in the next section. In the single application scenario, the adaptation approaches are evaluated considering one application in each experiment. This means each adaptation approach is evaluated using FR application as a heavy load, AR application as a medium load, and IHM application as a low load. The consideration of these scenarios is important to design the most suitable SAS for EC environment considering the adaptation approaches, workload, and applications.

TABLE III. IMPLEMENTATION SCENARIOS

Scenarios		Adaptation approaches	Considered applications
1: Mixed applications		Proactive	All
		Reactive	All
		Hybrid	All
2: Single application	A	Proactive	FR
		Reactive	FR
		Hybrid	FR
	B	Proactive	AR
		Reactive	AR
		Hybrid	AR
	C	Proactive	IHM
		Reactive	IHM
		Hybrid	IHM

B. Workload

This paper adopts the Shanghai Telecom dataset [46] which is ideal for the consideration of IoT workload as previously used by [48][49][50][51]. It provides six months of mobile phones records accessing the Internet and connecting to base stations that are distributed over Shanghai city.

The same workload pattern will be used as in our previous work [13]. In [13], the workload is divided into three patterns, which are decreasing, increasing, and fluctuating. Each pattern consists of a set of hours as each pattern represents a part of the day (i.e., late night and early morning as a decreasing pattern, morning as an increasing pattern, and afternoon to evening as a fluctuating pattern). Further, one hour from each pattern is selected (2nd hour from decreasing, 12th hour from increasing, and 14th hour from fluctuating) to be used in training/ testing the proposed forecasting models.

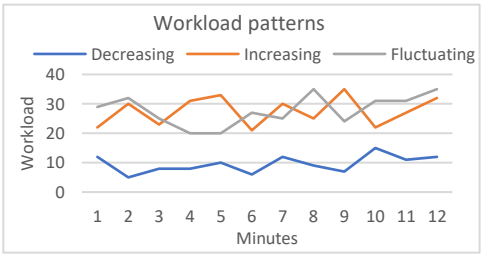


Figure 6. Workload patterns.

The testing part is used to evaluate the proposed SAS. It represents the last 12 minutes from each pattern named as decreasing, increasing, and fluctuating workload, which will be fed to the simulation environment. The workload is shown in Figure 6 over time interval. The use of 12 minutes is considered long enough to evaluate the SAS including the adaptation approaches and algorithms. Note that the workload in Figure 6 does not show the overall pattern (i.e., decreasing, increasing, or fluctuating) as it is a snip from the dataset by zooming in towards the adopted time frame by this paper. The patterns are named according to the previous work [13] to ensure consistency.

 ISBN: 978-1-61208-854-9

C. Simulation Setup

The proposed elasticity SAS framework is implemented using the EdgeCloudSim simulator [51], which is built upon the CloudSim simulator. It allows simulating the EC environment with consideration to the IoT-, edge-, and cloud-layers. It also can simulate different scenarios with/without cloud consideration and edge orchestrator. In this paper, two layers only are considered, the IoT- and edge-layers (the cloud layer out of our scope). The simulation duration is 14 minutes; the 1st minute is considered as a warm-up period and the last minute is waiting time to allow all tasks to be completed. The 12 minutes in-between is the real workload that is fed to the simulator. Further, six applications are considered with different requirements to evaluate the proposed SAS. Also, considering four ENs is deemed sufficient to allow performing the evaluation process. In terms of the number of IoT devices and requests, these values are specified for each workload pattern according to the number of devices and requests in the dataset. The most important simulation parameters are shown in Table IV.

TABLE IV. SIMULATION CONFIGURATION

Parameter	Value
Simulation time (min.)	14
Warm-up period (min.)	1
# of iterations	5
IoT Applications (mixed/ Single)	(6/1)
# of IoT devices (decreasing/ increasing/ fluctuating)	(108/277/271)
# of IoT requests (decreasing/ increasing/ fluctuating)	(115/331/334)
# of edge nodes	4
# of cores/edge node	4
Processing speed/edge node (MIPS)	2000
Resource check interval (sec.)	15

D. Evaluation Metrics

Two evaluation metrics are used. They are the *acceptance rate* and *servers' utilisation*. The acceptance rate evaluates the effectiveness of each adaptation approach when dealing with dynamic workload. On the other hand, the servers' utilisation refers to the CPU utilisation over the time intervals.

E. Hypothesis

Two hypotheses are considered to evaluate the effectiveness of adaptation approaches in the proposed elasticity SAS framework:

Hypothesis 1: *The use of the hybrid adaptation in an elasticity framework will provide the highest acceptance rate as compared to both proactive and reactive adaptations.*

Hypothesis 2: *The proactive adaptation will perform better than the reactive adaptation due to the prediction ability that helps acting prior (i.e., in advance) events happen.*

V. PERFORMANCE EVALUATION

This section evaluates and discusses the results as well as highlights the main findings.

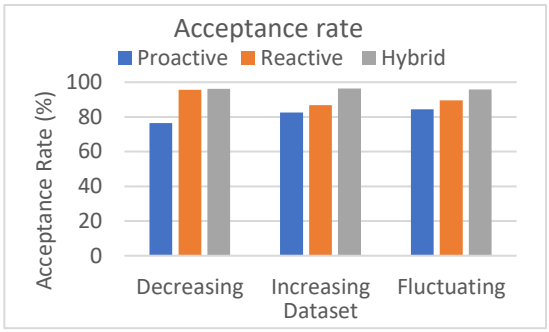
A. Adaptation Approaches Evaluation

The adaptation approaches are compared with respect to the stated scenarios in Section IV-A.

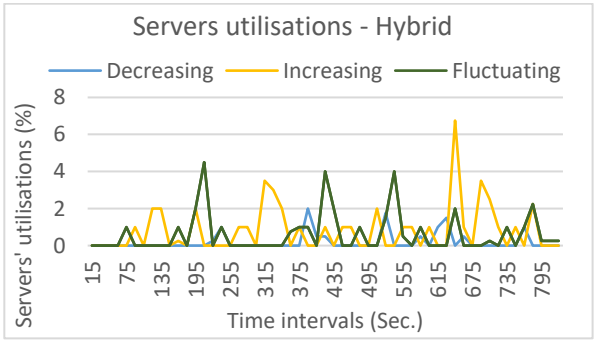
Scenario 1- mixed applications: it evaluates the adaptation approaches using all applications over decreasing, increasing, and fluctuating patterns.

Scenario 2- single application: it evaluates the adaptation approaches with respect to the application category (i.e., heavy-, medium-, and low-load).

1) Scenario 1 (All Apps.): the hybrid adaptation provides the highest acceptance rate overall pattern when compared to other adaptation approaches as shown in Figure 7.a. It performs about 10% higher than reactive adaptation for the increasing pattern and 7% for fluctuating pattern. It also performs about 20%, 14%, and 12% higher than proactive adaptation in decreasing, increasing, and fluctuating patterns, respectively. The hybrid adaptation superiority is due to its ability to trigger unpredicted requests thanks to the reactive adaptation side. The high acceptance rate leads to efficient utilisation of the ENs, see Figure 7.b, which shows the servers' utilisations over time for the hybrid adaptation as it has the highest acceptance rate and utilisation.



(a)

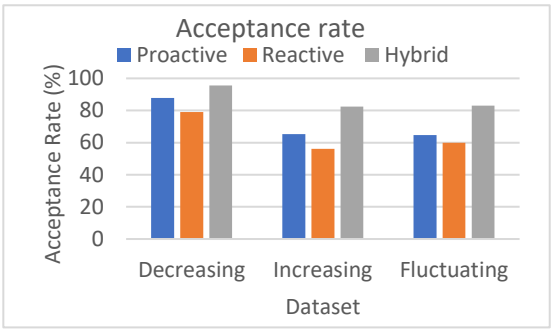


(b)

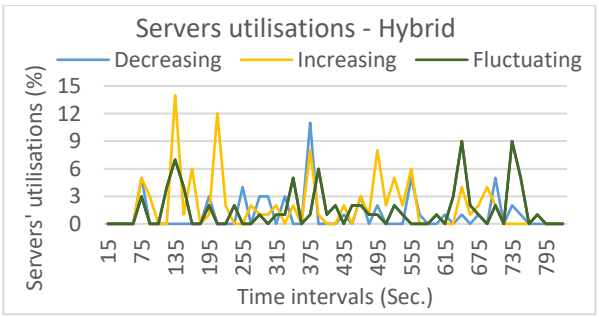
Figure 7. Scenario 1.

2) Scenario 2.A (FR): in this scenario, the hybrid adaptation also outperforms both proactive and reactive adaptation overall patterns as shown in Figure 8.a thanks to the consideration of both proactive and reactive adaptations where the reactive adaptation side can deal with unpredicted events. However, the proactive adaptation outperforms the reactive adaptation overall patterns. This due to the use of single application scenario. This means that all the submitted requests will be from the same type of application whereas in scenario 1 the predicted value will be divided over the

number of considered applications assuming that all these applications come on the same probability. In terms of servers' utilisation, the FR application is considered as a heavy-load application. This means a higher workload is expected as compared to Scenario 1 which considers all applications. Figure 8.b shows the servers' utilisation over thensimulation time for the hybrid adpatation which can reach about 15% in this scenario as a heavy-load application is considered.



(a)



(b)

Figure 8. Scenario 2A.

3) Scenario 2.B (AR): in this scenario, the results are similar to Scenario 2.A where the hybrid adaptation outperforms all the adaptation approaches over different patterns as shown in Figure 9. The proactive adaptation also outperforms the reactive adaptation. The main difference is the average server utilisation which is lower than the avergae servers utilisation in the FR scenario where the AR application is considered as medium-load.

4) Scenario 2.C (IHM): the acceptance rate of this scenario is similar to Scenario 2.B which is not presented due to space limitation. In terms of the servers' utilisation, it is the lowest as compared to all previous experiments in the sense that it considers applications with low-load.

B. Hypothesis Evaluation

This section tests the hypotheses based on the considered scenarios.

1) ***Hypothesis 1:*** *The use of the hybrid adaptation in an elasticity framework will provide the highest acceptance rate as compared to both proactive and reactive adaptations.* This holds true in all scenarios. The hybrid adaptation shows a great performance as compared to both proactive- and reactive-adaptation. This is due to the consideration of the proactive adaptation to prepare the containers prior receiving the requests as well as the use of threshold-based in the reactive adaptation to maintain the number of stand-by containers.

2) ***Hypothesis 2:*** *The proactive adaptation will perform better than reactive adaptation due to prediction ability that helps acting prior (i.e., in advance) events happen.* This hypothesis is disproved for the mixed scenario (i.e., Scenario 1), while correct for the single scenarios (i.e., Scenarios 2A, 2B, and 2C). In terms of the mixed scenario, the predicted workload by the CAPF is divided by the number of applications and assuming that all applications have the same arrival probability. This means the CAPF predicts the overall workload without any consideration to the applications' arrival probability. This assumption is made as there is no previous information available in the real dataset about the type of applications that will be requested. In contrast, the proactive adaption outperforms the reactive in single scenarios as the predicted workload will be utilised by the same application.

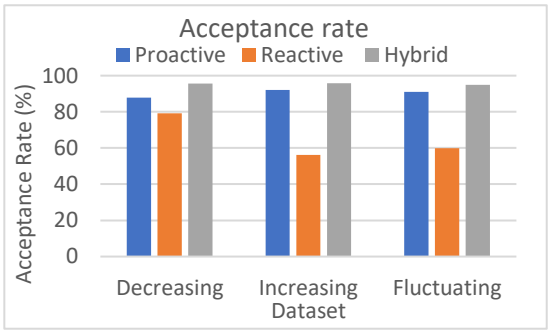


Figure 9. Scenario 2B.

C. Findings and Recommendations

The main findings of this paper can be summarised as follows with some recommendations:

1) Although the hybrid adaptation is complex and requires bringing both proactive and reactive adaptation together in a consistent manner, it provides the best performance over different scenarios and workload patterns and has the ability to adapt in a highly fluctuating environment. According to this finding, the hybrid SAS is recommended to be used in highly fluctuating environments, such as EC, as it provides a full monitoring loop with the ability to deal with unpredicted events. In other words, using either proactive or reactive adaptation approaches in a highly fluctuating environment may lead to low performance as there is a need to anticipate the future behaviour as well as using reactive adaptation for backup. The use of hybrid adaptation is also important even when the prediction models show great accuracy.

 ISBN: 978-1-61208-854-9

2) The hybrid adaptation efficiently utilises the EC resources as it is able to accept more requests and contributes to avoid over/under-prevision cases thanks to the use of the reactive adaptation as a back-up for the proactive adaptation to deal with the unpredicted workload. Its efficiency can be seen clearly in Scenario 1 when there is no previous knowledge about the request types. Thus, the hybrid adaptation is recommended in EC environments as these have limited resources by nature.

3) The available information about submitted requests to the edge layer plays an important role in designing the elasticity SAS framework. In fact, the proactive adaptation is preferable as compared to the reactive adaptation as it acts prior the event occurrence and prepares the resources in advance. However, in case limited information about the predicted events is available, this may lead to low performance as compared to the reactive adaptation as well as unpredicted results. This can be seen in scenario 1 when using mixed applications where the predicted value represents the overall workload without the consideration of the arrival probability for each application type.

4) It is important to evaluate the performance of the adaptation approaches in the implementation domain as their performance may vary according to the scenario and workload.

VI. Conclusion and Future Work

This paper has presented and evaluated an elasticity SAS which is supported by proactive, reactive, hybrid, and admission control approaches as well as various application scenarios. The experiment results show that the most appropriate adaptation approach in an EC environment is the hybrid where its performance is at least 10% better than other approaches. The results also reveal that the performance of the adaptation approaches is domain, application, and scenario dependent.

As future work, the proposed SAS will be evaluated using different workloads aiming to stress the SAS with higher request rates. In fact, the use of higher request rate to evaluate the proposed framework is important as some experiments show small servers utilization. Additionally, both the scalability and QoS will be considered with the aim to maximise the number of running applications with adequate QoS. Moreover, a policy management will be investigated to identify the trade-off between the service acceptance maximisation from the perspective of the service provider and the QoS from the consumer, respectively.

References

[1] P. Parwekar, “From Internet of Things towards cloud of things”, The 2nd International Conference on Computer and Communication Technology, 2011, pp. 329–333.

[2] I. Yaqoob et al., “Internet of Things Architecture: Recent Advances, Taxonomy, Requirements, and Open Challenges”, IEEE Wirel. Commun., vol. 24, no. 3, pp. 10–16, 2017.

[3] F. Wu, T. Wu, and M. R. Yuce, “An Internet-of-Things (IoT) Network System for Connected Safety and Health Monitoring Applications”, Sensors (Switzerland), vol. 19, no. 1, pp. 1–21 2018.

[4] J. Cheng, W. Chen, F. Tao, and C. L. Lin, “Industrial IoT in 5G environment towards smart manufacturing”, Journal of Industrial Information Integration, vol. 10 , pp. 10–19, 2018.

[5] S. Taherizadeh, A. C. Jones, I. Taylor, Z. Zhao, and V. Stankovski, “Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review”, Journal Systems and Software, vol. 136, pp. 19–38, 2018.

[6] G. Li, J. Song, J. Wu, and J. Wang, “Method of Resource Estimation Based on QoS in Edge Computing”, Wirel. Commun. and Mob. Comput., vol. 2018, pp. 1–9 ,2018.

[7] M. Etemadi, M. Ghobaei-Arani, and A. Shahidinejad, “Resource provisioning for IoT services in the fog computing environment: An autonomic approach”, Comput. Commun., vol. 161, pp. 109–131, 2020.

[8] B. Liu, J. Guo, C. Li, and Y. Luo, “Workload forecasting based elastic resource management in edge cloud”, Comput. & Ind. Eng., vol. 139, pp. 1–12, 2020.

[9] R. Kavanagh, K. Djemame, J. Ejarque, R. M. Badia, and D. Garcia-Perez, “Energy-aware Self-Adaptation for Application Execution on Heterogeneous Parallel Architectures”, IEEE Trans. Sustain. Comput., vol. 5, no. 1, pp. 81–94, 2019.

[10] M. D’Angelo, “Decentralized Self-Adaptive Computing at the Edge”, IEEE/ACM The 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, 2018, pp. 144–148.

[11] M. Xu and R. Buyya, “Brownout Approach for Adaptive Management of Resources and Applications in Cloud Computing Systems: A Taxonomy and Future Directions”, ACM Comput. Surv., vol. 52, no. 1, pp. 1–27, 2019.

[12] C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele, and C. Becker, “A survey on engineering approaches for self-adaptive systems”, Pervasive Mob. Comput., vol. 17, pp. 184–206, 2015.

[13] A. F. Aljulayfi and K. Djemame, “A Machine Learning Based Context-aware Prediction Framework for Edge Computing Environments”, The 11th International Conference on Cloud Computing and Services Science, 2021, pp. 143–150.

[14] G. Galante and L. C. E. de Bona, “A Survey on Cloud Computing Elasticity”, IEEE/ACM The Fifth International Conference on Utility and Cloud Computing, 2012, pp. 263–270.

[15] M. Amiri and L. Mohammad-Khanli, “Survey on prediction models of applications for resources provisioning in cloud”, Jou. Netw. Comput. Appl., vol. 82, pp. 93–113, 2017.

[16] R. Moreno-vozmediano, R. S. Montero, E. Huedo, and I. M. Llorente, “Efficient resource provisioning for elastic Cloud services based on machine learning techniques”, Jou. Cloud Comput. Adv. Syst. and Appl., vol. 8, no. 1, pp. 1–18, 2019.

[17] D. F. Kirchoff, M. Xavier, J. Mastella, and C. A. F. De Rose, “A preliminary study of machine learning workload prediction techniques for cloud applications”, The 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 2019, pp. 222–227.

[18] S. Ajila and A. A. Bankole, “Cloud Client Prediction Models Using Machine Learning Techniques”, in International Computer Software and Applications Conference, 2013, pp. 134–142.

[19] D. Spatharakis et al., “A scalable Edge Computing architecture enabling smart offloading for Location Based Services”, Pervasive Mob. Comput., vol. 67, pp. 1–15, 2020.

[20] M. Aldossary, I. Alzamil, and K. Djemame, “Towards Virtual Machine Energy-Aware Cost Prediction in Clouds”, in GECON: International Conference on the Economics of Grids, Clouds, Systems, and Services, 2017, pp. 285–299.

 ISBN: 978-1-61208-854-9

[21] M. Aldossary and K. Djemame, “Performance and Energy-based Cost Prediction of Virtual Machines Auto-scaling in Clouds”, The 44th Euromicro Conference on Software Engineering and Advanced Applications, 2018, pp. 502–509.

[22] M. Abdullah, W. Iqbal, A. Mahmood, F. Bukhari, and A. Erradi, “Predictive Autoscaling of Microservices Hosted in Fog Microdata Center”, IEEE Syst. Jou., pp. 1–12, 2020.

[23] A. V. Chandak and N. K. Ray, “Multi Agent Based Resource Provisioning in Fog Computing”, Trends Comput. Intell. Secur. Internet Things. Commun. Comput. Inf. Sci., vol. 1358, pp. 317–327, 2020.

[24] K. Kaur, T. Dhand, N. Kumar, and S. Zeadally, “Container-as-a-Service at the Edge: Trade-off between Energy Efficiency and Service Availability at Fog Nano Data Centers”, IEEE Wirel. Commun., vol. 24, no. 3, pp. 48–56, 2017.

[25] T. Yang, Y. Hu, M. C. Gursoy, A. Schmeink, and R. Mathar, “Deep Reinforcement Learning based Resource Allocation in Low Latency Edge Computing Networks”, The 15th International Symposium on Wireless Communication Systems, 2018, pp. 1–5.

[26] S. Mireslami, L. Rakai, M. Wang, and B. H. Far, “Dynamic Cloud Resource Allocation Considering Demand Uncertainty”, IEEE Trans. Cloud Comput., pp. 1–14, 2019.

[27] E. Ataie, R. Entezari-Maleki, S. E. Etesami, B. Egger, D. Ardagna, and A. Movaghar, “Power-aware performance analysis of self-adaptive resource management in IaaS clouds”, Futur. Gener. Comput. Syst., vol. 86, pp. 134–144, 2018.

[28] G. Anders, F. Siefert, M. Mair, and W. Reif, “Proactive Guidance for Dynamic and Cooperative Resource Allocation under Uncertainties”, in International Conference on Self-Adaptive and Self-Organizing Systems, 2014, pp. 21–30.

[29] C. Kan, “DoCloud: An Elastic Cloud Platform for Web Applications Based on Docker”, The 18th International Conference on Advanced Communication Technology, 2016, pp. 478–483.

[30] C. Li, J. Bai, Y. Ge, and Y. Luo, “Heterogeneity-aware elastic provisioning in cloud-assisted edge computing systems”, Futur. Gener. Comput. Syst., vol. 112, pp. 1106–1121, 2020.

[31] K. Djemame et al., “PaaS-IaaS Inter-Layer Adaptation in an Energy-Aware Cloud Environment”, IEEE Trans. Sustain. Comput., vol. 2, no. 2, pp. 127–139, 2017.

[32] L. Soh, J. Burke, and L. Zhang, “Supporting Augmented Reality: Looking Beyond Performance”, Proc, of The 2018 Morning Workshop on Virtual Reality and Augmented Reality Network, 2018, pp. 7–12.

[33] F. Faticanti, F. D. Pellegrini, D. Siracusa, D. Santoro, and S. Cretti, “Cutting Throughput with the Edge: App-Aware Placement in Fog Computing”, The 6th IEEE International Conference on Cyber Security and Cloud Computing, and The 5th IEEE International Conference on Edge Computing and Scalable Cloud, 2019, pp. 196–203.

[34] E. Yigitoglu, M. Mohamed, L. Liu, and H. Ludwig, “Foggy: A Framework for Continuous Automated IoT Application Deployment in Fog Computing”, Proc. IEEE The 6th International Conference on AI and Mobile Services, 2017, pp. 38–45.

[35] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, “iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments”, Softw.: Pract. and Exp., vol. 47, no. 9, pp. 1275–1296, 2017.

[36] S. Yi, C. Li, and Q. Li, “A Survey of Fog Computing: Concepts, Applications and Issues”, Proc. of The 2015 Workshop on Mobile Big Data, 2015, pp. 37–42.

[37] A. Shahzad, J. Choi, N. Xiong, Y. Kim, and M. Lee, “Centralized Connectivity for Multiwireless Edge Computing and Cellular Platform: A Smart Vehicle Parking System”, Wirel. Commun. and Mob. Comput., vol. 2018, pp.1–23, 2018.

[38] S. Kekki et al., “MEC in 5G networks”, ETSI white Paper No. 28, pp. 1–28, 2018.

[39] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, “Towards Wearable Cognitive Assistance”, in Proc. of The 12th annual international conference on Mobile systems, applications, and, services, 2014, pp. 68–81.

[40] G. White, C. Cabrera, A. Palade, and S. Clarke, “Augmented Reality in IoT”, In: Liu X. et al. (eds) Service-Oriented Computing- ICSOC 2018 Workshops, Lecture Notes in Computer Science, vol. 11434, Springer, Cham , 2019, pp. 149–160.

[41] J. Dolezal, Z. Becvar, and T. Zeman, “Performance Evaluation of Computation Offloading from Mobile Device to the Edge of Mobile Network”, in IEEE Conference on Standards for Communications and Networking, 2016, pp. 1–7.

[42] J. Grubert, T. Langlotz, S. Zollmann, and H. Regenbrecht, “Towards Pervasive Augmented Reality: Context-Awareness in Augmented Reality”, IEEE Trans. Vis. Comput. Graph., vol. 23, no. 6, pp. 1706–1724, 2017.

[43] G. Rahman and C. W. Chuah, “Fog Computing, Applications, Security and Challenges, Review”, Int. Jou. Eng. Technol., vol. 7, no. 3, pp. 1615–1621, 2018.

[44] 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 Commun. Surv. Tutorials, vol. 19, no. 4, pp. 2359–2391, 2017.

[45] T. N. Gia, M. Jiang, A. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, “Fog Computing in Healthcare Internet of Things: A Case Study on ECG Feature Extraction”, Proc. IEEE International Conference on Computer and Information Technology, Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, 2015, pp. 356–363.

[46] Sguangwang.com, “The Telecom Dataset”, 2018. [Online]. Available from: http://sguangwang.com/TelecomDataset.html. [Retrieved: 2021.5.27].

[47] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C. Hsu, “Edge server placement in mobile edge computing”, Jou. Parallel and Distrib. Comput., vol. 127, pp. 160–168, 2019.

[48] S. Wang, Y. Zhao, L. Huang, J. Xu, and C. Hsu, “QoS prediction for service recommendations in mobile edge computing”, J. Parallel Distrib. Comput., vol. 127, pp. 134–144, 2019.

[49] Y. Guo, S. Wang, A. Zhou, J. Xu, J. Yuan, and C. Hsu, “User allocation-aware edge cloud placement in mobile edge computing”, Softw.: Pract. and Exp., vol. 50, no. 5, pp. 489–502, 2019.

[50] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, “Delay-aware Microservice Coordination in Mobile Edge Computing: A Reinforcement Learning Approach”, IEEE Trans. Mob. Comput., vol.20, no. 3 pp. 939–951, 2021.

[51] C. Sonmez, A. Ozgovde, and C. Ersoy, “EdgeCloudSim: An environment for performance evaluation of edge computing systems”, Trans. Emerg. Telecommun. Technol., vol. 29, no. 11, pp. 1–17, 2018.

 ISBN: 978-1-61208-854-9