Fuzzy Workload Orchestration for Edge Computing

Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy, *Senior Member, IEEE*

***Abstract*—Edge computing is based on the philosophy that the data should be processed within the locality of its source. Edge computing is entering a new phase where it gains wide acceptance from both academia and the industry as the commercial deployments are starting. Edge of the network presents a very dynamic environment with many devices, intermittent traffic, high mobility of the end user, heterogeneous applications and their requirements. In this scene, scalable and efficient management and orchestration remains to be a problem. We focus on the workload orchestration problem in which execution locations for incoming tasks from mobile devices are decided within an edge computing infrastructure, including the global cloud as well. Workload orchestration is an intrinsically hard, online problem. We employ a fuzzy logic-based approach to solve this problem by capturing the intuition of a real-world administrator to get an automated management system. Our approach takes into consideration the properties of the offloaded task as well as the current state of the computational and networking resources. Detailed set of experiments are designed with EdgeCloudSim to demonstrate the competitive performance of our approach for different service classes.**

***Index Terms*—Edge computing, orchestration, edge orchestrator, fuzzy logic, performance evaluation.**

I. Introduction

WE ARE witnessing a tremendous change in the end user side that generates the traffic for today's networks. Proliferation of the commercially available mobile gadgets, sensors, wearable units rendered many novel applications feasible involving IoT, smart home/city, autonomous vehicles, augmented reality, instant video processing, tactile Internet, e-health and Industry 4.0. These applications dictate a totally different modality of workflow when compared with traditional client-server applications served over a cloud computing infrastructure: they create huge amounts of data, require prompt response (real-time or semi real-time), involve high user mobility and diverse spectrum of requirements.

Although, this new category of applications are demanding in terms of the expectation of its users, the equipments employed for realizing them is getting ever smaller resulting in stringent battery and computational limits. This in essence presents a new challenge where complex software is being implemented on constrained hardware. Edge computing addresses this challenge by boosting the end user device via Computational Offloading mechanisms [1] through the use of micro-cloud servers located in the vicinity.

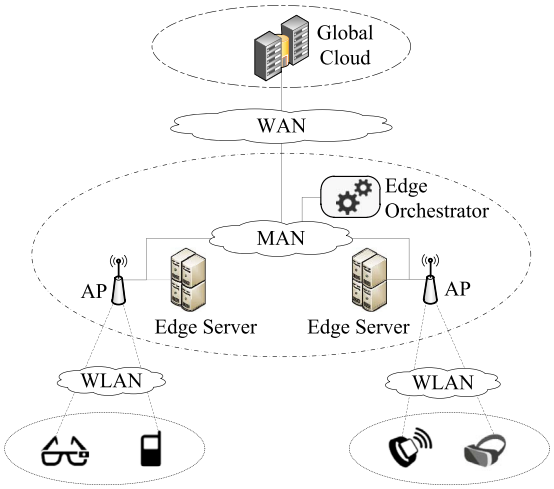


Fig. 1. Multi-tier edge computing architecture with edge orchestrator.

Edge computing is actually an umbrella term that signifies a family of relevant technologies such as Cloudlet [2], Mobile Cloud Computing (MCC) [3], Fog Computing [4] and Multi-Access Edge Computing (MEC) [5]. All these approaches are based on the sound engineering principle that computational resources are better be located on the edge where the real demand is physically present. By doing so (i) end user devices can get fast access to local computational resources using the local area network (LAN) (ii) the core network is not kept under heavy traffic load (especially critical for 5G), i.e., end user devices' data can be processed or filtered before sending to the cloud.

Edge computing infrastructure involves a series of edge servers located in a metropolitan area, connected with high speed network and managed from a single administrative domain. Cloud computing resources are also a natural part of this infrastructure aiding to serve whenever edge resources are insufficient. We depict this multi-tier view of the system in Fig. 1. In this picture, offloaded tasks from the end users can be served on any edge server in the domain based on the availability of the service requested and the spare capacity of the servers.

Envisioned edge computing scenarios constitute a very dynamic overall operation being executed on a fairly elastic

Manuscript received April 20, 2018; revised August 13, 2018, November 6, 2018, and January 21, 2019; accepted February 18, 2019. Date of publication February 25, 2019; date of current version June 10, 2019. This work is supported by the Galatasaray University Research Foundation under the Grant No. 18.401.003, the Bogazici University Research Fund under the grant number 12663 and the Republic of Turkey Ministry of Development, under the TAM project with the grant number 2007K120610. The associate editor coordinating the review of this paper and approving it for publication was X. Fu. *(Corresponding author: Cagatay Sonmez.)*

C. Sonmez and C. Ersoy are with NETLAB, Computer Networks Research Laboratory, Department of Computer Engineering, Bogazici University, Istanbul 34342, Turkey (e-mail: cagatay.sonmez@boun.edu.tr; ersoy@boun.edu.tr).

A. Ozgovde is with the Department of Computer Engineering, Galatasaray University, Istanbul 34349, Turkey (e-mail: aozgovde@gsu.edu.tr).

Digital Object Identifier 10.1109/TNSM.2019.2901346

infrastructure with (i) many devices and gadgets getting in and out of the local edge coverage, (ii) a diverse set of applications resulting in a stream of tasks with varying complexity, (iii) networking state instantaneously changing and (iv) fluctuating computational capacity which is virtualized and dispersed over the local edge and cloud servers [6], [7].

In this heterogeneous environment, how the edge system responds to the incoming stream of offloaded tasks determine the overall efficiency and scalability of the system. Moreover, the satisfaction of the end user depends on whether tasks are handled in accordance with the quality expectations. With this vision in mind, one of the significant challenges of the edge computing systems is designing its operation so that dynamic flow of requests can be efficiently processed. Our work focuses on the workload orchestration problem which can be defined as deciding on the destination computational unit for each offloaded task within an edge computing system. We model the edge computing system as a multi-tier organization which incorporates virtualized computational resources at various levels, as well as, networking capabilities at different capacities, such as LAN, metropolitan area network (MAN) and WAN. In this respect, the workload orchestration problem should take into account the states of both computational and networking resources at hand.

In this study, we propose Fuzzy Logic as the tool to solve the workload orchestration problem in the edge computing systems. Fuzzy Logic is among the most employed methods to cope with the rapidly changing uncertain systems where it is difficult to provide accurate mathematical models [8]. In addition, the computational complexity of the fuzzy logic based approaches is low compared to other decision-making algorithms [9]. Being a computationally efficient solution is one of the most significant requirement of the online and real-time problems. Furthermore, Fuzzy Logic is based on well understood principles and provides an expressive enough method for stating complex operational details in a high level human understandable format. Our approach uses fuzzy rules to define the necessary workload orchestration actions in terms of networking, computation and task requirements at hand to decide the task execution location within the overall edge computing system. This enables administrators or end users to be exempt from identifying operational states and assigning procedures related to these states. Our work resembles intent based solutions proposed in [10] in the sense that the parameters that are relevant to the operation of the whole system are expressed as rules in a high level description.

We evaluated the performance of our system through a series of detailed simulation experiments. To achieve this, we realistically modeled a multi-tier edge computing infrastructure using EdgeCloudSim [11]. EdgeCloudSim is a simulation tool specifically developed for the edge computing domain, therefore it is capable of modeling the networking as well as the computational aspects of the edge and cloud servers at the VM resolution. Similarly, in our experiments the dynamic behavior of LAN, MAN and WAN networking related resources are incorporated.

To realistically model the incoming task traffic four application classes are considered, namely they are: Augmented Reality, Heath, Compute Intensive and Infotainment applications. Each application class represents a different profile in terms of task arrival distribution, delay tolerance and task length. Our method is comparatively evaluated with four opponent algorithms using crucial performance metrics such as service time, failed tasks, and VM utilization. According to the results, our proposed method has competitive performance with respect to its opponents for the cases studied.

The remainder of this paper is organized as follows. Section II presents the background of edge computing and reviews the related work. Section III puts forward the multi-tier edge computing architecture that we employ in our study. Section IV describes the proposed fuzzy logic based workload orchestrator. Section V elaborates on the simulation experiment design and analyzes the results. Section VI concludes our study and provides possible directions for future research.

II. Related Work

Orchestration in edge computing is a broad concept that is comprised of several management efforts on multiple levels. The mobile edge orchestrator (MEO) proposed in the ETSI MEC reference architecture has comprehensive responsibilities such as managing the available computing/storage/network resources and the life cycle of the applications [12]. However, an edge orchestrator can also have a confined scope instead of serving the entire system. As an example, Hegyi *et al.* [13] propose the concept of application orchestration which solves the application deployment problem on the edge. Karagiannis and Papageorgiou [14] propose an edge computing orchestrator that decides which application should be placed on which edge server. Santoro *et al.* [15] propose the workload orchestration concept that solves the workload placement issue in the fog computing environments. The workload orchestration basically decides where to offload the incoming client requests. Where to offload decision should be made by taking many parameters into consideration such as the network latency, data transfer size, and the remote server capabilities [16]. In our architecture, we name the functional unit which monitors the rapidly changing conditions and determines the target VM for offloading as the *Workload Orchestrator*.

Baktir *et al.* [17] propose a service-centric approach at the edge servers by using the orchestration capabilities of SDN. As a solution for tracking the services offered by the edge server in highly dynamic and mobile conditions, a service-centric edge computing solution based on SDN is used. SDN decouples the data plane from the control plane and provides networking capabilities via the north-bound interface. Two of the major components of their proposed system, the orchestration and load-balancing, are implemented by the north-bound applications. In their system, the orchestrator takes the list of IP addresses and forward it to the load balancer. The load balancer north-bound application decides on the destination server and sends that server's IP address to the controller.

Bittencourt *et al.* [18] discuss the workload offloading on the edge/fog computing infrastructures. The application requests

are served either in the edge or cloud according to three different policies. The decision can be based on the CPU capacity and delay requirements, but they use a static network delay for the WLAN, MAN and WAN communication. Therefore, the network congestion is not available in their work and this makes a fundamental difference in the problem modeling. As a result, Bittencourt *et al.*'s work is not our direct competitor, as our study takes into account network resources as well as computing resources.

Maio and Brandic [19] propose and evaluate a task offloading scheme that focuses on how an application composed of sub-tasks can be deployed over an MCC infrastructure with simplistic networking assumptions. Our work, on the other hand, focuses on a variety of applications being simultaneously run from numerous mobile devices and gadgets on the whole system within a realistic setting that connects edge servers through MAN whose performance varies as function of the current load.

The benefits of utilizing cloud computing and its central role on the offloading decision are explained in [20]. The authors state that, although the code offloading approach is studied largely on the mobile cloud context, many real cloud computing advantages are still not exploited properly. The overall offloading decision can take the advantage of cloud computing's characteristic features such as performance metrics, parallelization of tasks, and elasticity. To accomplish this, they proposed a fuzzy decision engine for offloading which considers both the mobile and the cloud variables. Our work also employs the fuzzy logic to accomplish its orchestration functionality. However, our work diverges from the authors' work on a multitude of levels including the technical problem focused on and the architecture assumed. Despite these differences, as discussed in Section IV-E, we adapted Flores *et al.*'s proposal to obtain a qualified competitor algorithm.

Another study proposing a fuzzy logic based code offloading method is done by Hosseini *et al.* [21]. In their work, the offloading ratio is calculated via a fuzzy logic based method by using parameters link delay and Signal-to-Noise Ratio (SNR). The authors demonstrate that using the fuzzy logic approach provides better results than using a weighted average of the two offloading parameters used. Similar to the previous fuzzy logic based approach, this study also is not a direct candidate to comply as a rival algorithm since it operates at the different layers, focusing on different decisions.

Fuzzy logic is an effective method that has been used for many years on uncertain nonlinear systems. In addition to the work done in [20] and [21], there are other studies that use fuzzy logic to solve different problems in the literature. Rathore *et al.* [22] solve the multi-criteria decision making problem of optimal security service selection by using fuzzy sets. Soleymani *et al.* [23] use fuzzy logic to measure the trust level of the sender vehicle of an event for the vehicular ad-hoc networks. However, the edge computing and the workload orchestrator concepts have been discussed for the last few years. To the best of our knowledge, a fuzzy logic based workload orchestrator has not been studied yet for this domain.

III. Proposed Workload Orchestration in Multi-Tier Edge Computing Systems

In the early proposals of edge computing, the interaction between the user device and a single edge server was of primary concern [2]. However, as edge computing moves toward becoming a mainstream paradigm, a more complete view of the overall system including both edge and cloud servers has emerged. As the edge services are envisioned to operate in coordination with the cloud tier as well, a multi-tier architecture covering both types of components becomes a necessity. In that respect, we incorporate multiple tiers in our edge computing architecture as previously depicted in Fig. 1.

A. Multi-Tier Edge Computing Architecture

There are basically three tiers available in our architecture. The mobile devices themselves represent the first tier where they make use of their wireless transmission interference to interact with wireless local area network (WLAN) gateways. In the second tier, both the edge servers and the edge orchestrator are located. The edge orchestrator is aware of the status of the edge servers as well as the network resources which are used for workload orchestration. Finally, the third tier consists of the global cloud servers which are typically provided by the cloud service providers such as Amazon EC2 and Microsoft Azure.

To clearly demonstrate how our proposed approach fits into a real world implementation, we modeled the experimentation phase with a focus on university campus scenario. According to this model, the students and other participants of a typical campus demand services from the edge servers within the campus while moving around the predesignated locations such as the classroom buildings, cafeteria, dormitories, and administrative offices. The individuals are assumed to carry and/or wear the mobile gadgets which continuously run applications that in turn create a traffic due to offloading of the tasks. There exists certain number of edge servers located in the campus such that they provide coverage for edge computing requests. A user is connected to the nearest edge server via WLAN whereas the edge servers are interconnected by MAN. Also, the standard cloud infrastructure can be accessed over WAN. In this setting, it is possible for the user to offload a task to (i) Nearest edge server (we call this the local edge server) (ii) a neighboring edge server within the campus connected by MAN (we call this the remote edge server), and (iii) to the global cloud servers. In our work, this decision is handled by the workload orchestrator. To realistically model the mobility of the users, the locations in the campus are assumed to have different attractiveness levels. The attractiveness level affects the dwell time of the related place. For example, the students may spend more time in a library or in a cafe compared to the other locations. The user density, hence the requested traffic volume, at each place varies depending on the attractiveness level of the place. We use three location categories with different attractiveness levels as shown in Fig. 2. A *Type 1* location is shown in red ellipse and it has the highest attractiveness level. On the other hand, a *Type 3* location is shown in blue ellipse and it has the lowest attractiveness level. When it comes

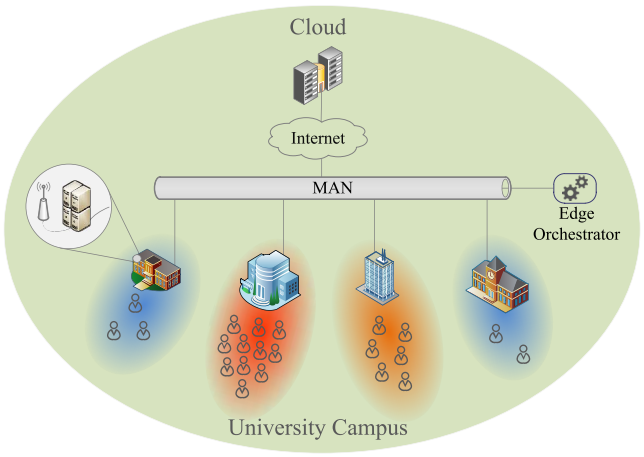


Fig. 2. Proposed scenario representing multiple locations of a campus distributed in a university.

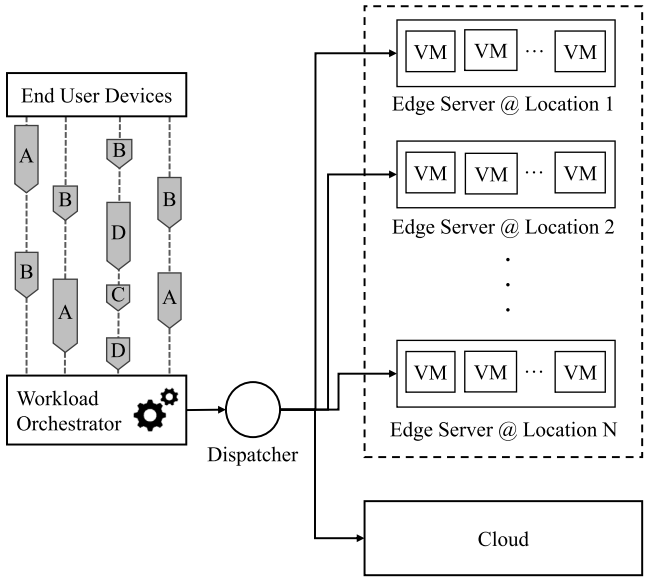


Fig. 3. Workload orchestration flow.

to the *Type 2* location, it has the medium attractiveness level and shown in orange ellipse. The mobile device density affects the overall system performance due to its direct relation with the requested traffic size. For example, both the network and the computational resources are likely to be congested in a *Type 1* location because of the high device density.

B. Workload Orchestration

The orchestration is critical for the efficient and robust operation of the multi-tier edge computing systems. With this need in mind, MEO has been already defined by the ETSI [5]. According to ETSI's proposal, MEO knows the topology of the network, (the deployed hosts and the mobile devices), the services provided by the hosts, available resources on each host and the applications used by the mobile devices. MEO works in a centrally controlled manner and it is mainly responsible for selecting the appropriate mobile edge host(s) based on constraints, such as latency, available resources, and available services [24]. In our work, an important component of the overall orchestration effort, namely the workload orchestration, is focused on. The workload orchestrator determines the target computational unit against an incoming request by monitoring multiple factors such as the network bandwidth, edge server utilization, and task characteristics. In our architecture, the workload orchestrator receives the offloaded tasks from the user devices. The target server is decided via the two-stage decision making process. As the first step, the workload orchestrator finds the favorable server in the edge layer. Then, it determines whether the task should be executed on the corresponding edge server or the cloud.

There exist many system parameters which affect on this decision. In Section IV, we explain how this problem can be solved using fuzzy logic. The task flow in the proposed multi-tier edge computing architecture is illustrated in Fig. 3.

In order to execute a task on the servers, some requirements should be met. Firstly, there should be enough capacity on the network for successful offloading. Secondly, the tasks increase the utilization of the servers based on their complexity. Task complexity is chosen from a distribution characterizing the application profile. There should be enough capacity on the server, to execute the related task on it. When it comes to the task length, it is not a conditional requirement, in fact, it directly affects the execution time of the task. Different task length distributions are selected to realistically model the incoming task traffic to the proposed edge system. The dependency between the tasks is not addressed in this study. Therefore, the tasks are considered as stateless jobs. However, a separate framework that handles the task dependency can easily utilize our proposed solution. Since the workload orchestrator contains centralized information, the tasks tagged as dependent could be executed on the same edge server to eliminate the extra communication overhead.

In order to design an efficient edge orchestrator, sufficient amount of information will be needed. However, the data transmission frequency between the edge orchestrator and the other devices should be carefully designed in order not to cause a communication overhead. Therefore, the location of the edge orchestrator becomes an important design decision. Practically, the edge orchestrator can be deployed in any tier in the topology, independent of the technology but dependent on the business model. It can be located on the cloud, on the edge or directly on the mobile devices in a distributed manner. For example, Imagane *et al.* [25] deploy the edge orchestrator on the edge layer for dynamic scaling of the computational and network resources. They use OpenStack to construct the edge computing platforms in their university. Each edge computing platform also includes a regional controller node and the orchestrator connects to each controller node via an edge network. Our work assumes WLAN for the edge access, however, the proposed system can equally be applied to the telecom based networks with cellular access technologies. This would result in a different delay model. Also, the orchestrator location should then be chosen in accordance with the requirements and constraints of the mobile operator. For instance, operating the multi-access edge orchestrator on the Cloud

Radio Access Network (Cloud RAN) in next generation 5G networks could be an alternative.

IV. Fuzzy Logic Based Workload Orchestrator

For the effective workload orchestration of a multi-tier edge computing environment, main factors that should be considered are the utilization level of the servers and the network conditions on the routes to these servers. These parameters can be highly dynamic especially under unexpected variation of the load. For instance, as the number of users increase in a shared network, the bandwidth fluctuations are inevitable. Similarly, the CPU utilization of a VM can exhibit frequent change depending on the tasks running on it. This uncertainty makes it difficult to decide where the offloaded task should run. A realistic model of the edge computing system involves many parameters coming not only from the edge infrastructure itself but also from the end user device and application characteristics, which further complicates the workload orchestration problem. Multi-constraint optimization with a mathematical model is not an easy task, and generally, a robust model cannot be found due to the complexity of the problem. Moreover, the conventional offline optimization techniques cannot be applied for the orchestration problem since the offloading requests are not known in advance. Due to the incomplete information on the arrival times and sizes of the tasks, the problem dictates an online solution. Fuzzy logic in this respect is a good alternative to handle real-world imprecision in the area of computer networking [26]. It has been successfully applied to many networking related areas such as load management, network management, buffer management and congestion mitigation. We employ fuzzy logic in our workload orchestrator design for a multitude of reasons. Firstly, it can handle the uncertainty in unpredictable environments without the need of detailed mathematical models. In addition, fuzzy logic easily handles the multi-criteria decision making process by allowing the consideration of multiple parameters in the same framework [27]. Secondly, it has low computational complexity which is an important criterion for an online algorithm. Thirdly, it provides an intuitive interface for system operators since fuzzy logic mimics expert judgment by modelling through the use of imprecise information provided in the form of human language. Finally it enhances the performance of the system significantly as demonstrated in the result section.

A. Two Stage Fuzzy Workload Orchestration

The proposed fuzzy logic based workload orchestrator finds a target server which can be the local edge server, remote edge server or the cloud server. Handling all of these servers in a single FLS is a complex operation. Therefore, we use two stage FLS to decrease this complexity as shown in Fig. 4. In the first stage, the most convenient edge server in the edge layer is found. In the second stage, the candidate edge server and the cloud server are compared. As a result of these operations, the target edge or cloud server is determined.

FLS is basically used for mapping uncertain variables from complex real world to a scalar data [28]. We use a fuzzy logic based orchestrator to map different metrics into a single value.

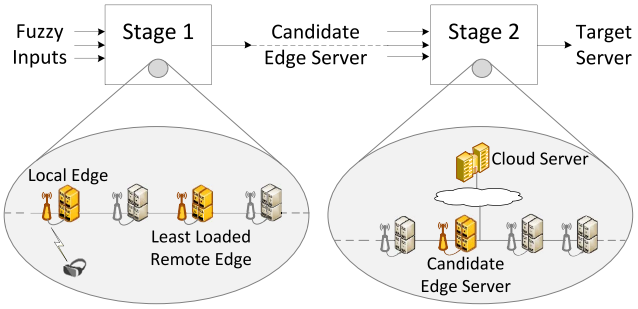


Fig. 4. Two stage fuzzy workload orchestration.

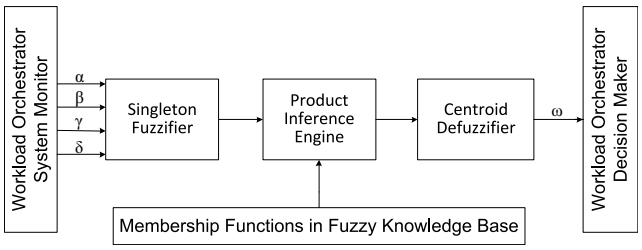


Fig. 5. Fuzzy inference system for the proposed workload orchestrator.

The components of the proposed FLSs are singleton fuzzifier, product inference engine and the centroid defuzzifier. These components are depicted in Fig. 5. Essentially, both FLS used in both stages follow the same steps with different values. Since a more complex problem is solved in the second stage, the detailed explanation of the proposed FLS used in this stage is explained first.

B. Fuzzy Input Variables of the Second Stage

The second stage of our fuzzy logic based workload orchestrator operates on four input variables as shown below:

$$F = (\alpha, \beta, \gamma, \delta) \tag{1}$$

where α is *WAN Bandwidth*, β is *the length of the incoming tasks*, γ is the VM utilization on the edge server and δ is the delay sensitivity of related task.

Normally, any number of input variables can be used in fuzzy logic. In our work, we chose variables that have significant effect on the system task execution performance. WAN bandwidth and the utilization of the edge servers are the essential factors that contribute to the probable bottleneck experienced on the system. Compared to these factors, the probability that the WLAN itself or the computational resources on the cloud servers being a performance barrier is very low. Characteristics of the requested tasks of the applications also plays a decisive role on the system performance. To provide a formal input representing the task properties to our fuzzy system, we make use of the length of task and its sensitivity to delay, both of which are determined according to the profile of the application creating the demand.

1) WAN Bandwidth: The WAN Bandwidth is an important indicator which has to be considered while offloading tasks to the cloud server. For example; if the WAN communication latency is higher than the application QoS requirement or if

the network congestion is too high to cause packet losses, offloading to the cloud server is not beneficial.

2) Task Length: The Length of Tasks basically determines the execution time of the task. In our study, the length of a task is denoted by the number of instructions. It is assumed that the system has a load characterization approach to estimate the service type of the incoming tasks. The computational capacity of the VMs used in our simulations is measured in terms of giga instructions per second (GIPS), and the execution time of a task is calculated in accordance with its length. The task length can be used as a metric for the offloading decision. A complicated task containing too many instructions might be a good candidate for offloading to a cloud server. On the other hand, an simple task can be executed on the edge server or even on the end device itself.

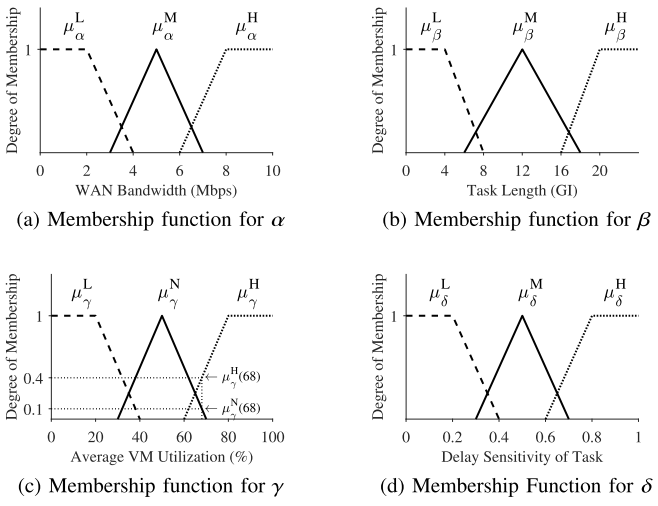
3) Delay Sensitivity of Task: The delay sensitivity indicates the tolerance of the task to taking longer time to execute either due to network conditions or due to server utilization levels. Using this indicator while making "where to offload" decision is beneficial for increasing the overall QoS provided by the edge computing system. In our model, the delay sensitivity parameter is assumed to be determined by the application profile. In the real life examples, it can either be defined by the network administrator or embedded into the tasks by the application developer.

4) VM Utilization: The VM utilization figure gives us information on the residual computational capacity of the edge server. The edge server can be considered as congested if the utilization is above a threshold level. In such a case, offloading to the central cloud server might be a feasible alternative depending on the WAN conditions and the delay sensitivity of the task. The edge servers run multiple VMs in our study, therefore the VM utilization represents the average utilization of all VMs running on the related edge server.

C. Fuzzy Inference System

Inputs and outputs of the FLS are non-numeric linguistic variables. Instead of using the numerical values, FLS uses variables from the natural language. Our FLS has different linguistic variables for each indicator. For α, β and δ, we use *low* (L), *medium* (M), and *high* (H) as the linguistic variables. For γ, we use different terminology; here we have *light* (L), *normal* (N), and *heavy* (H) as the linguistic variables.

The membership functions are defined to be used in fuzzification and defuzzification steps. We use the membership functions to quantify a linguistic term. For each fuzzy variable, a set of membership functions are defined, and this set consists of different functions for each linguistic term. In our case, we have four membership function sets and each of the sets has a different function for related linguistic variables. The membership functions can be in different forms such as triangular, trapezoidal, piecewise linear, Gaussian, or singleton [28]. We chose the triangular form which is the most commonly used one. Our membership functions are depicted in Fig. 6. Deciding the values used in the membership functions is a critical task; since it has significant impact on the



(a) Membership function for α (b) Membership function for β

(c) Membership function for γ (d) Membership Function for δ

Fig. 6. Membership functions for the fuzzy variables representing both network and computational resources.

FLS performance. Similar to the other fuzzy decision mechanisms in other applications in the literature, the degree of membership values for each fuzzy variable are determined empirically. We tested various membership functions in our experiments and the best combination among the trials is used.

The first step of FLS is the fuzzification step. The fuzzification is the process of transforming crisp values into fuzzy values by using the membership functions which associate a grade to each linguistic term. The fuzzifier defines fuzzy classes for each fuzzy variable as given below:

$$F_\alpha(x) = \left[\mu_\alpha^L(x), \mu_\alpha^M(x), \mu_\alpha^H(x)\right] \tag{2}$$

$$F_\beta(x) = \left[\mu_\beta^L(x), \mu_\beta^M(x), \mu_\beta^H(x)\right. \tag{3}$$

$$F_\gamma(x) = \left[\mu_\gamma^L(x), \mu_\gamma^N(x), \mu_\alpha^H(x)\right] \tag{4}$$

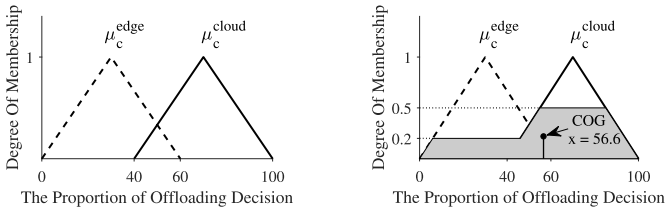
$$F_\delta(x) = \left[\mu_\delta^L(x), \mu_\delta^M(x), \mu_\delta^H(x)\right] \tag{5}$$

Fuzzy values are calculated via membership functions of the fuzzifier which are presented in Fig. 6. For example, if the VM utilization on edge server is 68%, the degree of membership values of light, normal and heavy fuzzy variables will be 0, 0.1, and 0.4 respectively, since $F_\gamma(68) = [0\ 0.1\ 0.4]$ as shown in Fig. 6(c).

The second and the most important step of our FLS is the inference step. Inference is used to evaluate and combine the fuzzy rules from the fuzzy rule base. The result of inference system is a fuzzy variable that is used in defuzification step. A fuzzy rule is a simple IF-THEN rule with a condition and a conclusion. In fuzzy rules linguistic variables are used, such as ***IF*** *WAN Bandwidth is high* ***AND*** *Task Length is very high* ***THEN*** *offload to cloud*. Determining the fuzzy rules is very critical since the overall system performance highly depends on these rules. In our study, the relatively better fuzzy rule set is found empirically and the best rule combination in the computational experiments are used. Since there are 4 membership functions with 3 linguistic terms, the number of fuzzy rules is 81. Some example rules are given in Table I.

TABLE I
EXAMPLE FUZZY RULES IN THE KNOWLEDGE BASE

Rule Index	α	β	γ	δ	c
R1	low	low	light	high	edge
R2	low	high	normal	high	edge
R3	high	high	normal	low	cloud
R4	medium	medium	heavy	high	cloud
R5	high	high	heavy	high	cloud



(a) Output membership function (b) Center of gravity calculation

Fig. 7. Membership function of the consequence classes.

Basically, the aggregation, activation and the accumulation methods are used in inference steps. The aggregation method, which is called as combination as well, is the rule connection method. In aggregation phase, we use *Minimum* and *Maximum* functions for *AND* and *OR* operators respectively. The activation method defines how the evaluated IF part result is applied to the THEN part of the rule. The most common activation functions are *Minimum* and *Product*. In the activation phase, we use Minimum function. Finally, the accumulation method determines how the results of multiple rules are combined within a rule set. Generally, *Minimum* and *Maximum* functions are used as the accumulation method. In the accumulation phase, we use Maximum function. The inference can be explained with a simple example. Assume that there are only five rules, *R*1 to *R*5, defined in our rule set. If the WAN bandwidth, task length, VM utilization and delay sensitivity of the task values are given as 2 Mbps, 6 giga instructions (GI), 35 percent and 0.7 respectively, fuzzy value for selecting the edge server can be calculated by using Equation (6).

$$\mu_{edge} = max\left\{\mu_{edge}^{R1}, \ldots, \mu_{edge}^{Rn}\right\} \tag{6}$$

Since selecting the edge server is being analyzed in this example, the aggregation and activation phases are applied for *R*1 and *R*2:

$$\mu_{edge}^{R1} = min\left[\mu_{\alpha}^{L}(2), \mu_{\beta}^{L}(6), \mu_{\gamma}^{L}(35), \mu_{\delta}^{H}(0.7)\right] \tag{7}$$

$$\mu_{edge}^{R2} = min\left[\mu_{\alpha}^{L}(2), \mu_{\beta}^{H}(6), \mu_{\gamma}^{N}(35), \mu_{\delta}^{H}(0.7)\right] \tag{8}$$

After calculating μ_{edge}^{R1} and μ_{edge}^{R2}, the fuzzy value for selecting the edge server can be found as 0.25 in the accumulation phase:

$$\mu_{edge}^{R1} = min[1, 0.5, 0.25, 0.5] = 0.25 \tag{9}$$

$$\mu_{edge}^{R2} = min[1, 0, 0.25, 0.5] = 0 \tag{10}$$

$$\mu_{edge} = max[0.25, 0] = 0.25 \tag{11}$$

The last step of the proposed FLS is the defuzzification step. Defuzzification is a process of transforming the fuzzy output inferred from the inference engine to a crisp value. We use a centroid defuzzifier in the defuzzification step. The centroid defuzzifier returns the center of the inferred fuzzy outputs. The centroid calculation is the most popular defuzzification method which returns the center of gravity of the area under the curve [29]. In Fig. 7(a), membership functions of the consequence classes are depicted. Output of the centroid defuzzifier can be calculated by using Equation (12).

$$\omega = \frac{\int x\mu(x)dx}{\int \mu(x)dx} \tag{12}$$

This method is similar to the formula for calculating the center of gravity. After applying the centroid defuzzifier, the output of the FLS, ω, becomes a crisp value between 0 and 100. We offload to the edge server if ω is less than 50, otherwise we offload to the cloud server. For example, if the μ_{edge} and μ_{cloud} are calculated as 0.2 and 0.5 by the fuzzy inference block, the crisp result will be 56.6 as shown in Fig. 7(b). So, the incoming task is offloaded to the cloud.

D. Fuzzy Input Variables of the First Stage

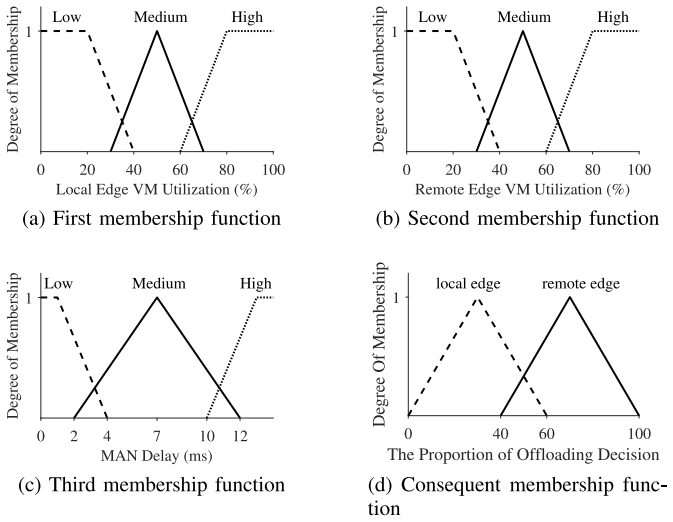
The fuzzy inference system used in the first stage is similar to the second stage. There are only three differences as the input variables, fuzzy membership functions and the fuzzy rules. The number of fuzzy rules used in the first stage is 27, because there are 3 input variables with 3 linguistic variables each. The fuzzy rules are not given in this study, but the input variables and fuzzy membership functions are briefly addressed. In this stage, we use 3 input variables which are MAN delay, CPU utilization of the least loaded remote edge server and the CPU utilization of the local edge server.

1) MAN Delay: When the number of clients utilizing the MAN is very high, the MAN delay becomes an important bottleneck in the system. Therefore, we consider the MAN delay while deciding the most favorable edge server. If the MAN delay is higher than expected, it can be considered that the MAN resource is congested. In this case, offloading to the local edge server is advantageous.

2) Least Loaded Remote Edge CPU Utilization: CPU utilization of the edge servers provides the status of computational resources. The MAN utilization cost of offloading to any remote edge server is the same. Therefore, we use the CPU utilization of the least loaded remote edge server. If the CPU utilization of the least loaded remote edge server is low compared to the local edge server, offloading to the remote edge server is advantageous if the MAN delay is low.

3) Local Edge CPU Utilization: Generally, offloading to the local edge server is beneficial in order not to consume the MAN resource. However, choosing the local edge is not rational if the number of users at the different locations are not uniformly distributed. Always offloading to the local edge may cause too many task failures due to the inadequate computational resources especially in the hotspot locations. If the MAN resource is not congested, distributing the incoming tasks among the edge servers may increase the system performance.

In the first stage of the proposed FLS, we use *low*, *medium*, and *high* as the linguistic variables. As it is explained, the



(a) First membership function (b) Second membership function

(c) Third membership function (d) Consequent membership function

Fig. 8. Membership functions used in the first stage.

corresponding membership functions are used for each fuzzy input variable. The membership function of the consequence classes is also used in the defuzzification step. The membership functions of the fuzzy input variables and consequence classes are shown in Fig. 8.

E. Benchmark Solutions

The workload orchestration on the multi-tier edge computing in which the edge and cloud servers are used is a new concept. Therefore, a competitor workload orchestrator algorithm could not be found in the literature. Flores and Srirama [20] work is one of the fuzzy logic based approaches as for code offloading. Originally, their work utilizes the fuzzy logic to decide executing the tasks on the mobile device or the cloud server. Whereas, our work assumes that the tasks are already offloaded from the mobile devices and the fuzzy logic is used to decide for the edge or cloud offloading. We adapted Flores *et al.*'s work in a way to execute their fuzzy logic on the edge server to decide on executing the tasks on the edge or cloud server. They use the *bandwidth*, *transferred data size*, *CPU speed* and *video execution ratio* as the input variables of FLS. All these variables are already available in our simulation environment except the video execution ratio which indicates how much face recognition request is generated by the offloaded task. According to the explanation of authors, the face recognition is a time-critical operation which requires significant computational resources. It can be inferred that the video execution ratio determines the urgency type of compute-intensive tasks to be offloaded. In our simulation environment, we use the delay sensitivity variable to define the urgency of requests, and the augmented reality application has the highest delay sensitivity value. Therefore, the video execution ratio is a metric similar to the delay sensitivity of tasks. As a result, while adapting Flores *et al.*'s approach we use the delay sensitivity value instead of the video execution ratio.

In addition to the Flores *et al.*'s work, other competitor algorithms are also implemented. Generally, the WAN bandwidth and the CPU utilization of edge servers are the main bottlenecks of the multi-tier edge computing architectures. Therefore, they have a significant impact on the overall system performance. We implement competitor algorithms which handle workload orchestration depending on the status of these parameters. This strategy is commonly used because it is easy to implement and it does not bring a considerable overhead on the system. For example, load balancing in enterprise cloud services is done by considering the CPU utilization value. Our algorithms make use of *WAN Bandwidth*, *CPU utilization of VMs*, and *Hybrid (both bandwidth and CPU utilization)*. Bandwidth based method prefers to offload to the cloud servers if the WAN is not congested. Therefore the main aim of this algorithm is selecting the cloud server as long as the Internet connection is satisfactory. On the other hand, CPU utilization based method favors offloading to the edge servers if the CPU utilization of VMs is not too high. The objective of this algorithm is selecting the edge servers as long as they are not congested in terms of CPU utilization. The Hybrid method is more complex when compared to others. It checks both the WAN bandwidth and the CPU utilization of VMs in the decision process. These algorithms use the threshold values to decide offloading an incoming task to the edge or cloud server. The selected WAN bandwidth threshold should be as low as possible in order to maximize cloud offloading, but high enough to avoid network congestion. CPU utilization based method prevents offloading the incoming tasks to the edge server if the CPU utilization is higher than a specific threshold value. However, there is no utilization limit in bandwidth based and hybrid methods, so they allow for full CPU utilization on the edge server side. The threshold of average VM CPU utilization on edge server should be as high as possible in order to maximize edge offloading, but low enough to prevent CPU congestion. We run various simulations with different parameters to find the minimum WAN bandwidth (*min-bw*) and the maximum CPU utilization (*max-cpu*) values which provide the best performance. The candidate threshold values are 12, 8, 5, 3 Mbps and 90, 80, 70, 60 percent for *min-bw* and *max-cpu* in our experimental study. According to the simulation results, the best values for *min-bw* and *max-cpu* are observed as 5 Mbps and 80% respectivelly.

As explained in Section IV, the proposed workload orchestrator can offload an incoming task to the cloud, to the local edge server or to the remote edge server. On the other hand, the competitor algorithms are only capable of determining the layer (edge or cloud) to be selected for task offloading. If the task is decided to be offloaded to the edge layer, the target edge server is selected via the least loaded algorithm. This algorithm basically checks the CPU utilization of all VMs on the edge layer, and selects the VM with the least CPU usage. If MAN communication is not creating significant delay, this approach can give successful results. Since, we consider limited MAN resources in our decision logic choosing a VM among all edge servers can cause a serious congestion problem on the MAN. We also evaluated different task provisioning algorithms on the edge such as first-fit, next-fit, and best-fit. The least loaded and the next-fit algorithms provide the best performance. We selected the least loaded algorithm for our system. If the

Algorithm 1 Unified Algorithm Representing Competitor Orchestrators

Input: offload strategy S, incoming task T
Output: target layer to offload O

initialization :
read WAN bandwidth bw
read average VM Utilization u
if $S = Bandwidth\ Based$ **then**

  if $bw > min{-}bw$ **then**
    $O = cloud$
  else
    $O = edge$
  end if
else if $S = Utilization\ Based$ **then**

  if $u < max{-}cpu$ **then**
    $O = edge$
  else
    $O = cloud$
  end if
else if $S = Hybrid$ **then**

  if $bw > min{-}bw$ **and** $u > max{-}cpu$ **then**
    $O = cloud$
  else
    $O = edge$
  end if
else if $S = Flores*$ **then**
  $O = use\ modified\ Flores\ algorithm$
end if
return O

objective function is different, such as minimizing the energy consumption on the edge servers, other sophisticated strategies can be also operated to find the optimum edge server to offload. The competitor workload orchestrator algorithms can be unified in a single algorithm as given in Algorithm 1.

V. Performance Evaluation

We performed extensive number of experiments to evaluate the performance of our fuzzy logic based workload orchestrator. EdgeCloudSim [11] which enables us to model a realistic edge computing environment is used as the simulator. It provides a modular architecture to provide support for a variety of crucial functionality such as network modeling specific to WLAN and WAN, device mobility model, realistic and tunable load generator. For modeling the computational tasks such as VM creation with a given capacity, EdgeCloudSim relies on the capabilities of CloudSim [30]. CloudSim has a long known and reliable code base for the simulation of computational actions. EdgeCloudSim is publicly available as an open source project. Modular design and open-source code base of EdgeCloudSim allow its users to incorporate their own needs in the tool.

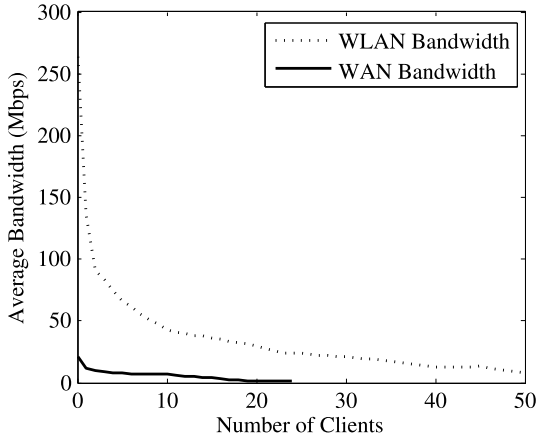


Fig. 9. Empirical evaluation of WAN/WLAN connections.

A. Modelling Networking Delay

We use the result of an empirical study for the WLAN and WAN delays instead of a mathematical model to achieve a more realistic simulation environment by using values from the real life. However, the MAN delay is observed via a single server queue model with Markov-modulated Poisson process (MMPP) arrivals. The mean arrival rates of the tasks are updated when the state of the system congestion level is changed. It is assumed that all clients use a shared MAN which has Gigabit Ethernet like capacity. The clients use MAN if they must be served by a remote edge server due to the congestion at their own location. In this case, the edge server is accessed via two hops where the packets go through WLAN and MAN.

While measuring the WLAN bandwidth, two computers representing the server and client are used. The computer used as a server is connected to the access point via Gigabit Ethernet. The other computer representing the clients is connected to the access point via an 802.11 family WLAN. The available bandwidth value is calculated while multiple clients (processes) are communicating with a server simultaneously. The average bandwidth of ten consecutive experiments is used for the WLAN bandwidth. To measure the WAN bandwidth, an empirical study is carried out for characterizing the typical home/office Internet connection capacity in Istanbul, Turkey. In this study, the static files located in Amazon simple storage service (S3) are downloaded simultaneously by the multiple clients. The WAN bandwidth values are determined by the average values of the measurements taken at different times of the day for a week. The results of the WLAN and WAN bandwidth measurements are shown in Fig. 9. The number of clients given in Fig. 9 is not the number of mobile devices connected to the same access point. In fact, that number indicates the number of mobile devices communicating with the edge or cloud server at the same time. We observed that it is unlikely to transmit data when more than a certain number of clients use the same network resources simultaneously. Therefore, it is assumed that the packets are dropped due to the congestion when the maximum number of clients that can use the network resources is reached. This number is determined as 25 for the WAN and 50 for the WLAN communication.

TABLE II
APPLICATION TYPES USED IN THE SIMULATIONS

	Augmented Reality	Health App	Compute Intensive	Infotainment App
Usage Percentage (%)	30	20	20	30
Task Interarrival (sec)	2	3	20	7
Delay Sensitivity (%)	0.9	0.7	0.1	0.3
Active/Idle Period (sec)	40/20	45/90	60/120	30/45
Upload/Download Data (KB)	1500/25	20/1250	2500/200	25/1000
Task Length (GI)	9	3	45	15
VM Utilization on Edge (%)	6	2	30	10
VM Utilization on Cloud (%)	0.6	0.2	3	1

TABLE III
SIMULATION PARAMETERS

Parameter	Value
Simulation Time/Warm-up Period	33/3 minutes
Number of repetitions	60
WAN/WLAN Bandwidth	empirical
MAN Bandwidth	MMPP/M/1 model
LAN Propagation Delay	5 milliseconds
Number of VMs per Edge/Cloud server	8/4
Number of Cores per Edge/Cloud VM CPU	2/4
VM CPU Speed per Edge/Cloud	10/100 GIPS
Mobility Model	Random Way Point
Probability of selecting a location type	Equal
Number of locations Type1/2/3	2/4/8
Mean dwell time in Type 1/2/3	2/5/8 minutes

B. Simulation Setup

In a generic edge computing environment, mobile devices can run several services/applications. The characteristics of the potential applications may vary according to the requirements. The task offloading request to the edge or cloud server generates a different level of CPU and network load on the system. For instance, a heavy computational task requires large amount of CPU resources, whereas a data backup application requires small amount of CPU resources. On the other hand, the data backup process needs significant amount of data transmission, while the compute intensive task may not use too much network capacity. In order to simulate real life more realistically, we use four different application types in our study. To decide on the application types we considered the most presented edge computing use cases given in the literature. For example, Silva *et al.* [31] propose an augmented reality application on Google Glass, Guo *et al.* [32] discuss the infotainment applications, and Tunca *et al.* [33] present a health application which uses a foot-mounted inertial sensor to analyze the walking pattern of the users. In our simulations, it is assumed that the mobile devices offload tasks which belong to a predefined set of application categories. The user wearing the smart glass offloads captured pictures to the remote servers which provide face recognition service. The user carrying a foot-mounted inertial sensor offloads sensor data to the remote servers which provide fall risk detection service. Similarly, infotainment and the compute intensive applications send their tasks to the remote servers which provide related services. The characteristics of application types used in the simulations are listed in Table II. The *usage percentage* of the application stands for the portion of the mobile devices running this application. The *task interarrival* time defines how frequently the related task is sent to the edge orchestrator. In our study, interarrival time of the tasks is exponentially distributed. The *delay sensitivity* is basically used for determining if the task is delay intolerant (real-time) or not. If this value is high, the related application is delay intolerant. The *active/idle* period is used while generating the task. Related mobile device generate tasks during the *active* period, and it just waits in the *idle* period. The *upload/download* data sizes depict the typical values for data sent to/received from the server. For instance, <1.5 MB,25KB> upload/download size is chosen for augmented reality to reflect the fact that an image is uploaded and a text metadata is received as the response. The *task length* determines the required CPU resource for the related task in GI unit. Since the computational capacity of the server is measured in terms of its GIPS value, the task size is a determinant of the duration of its execution. Like the task interarrival time, the tasks length is also an exponentially distributed random variable. The *VM utilization* parameter is the CPU overhead on the VM while the related tasks is running on it. The VMs are capacitated in terms of CPU in our simulations, if there is not enough CPU resource on the VM for the incoming task, it would fail.

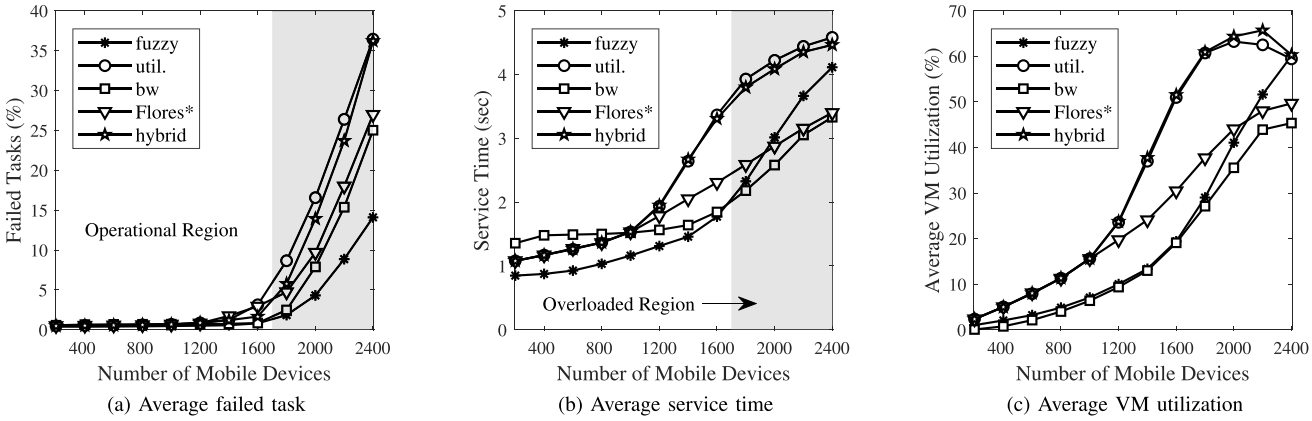
Other important simulation parameters are listed in Table III. We assume that residents move within the campus according to a nomadic mobility model [34]. The nomadic model dictates a user goes to a location stay there (dwell time) and moves to another location. In our model, there are three location types with different attractiveness levels and the residents stay in these locations for random dwell times proportional to the attractiveness level. Average dwell values for each location is given in Table III.

In the real life scenarios, the applications depict an intermittent behavior where they actively generate tasks for a duration and stay idle between active periods. For example, the health application may record the sensor data for a while, then it sends the collected data to the remote server for further processing. We model this pattern using values presented in Table II. It is assumed that each location is covered by a dedicated wireless access point and the mobile devices join related WLAN when they move to the related location. After joining the WLAN, mobile devices start sending tasks to the edge or cloud server. If the task is offloaded to the cloud server, broadband connection provided by the wireless access point is used. Whereas, MAN connection is used if the task is offloaded to the remote edge server. In this scenario, MAN can be a Gigabit Ethernet of campus LAN.

In our previous work, we investigated other architectures where the edge orchestrator is not available [35]. When there is no edge orchestrator, all the tasks are executed at the local edge server which is connected to the same access point with the mobile device. Therefore, the resources are not utilized efficiently and high congestion is observed on edge servers located at the hotspot locations. The result of our previous work is not presented in this study, because its performance is too poor to compare with the proposed algorithms.

C. Simulation Results

The results of our simulation in EdgeCloudSim consist of the average number of failed tasks, the average service



(a) Average failed task (b) Average service time (c) Average VM utilization

Fig. 10. Comparative evaluation of the proposed mechanisms based on all application types.

time, the average VM utilization and the average network delay. In our figures, we use *Flores**, *fuzzy*, *util.*, *bw*, and *hybrid* abbreviations for *Modified Flores-Based*, *Fuzzy-Based*, *Utilization-Based*, *Bandwidth-Based*, and *Hybrid* workload orchestrators respectively. The main performance criteria are compared in Fig. 10 by considering the average results of all applications.

1) Failed Task Performance: The average unsuccessful packet or task ratio is one of the most important performance criteria in computer network design. The average failed task percentage in our architecture is given in Fig. 10(a). The performance of all approaches are similar when the system is lightly loaded. However, the *Fuzzy-Based* approach provides better results when compared to its competitors as the load increases. It can balance both computational and the communication (networking) resources better than its competitors. When it comes to the other approaches, one of the main reasons of providing poor performance is the MAN congestion. Since the competitors do not consider the status of the MAN resources, they experience too many network losses on the MAN when the system is loaded. The *Utilization-Based* approach provides the worst results because it uses the CPU utilization as a threshold and after the computational resource of VMs are congested, incoming tasks are sent to the cloud. As a result, after some time most of the tasks drop on the network due to the WAN congestion.

2) Service Time Performance: Another important performance metric is the service time, since the completion time of an objective is generally significant for many users. The average service time of our algorithms are shown in Fig. 10(b). The service time includes both network delay and the processing time. When the system load is light, the *Fuzzy-Based* approach provides the best result, because it takes both computational and network resources into the consideration and makes better decisions. Since the *Bandwidth-Based* approach prefers offloading the tasks to the cloud, it provides the worst performance due to the WAN delay. When the system load is high, both the processing time of the tasks and network delay are getting higher due to the congestion. In this case, especially the WAN resources become congested. The *Bandwidth-Based* and *Modified Flores-Based* approach prefer offloading the tasks to the edge, so they provide better results. The *Utilization-Based* and *Hybrid* algorithms decide to offload to the cloud due to the congestion on CPU resources on the edge servers. Since the WAN delay is higher than the processing time, they provide the worst performance. It can be argued that the average service time of the *Fuzzy-Based* approach becomes worse than the others when the number of mobile users is higher than 1800. However, in delay and loss systems the average loss and delay should be considered together. In the shaded region of Fig. 10(a) it can be seen that a considerable amount of tasks cannot be served by the competitor algorithms. Therefore, the *Fuzzy-Based* approach provides longer service time than the others if the system is overloaded as shown in the shaded region of Fig. 10(b). Considering the average number of failed tasks, it is quite likely that the desired operational range of this particular instance is limited to 1800 mobile devices. If the number of mobile devices are too high, the ceiling effect in the average service time results is more likely to be experienced by all algorithms, since most of the requests are blocked.

3) CPU Utilization Performance: The average CPU utilization of VMs running on the edge servers are shown in Fig. 10(c). Providing low CPU utilization is not always a desired result. If the system has lower CPU utilization with more VMs, it means that, there is a waste of resources. However, if it has lower CPU utilization by using the same number of VMs as the competitors, it means that, the related system uses resources more efficiently. According to the average CPU utilization results of our algorithms, it can be seen that the *Fuzzy-Based* and *Bandwidth-Based* approaches use computational resources of VMs better than the other algorithms when the system is not heavily loaded. For all the algorithms except the *Fuzzy-Based*, the average CPU utilization increases to a certain point, then it becomes stationary and finally a decrease trend is observed. Since, if the system is heavily loaded, the average CPU utilization in hotspot locations becomes very high, but the edge servers located at the locations with low attractiveness levels could not be utilized well by the competitor algorithms due to the MAN communication failures. Since the *Fuzzy-Based* approach considers

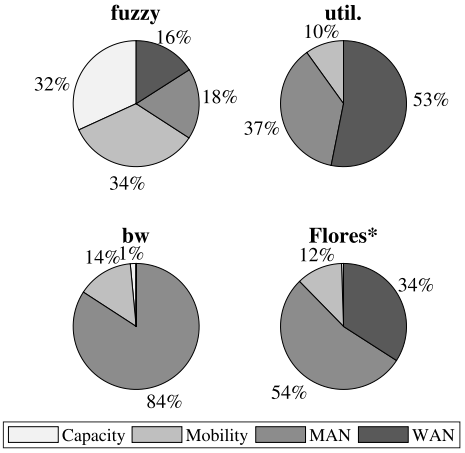
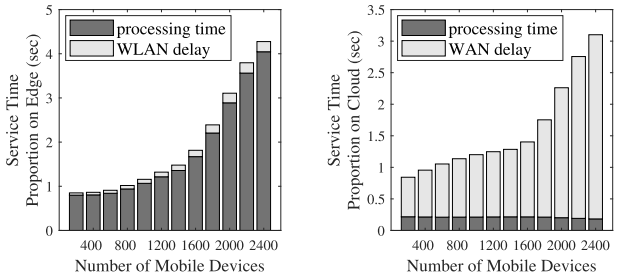


Fig. 11. Distribution of task failure reasons.



(a) Proportion of service time on edge (b) Proportion of service time on cloud

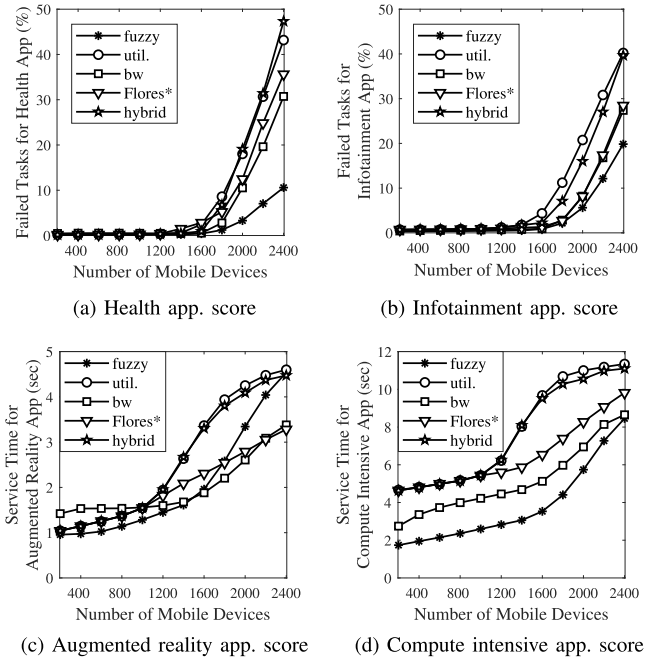
Fig. 12. Average service time values for fuzzy based workload orchestrator.

the MAN delay, it can adapt to the dynamic environment. As a result, it utilizes the edge servers more efficient than its competitors.

4) Task Failure Analysis: In our simulations, the tasks fail due to four reasons which are the mobility, lack of VM capacity on the edge servers, lack of MAN and the WAN capacity. In theory, the lack of WLAN capacity also causes task failures, but very few task losses due to the WLAN capacity are observed for the cases studied. The proportion of task failures for 2000 mobile devices is shown in Fig. 11. The failures due to the mobility is unavoidable since the task level handoff is not considered due to the real-time properties of the target applications. If the users change their location before getting the response of the previously requested tasks, related tasks fail because the users are not connected to the related WLAN anymore. So, for all the algorithms, we can observe somewhat a similar number of failed tasks due to the mobility in average. However, the ratio of this kind of failures may vary according to the other task loss results. First of all, the MAN communication failure is the common problem for all competitor algorithms since they do not take the MAN capacity into consideration while selecting a server in the edge layer. In the *Utilization-Based* approach, majority of the tasks drop because of the WAN congestion, hence it only considers the CPU utilization of the edge VMs and starts offloading tasks to the cloud server after the VMs are congested. On the other hand, in the *Bandwidth-Based* approach, most of the tasks drop due to the MAN capacity, because it only considers the WAN bandwidth and starts offloading tasks to the edge server after the WAN is congested. When it comes to the *Modified Flores-Based* approach, the impact of the MAN congestion on the task failures is more than that of the WAN congestion. Finally, in the *Fuzzy-Based* approach, the losses are more balanced than the other algorithms.

5) Service Time Analysis: The service time includes both processing delay and the communication delay. If the task is executed on the edge server, the total service time consists of the WLAN/MAN delay and the edge VM processing delay. On the other hand, if the task is offloaded to the cloud server, the total service time consists of the WAN delay and the cloud VM processing delay. In Fig. 12, service time score of the tasks are provided by considering both the processing delay and the communication delay for the *Fuzzy-Based* algorithm. The results for the tasks which are executed on the edge server is shown in Fig. 12(a). The dominant time period consumed on the edge server offloading process is the processing time. As expected the WLAN communication time is very low. As the load increases, both the processing and the networking time rises, but the increase rate of the processing time is much more than the networking time. The results for the tasks which are executed on the cloud server is shown in Fig. 12(b). In this case, the dominant time period spend on the cloud server offloading process is the WAN communication time. Since the VMs running on the cloud servers are very powerful in our scenario, the processing time is low. As the load increases, only the networking time increases, in fact, in the worst case it may take 5 to 6 seconds for the *Utilization-Based* algorithm.

6) Application Performance Analysis: Instead of the average results for all application types, we can separately investigate the performance of the algorithms for each application. In Fig. 13, the result of different applications are compared. In Fig. 13(a) and 13(b), the average failed task performance of the algorithms are evaluated by considering the health and the infotainment applications. As explained in Section V-B, the infotainment application generates big tasks (results in 10% CPU utilization on the corresponding edge VM) whereas the health application generates small tasks (results in 2% CPU utilization on the corresponding edge VM). In addition, the tasks generated by the health application have higher responsiveness level. According to the results, the *Fuzzy-Based* algorithm outperforms others while serving time-critical small tasks. We can say that our fuzzy based algorithm distinguishes the tasks by their size and responsiveness level better than its competitors. It also provides better performance for the big tasks. In Fig. 13(c) and 13(d), the average service time performance of the algorithms are evaluated by considering the augmented reality and the compute intensive applications. The augmented reality application is a delay intolerant application and it generates medium sized tasks (results in 6% CPU utilization on the corresponding edge VM). Whereas, the compute intensive application is a delay tolerant application and it generates very big tasks (results in 30% CPU utilization on the corresponding edge VM). Among the competitor algorithms, the *Bandwidth-Based* algorithm provides better service delay performance for the compute intensive application. It can be



(a) Health app. score (b) Infotainment app. score

(c) Augmented reality app. score (d) Compute intensive app. score

Fig. 13. Performance evaluation of the system for different application types.

said that executing the tasks generated by the compute intensive application mostly on the cloud servers provides a better performance. The *Fuzzy-Based* algorithm provides the lowest service time since it also uses the edge servers when they are lightly loaded. When it comes to the augmented reality application, the *Fuzzy-Based* approach provides better results when the number of mobile devices is low. If the system is very congested (e.g., 2400 devices are used), the *Fuzzy-Based* approach has the highest service time. The explanation for Fig. 10(b) is also valid for this case. If the system is heavily loaded, the *Fuzzy-Based* solution can execute more augmented reality application tasks than the competitors. As a result, the service time performance of the *Fuzzy-Based* approach becomes worse than the other algorithms. This situation is observed on the augmented reality application, because the number of tasks generated by this application is higher compared to the compute intensive application.

VI. Conclusion

Edge computing promises to take over the complex operations of the mobile devices which have limited processing storage, and battery capabilities. The main objective of this paradigm is to improve the responsiveness and battery lifetime of the mobile devices by using the nearby micro datacenters. The dynamic state of the network, mobile devices and edge servers brings some challenges which negatively affects the overall system performance. In order to overcome these challenges, the researchers propose the edge orchestrator concept which manages the edge resources to increase the performance. Since the demand of the mobile devices is not known in advance in such a dynamic environment, the orchestration problem cannot be solved via the offline optimization approaches. As a result, the resource management becomes a difficult online problem which should be solved by the edge orchestrator. In this study, a fuzzy logic based workload orchestrator is proposed due to its low complexity and efficiency in handling uncertain nonlinear systems by considering multiple criteria. Orchestrating in the multi-tier edge computing architecture addressed in this work is a new concept, hence a fuzzy logic based approach has not been studied so far. The mobile devices offload their tasks to the edge or cloud servers via WiFi access technologies and the orchestrator allocates the edge resources. We set up a simulation environment to evaluate the performance of our workload orchestrator design by comparing it with the classical workload orchestrator algorithms. According to the simulation outcomes, using a fuzzy logic based orchestrator provides better results compared to the other algorithms for the cases studied. As a future work, we plan to apply intelligent optimization techniques to improve our fuzzy membership functions and rules. In addition, we want to compare our FLS with other techniques such as using a reinforcement learning approach for workload orchestration [36]. Finally, it can be interesting to analyze the impact of the task migration among the edge/cloud servers.

References

[1] T. Taleb *et al.*, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

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

[3] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, 2013.

[4] "Fog computing and the Internet of Things: Extend the cloud to where the things are," San Jose, CA, USA, Cisco, White Paper, 2015.

[5] *Multi-Access Edge Computing*. Accessed: Oct. 15, 2017. [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing

[6] A. Ceselli, M. Premoli, and S. Secci, "Mobile edge cloud network design optimization," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1818–1831, Jun. 2017.

[7] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[8] D. Zhou *et al.*, "Integration of fuzzy CMAC and BELC networks for uncertain nonlinear system control," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, Jul. 2017, pp. 1–6.

[9] V. Salmani, R. Ensafi, N. Khatib-Astaneh, and M. Naghibzadeh, "A fuzzy-based multi-criteria scheduler for uniform multiprocessor real-time systems," in *Proc. 10th Int. Conf. Inf. Technol. (ICIT)*, Dec. 2007, pp. 179–184.

[10] Y. Tsuzaki and Y. Okabe, "Reactive configuration updating for intent-based networking," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2017, pp. 97–102.

[11] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of edge computing systems," in *Proc. 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, May 2017, pp. 39–44.

[12] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[13] A. Hegyi *et al.*, "Application orchestration in mobile edge cloud: Placing of IoT applications to the edge," in *Proc. IEEE 1st Int. Workshops Found. Appl. Self Syst. (FASW)*, Sep. 2016, pp. 230–235.

[14] V. Karagiannis and A. Papageorgiou, "Network-integrated edge computing orchestrator for application placement," in *Proc. 13th Int. Conf. Netw. Service Manag. (CNSM)*, Nov. 2017, pp. 1–5.

[15] D. Santoro, D. Zozin, D. Pizzolli, F. D. Pellegrini, and S. Cretti, "Foggy: A platform for workload orchestration in a fog computing environment," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2017, pp. 231–234.
[16] H. Flores *et al.*, "Large-scale offloading in the Internet of Things," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2017, pp. 479–484.
[17] A. C. Baktir, A. Ozgovde, and C. Ersoy, "Enabling service-centric networks for cloudlets using SDN," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manag. (IM)*, May 2017, pp. 344–352.
[18] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Comput.*, vol. 4, no. 2, pp. 26–35, Mar./Apr. 2017.
[19] V. D. Maio and I. Brandic, "First hop mobile offloading of DAG computations," in *Proc. 18th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput. (CCGRID)*, May 2018, pp. 83–92.
[20] H. Flores and S. Srirama, "Adaptive code offloading for mobile cloud applications: Exploiting fuzzy sets and evidence-based learning," in *Proc. 4th ACM Workshop Mobile Cloud Comput. Services*, 2013, pp. 9–16.
[21] S. M. Hosseini, M. Kazeminia, M. Mehrjoo, and S. M. Barakati, "Fuzzy logic based mobile data offloading," in *Proc. 23rd Iran. Conf. Elect. Eng.*, May 2015, pp. 397–401.
[22] S. Rathore, P. K. Sharma, A. K. Sangaiah, and J. J. Park, "A hesitant fuzzy based security approach for fog and mobile-edge computing," *IEEE Access*, vol. 6, pp. 688–701, 2018.
[23] S. A. Soleymani *et al.*, "A secure trust model based on fuzzy logic in vehicular ad hoc networks with fog computing," *IEEE Access*, vol. 5, pp. 15619–15629, 2017.
[24] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of MEC in the Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 84–91, Oct. 2016.
[25] K. Imagane, K. Kanai, J. Katto, T. Tsuda, and H. Nakazato, "Performance evaluations of multimedia service function chaining in edge clouds," in *Proc. 15th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2018, pp. 1–4.
[26] S. Ghosh, Q. Razouqi, H. J. Schumacher, and A. Celmins, "A survey of recent advances in fuzzy logic in telecommunications networks and new challenges," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 3, pp. 443–447, Aug. 1998.
[27] L. Abdullah, "Fuzzy multi criteria decision making and its applications: A brief review of category," *Procedia Soc. Behav. Sci.*, vol. 97, pp. 131–136, Nov. 2013.
[28] J. M. Mendel, "Fuzzy logic systems for engineering: A tutorial," *Proc. IEEE*, vol. 83, no. 3, pp. 345–377, Mar. 1995.
[29] M. Mokarram, A. Khoei, K. Hadidi, and K. Gheysari, "Implementation of centroid defuzzifier block using CMOS circuits," in *Proc. 4th Int. IEEE Conf. Intell. Syst.*, vol. 1, Sep. 2008, pp. 2–29.
[30] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exp.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.
[31] M. Silva *et al.*, "Glassist: Using augmented reality on Google glass as an aid to classroom management," in *Proc. XVI Symp. Virtual Augmented Reality*, May 2014, pp. 37–44.
[32] J. Guo, B. Song, Y. He, F. R. Yu, and M. Sookhak, "A survey on compressed sensing in vehicular infotainment systems," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2662–2680, 4th Quart., 2017.
[33] C. Tunca *et al.*, "Inertial sensor-based robust gait analysis in non-hospital settings for neurological disorders," *Sensors*, vol. 17, no. 4, p. E825, 2017.
[34] S. Kamouskos, "Supporting nomadic users within virtual private networks," in *Proc. IEEE Service Portability Virtual Custom. Environ.*, 2000, pp. 128–133.
[35] C. Sonmez, A. Ozgovde, and C. Ersoy, "Performance evaluation of single-tier and two-tier cloudlet assisted applications," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2017, pp. 302–307.
[36] Y. Martínez, A. Nowé, J. Suárez, and R. Bello, "A reinforcement learning approach for the flexible job shop scheduling problem," in *Proc. 5th Int. Conf. Learn. Intell. Optim. (LION)*, Berlin, Germany: Springer-Verlag, 2011, pp. 253–262.

Cagatay Sonmez received the B.S. degree in computer engineering from Dokuz Eylül University, İzmir, in 2008, and the M.S. degree in computer engineering from Bogazici University, Istanbul, in 2012, where he is currently pursuing the Ph.D. degree with the Department of Computer Engineering. He has been with Arcelik Electronics as a Group Leader in Research and Development Software Department, Istanbul, since 2008. His research interests include design and performance evaluation of communication protocols, mobile cloud computing, fog computing, and IoT.

Atay Ozgovde received the B.S. and M.S. degrees from Bogazici University, Istanbul, in 1995 and 1998, respectively. He was with Nortel Networks as a Research and Development Engineer in various telecommunications projects from 1998 to 2001. In 2002, he was a Research Assistant with the Computer Engineering Department, Bogazici University. He is currently an Assistant Professor with the Computer Engineering Department, Galatasaray University. His research interests include wireless sensor networks, embedded systems, distributed systems, pervasive computing, SDN, and mobile cloud computing.

Cem Ersoy received the Ph.D. degree from Polytechnic University, New York, in 1992. He became a Professor of computer engineering with Bogazici University. He was a Research and Development Engineer with NETAS A.S. from 1984 to 1986. His research interests include wireless/cellular/ad-hoc/sensor networks, activity recognition and ambient intelligence for pervasive health applications, green 5G and beyond networks, mobile cloud/edge/fog computing, software-defined networking, and infrastructureless communications for disaster management.