A Scalable Edge Computing Architecture enabling Smart Offloading for Location Based Services

Dimitrios Spatharakis[a,*], Ioannis Dimolitsas[a], Dimitrios Dechouniotis[a], George Papathanail[b], Ioakeim Fotoglou[b], Panagiotis Papadimitriou[b], Symeon Papavassiliou[a]

[a]*School of Electrical and Computer Engineering, National Technical University of Athens, GR 157 80, Greece*
[b]*Department of Applied Informatics, University of Macedonia, GR 546 36, Greece*

Abstract

The evolution of Location Based Services (LBS) is expected to play a significant role in the future smart city. The ever-increasing amount of data produced, along with the emergence of next-generation computationally intensive applications, requires new service delivery models. Such models should capitalize on the Edge Computing (EC) paradigm for supporting the data offloading process, by considering user's contextual information in the offloading decision along with the infrastructure resource allocation operations, towards meeting the stringent performance specifications. In this article, a two-level Edge Computing architecture is proposed to offer computing resources for the remote execution of an LBS. At the Device layer, an initial offloading decision is performed taking into consideration the estimated position and quality of wireless connection of each user. At the Edge layer, a resource profiling mechanism maps the incoming workload to EC computing resources under specific performance requirements of the LBS. Dealing with the dynamic workload, a scaling mechanism simultaneously takes the offloading decision and allocates only the necessary resources based on the

*Corresponding Author
Email address: dspatharakis@netmode.ntua.gr (Dimitrios Spatharakis), jdimol@netmode.ntua.gr (Ioannis Dimolitsas), ddechou@netmode.ntua.gr (Dimitrios Dechouniotis), papathanail@uom.edu.gr (George Papathanail), fotoglou@uom.edu.gr (Ioakeim Fotoglou), papadimitriou@uom.edu.gr (Panagiotis Papadimitriou), papavass@mail.ntua.gr (Symeon Papavassiliou)

resource profiles and the estimation of a workload prediction technique. For the evaluation of the proposed architecture, a smart touristic application scenario was realized on a real large-scale 5G testbed, following the principles of Network Function Virtualization (NFV) orchestration. The experimental results indicate the high accuracy of the localization technique, the success of the two-stage offloading decision and the scaling mechanism, while meeting the performance requirements of the LBS.

Keywords: Location Based Services, Edge Computing, Resource Scaling, Offloading Decision, NFV orchestration

1. Introduction

The proliferation of fifth-generation (5G) networks, the ever-expanding need for mobile or cloud applications and the Internet of Things (IoT), brings a new era in Information and Communication Technology (ICT). As studies show [1], [2], in a few years, one of the most anticipated challenges will be the immense growth in data produced and devices connected, since the majority of the devices (sensors, wearables, etc.) will be interconnected. Moreover, new types of applications, in the context of massive Machine Type Communications (mMTC) [3], will be available to support every aspect of everyday life, significantly affecting the way we perceive the world.

This remarkable increase of the connected devices and sensors has made the vision for living in a smart city environment possible [4]. However, for this vision to become reality, new demands arise from these large-scale infrastructures, such as the ability to manage and orchestrate the massive amounts of data and devices, while enabling automated instantiation and communication of the corresponding services. NFV [5] and Software-Defined Networking (SDN) [6] are the key enablers of the prominent 5G infrastructure facilitating the decoupling of software deployment from hardware. Enabling 5G network slicing, NFV and SDN automatically orchestrate, instantiate and manage the computing and network resources of virtual networks over a single physical infrastructure.

Nowadays, smart applications are mainly based on portable devices. Thus, the location-awareness of people and devices is one key ingredient for provisioning these services, and enabling their interaction with the surroundings. In the last years, Location Based Services (LBS) are emerging applications that rely on information about one's exact position. The estimation of the position of a target (or user) in an outdoor environment is trivially determined by the Global Positioning System (GPS) or the cellular-based localization approaches [7], while location awareness is more challenging in the case of indoor environments. Therefore, the service providers must select the appropriate localization methodology considering the number of the mobile users and the accuracy and cost requirements of the application.

In this new era of ICT, cloud computing evolves to the emerging models of Fog Computing [8] and Multi-access (or Mobile) Edge Computing [9], which focus on handling the compelling demand for computational resources along with time-sensitive or location-aware applications. Augmenting the cellular or Wi-Fi networks with processing capabilities aims at reducing the network latency for mobile users and optimizing the core networks and the utilized cloud resources. For the rest of the article, the term of Edge Computing (EC) is used to refer to both Fog Computing and MEC. The EC architecture enables the offloading of users' computational-intensive tasks, *e.g.,* object recognition, in order to reduce energy consumption of their mobile devices and meet the Quality of Service (QoS) requirements of the application. Contrary to the cloud data centers, an edge data center has limited computing and storage resources. Hence, a smart offloading decision heavily depends on and should be accompanied by dynamic resource allocation and admission control mechanisms. The latter are responsible for the optimal utilization of the edge resources, and the distribution of offloaded requests among them.

In this work, we develop a two-layer architecture to support smart offloading of a location based service from IoT or mobile devices to an EC infrastructure. The proposed architecture has been particularly designed to accelerate the execution of an object identification service for mobile users. However, it is generic

and applicable to several types of EC environments and smart city applications. In the particular smart city scenario under consideration in this work, the visitors of a crowded touristic area use their camera-equipped devices to take snapshots or short-duration videos of a Point of Interest (PoI) (*e.g.,* exhibits of a museum) in order to receive additional information about them. Since image recognition is a computational-intensive and energy-consuming task, a cluster of proximate edge servers offer the essential computing resources to meet the strict time requirements of massive users. Furthermore, a localization technique is realized and evaluated. We assume that users are active in a crowded touristic area; hence, the shared bandwidth and foremost the interference of signals plays an important role in the overhead of transmission of the offloaded requests. As a result, to address this dynamic behavior, the offloading decision of a users' request takes into consideration the users' position in addition to the available resources of the edge servers at each time. The main contributions of this work are summarized, as follows:

- A resource profiling mechanism interprets the performance requirements of the application to computing resources on the EC side. This mechanism enables the modular design and allocation of the EC resources in order to meet the varying offloading demand. The resource profiling of the LBS is based on linear models from System Theory, which effectively capture the system dynamics of the service, and determines the essential resources for serving various amounts of offloaded requests.

- A two-step mechanism to support and realize the offloading decision is devised, considering both users' contextual information and the availability of the edge resources. Towards this direction, the offloading decision becomes a key-enabler for maintaining high performance results, benefiting both the infrastructure provider and the users.

- An Edge Computing architecture to support the deployment, orchestration and management of LBS is introduced. Following the principles of 5G directives, an NFV-based deployment of the architecture is presented,

using state-of-the-art software tools for the orchestration and management of the infrastructure.

- A real-world evaluation over an open 5G infrastructure is conducted to verify the validity and applicability of the proposed methodology. Also, the proposed resource profiling mechanism and the workload estimation approach are compared against well-established counterpart methodologies in the literature [10, 11].

The rest of the article is organized as follows. In Section 2, the current state of the art is presented, while in Section 3, the proposed architecture is described and analyzed. In Section 4, the system orchestration is presented and details about its realization are highlighted. The evaluation of the proposed architecture and the localization technique are performed and presented in Section 5. Finally, Section 6 concludes the paper and presents our future plans.

2. Related Work

This section present the most relative studies of the literature to our proposed EC solution for LBS. The following studies are classified based on three interdependent pillars; (i) Edge Computing, (ii) Computation Offloading, and (iii) Location Estimation.

(i) Edge Computing

Puliafito et al. [12] presented a detailed survey for Fog Computing platforms for IoT-based applications. In [13], a network slicing orchestration system was proposed for federated data centers. The orchestration is based on Open Source MANO [14], which is aligned with ETSI NFV standard. Based on G/G/m queuing model, a heuristic scaling algorithm regulated the mean overall processing time of the incoming slice orchestration requests. However, no further details are given on how the network slice placement problem is addressed. MESON [15] focuses on communication between slices deployed on edge data centers.

The cross-slice communication aims to alleviate the congestion of the core network, minimize the utilization of cloud resources and meet the requirement of time-sensitive LBS.

Leivadeas et al. [16] proposed a meta-heuristic approach for the placement of network slices in cloud and edge infrastructure aiming at the minimization of the deployment cost and the end-to-end network delay. This approach considered that the resource demands of the slices are static and it can be applied only for their initial placement. Sonmez et al. [17] proposed a fuzzy workload orchestrator for various EC scenarios. For each offloaded request, a set of fuzzy rules decided the destination computational unit within an hierarchical EC architecture. However, the authors empirically defined the fuzzy rules sets, which might not be applicable for services with different workload characteristics. In [18], the authors addressed jointly the problems of network selection and service placement for MEC infrastructure. To reduce the complexity of the general problem, they decomposed it into a series of sub-problems and solved them in an iterative fashion. However, the proposed QoS model included only network parameters ignoring the processing delay of the service.

Zenith et al. [19] proposed resource sharing contracts between edge infrastructure providers and service providers. Specifically, an auction-based contract establishment and resource allocation mechanism was introduced, focusing on ensuring the utility-maximization for both entities and the latency constraints. However, the authors did not provide any details on how the tasks were distributed among the containers of the established contracts. Wang et.al. [20] studied the problem of VM placement along with the workload assignment for mobile cloud applications in MEC. An MILP method was designed to minimize hardware consumption in order to deploy VMs, while satisfying diverse latency requirements of different applications. In [21], the authors proposed a novel solution to determine both the optimal number of the VMs to spawn, as well as their placement. For VM spawning, a Mixed Integer Linear Program (MILP) was formulated, while a game theoretic approach was employed (i.e. Coalition Formation Games), in order to treat the latter problem.

(ii) Computation Offloading

The offloading decision is critical for guaranteeing the high QoS of LBS. In the literature, most of the proposed studies focus on a single or group of performance criteria, i.e., latency or energy, to decide whether the offloading is profitable or not. This paragraph illustrates the most recent relative studies to our work. The interested reader can refer to [22] for a comprehensive detailed review of older studies on computation offloading. In [23] a mobility-aware offloading mechanism (referred to as MAGA) was proposed, based on frequent moving patterns of the users and a genetic algorithm. MAGA mechanism focused on partial task offloading to EC servers. However, the dynamic resource allocation of these resources is not considered. Tang et al. [24] and Apostolopoulos et al. [25] proposed the adoption of Prospect Theory to design user behavior-aware MEC offloading frameworks. In both works, the communication and the computation models for the users were provided and Prospect Theory-based offloading games are formulated taking into account, user risks, as well as weighting and framing effects. In [26], the authors proposed a task offloading framework for Software Defined Ultra-Dense Networks that focused on the reduction of the task duration and the energy saving. The offloading problem was split in two sub-problems;(i) task placement problem and (ii) resource allocation. Lyu et al. [27] presented a collaborative Cloud-MEC-IoT architecture and proposed a request modelling scheme and an admission control framework to address the scalability problem of these platforms. Although the authors considered heterogeneous edge resources, the computation model was not dynamic.

(iii) Location Estimation

In the context of indoor localization, many approaches can be found in the literature that addresses different scenarios and accuracy requirements. In [28], the authors proposed a mechanism that takes advantage of the information by the WiFi signals and calculates the position of a moving user with the fingerprinting technique. The mean estimation error of this mechanism is about

$0.85m$, showing the advantages of the weighted K-Nearest Neighbor algorithm. Kang et al. [29] proposed a method to track pedestrian movement in an indoor environment under the existence of sensing inaccuracy. Similar to our work, they used a smartphone equipped with accelerometer, magnetometer, and gyroscope sensors and exploited the characteristics of human motion to estimate the position of a user. The proposed method reduces the influence of the cumulative errors of the dead reckoning technique. In [30], exploiting Machine Learning techniques, a tracking system was proposed, estimating again the position of a user, who moves in an indoor place and interacts with a wireless sensor network. The classification of speed and heading of the trajectory of the pedestrian was implemented offline supervised by Neural Networks. A different approach to handle the challenges of indoor localization in a robotic scenario in the Industry 4.0 context was presented in [31]. The single vision-based method is landmark assisted, exploiting natural features of the landmarks and the core principles of projective geometry, for the self-localization of an indoor autonomous mobile robot agent. The results of the proposed algorithm showed a significant accuracy in close range. However, as most of the vision-based methodologies, this method is heavily dependent on ambient conditions (*e.g.,* light conditions). In a similar vein, authors in [32], realized a novel unsupervised learning framework requiring video frames, to extract pose estimation and introduce a Simultaneous Localization and Mapping (SLAM) system, that could be also applicable in an indoor environment.

Most of the aforementioned studies usually proposed solutions addressing only one of the above research problems in isolation, ignoring the rest dominant parameters, thus failing to allow the design of a holistic solution that achieves both high performance of LBS and optimal utilization of EC resources. To exactly fill this gap, in this article we propose an EC architecture for LBS that considers the user's contextual information on the offloading decision and the resource allocation of the edge resources. Towards this direction, the proposed framework can meet the performance requirements of both LBS and EC infrastructure.

3. System Architecture

As mentioned before, in the smart city scenario under consideration in this article, visitors use an LBS to retrieve information about PoIs in a crowded touristic area. The LBS can be executed either on the user's mobile device or on the EC servers. Accordingly, the proposed EC architecture consists of two layers: namely the Device layer and the Edge layer.

At the Device layer, portable IoT devices, *e.g.,* Raspberry Pis and mobile phones, equipped with sensors and camera modules provide location information and media content in order to identify PoIs (*e.g.,* museum exhibits) and retrieve information about them through an image recognition service. Since image processing is a computationally-intensive task, the users forward their requests to a cluster of edge servers for further processing in order to save energy and get the response in a timely manner. At the Edge layer, a control mechanism is responsible for scaling the resources of the instances of the virtualized image processing service and distributing the admitted requests among them. However, as specific may the above scenario seems, as generic and versatile the proposed architecture is, and it can be applied in any smart city application. To further elaborate on the generic essence of this work, we have identified the following use-case requirements and challenges.

- Every smart city application has specific dominant parameters that affect its performance. Thus, it is essential to build a profile of the application to include all important features. In this work, we focus on the user's location and propose a generic resource profiling of the application. The profiling mechanism provides accurate performance modeling under varying conditions and application requirements and facilitates the design of effective resource allocation strategies. It is an interpretation mechanism that translates the application's QoS requirements to EC computing resources and facilitate the interoperability in the case of heterogeneous resources.

- A successful offloading decision of mMTC applications must consider both

the user's contextual information and the availability of EC resources. With this capacity, the user should decide whether the offloading is beneficial based on the location and the network/communication parameters (*e.g.,* signal strength). Especially in crowded places, communication related aspects such as the transmission time of a request, may present a significant overhead in the overall response time. Additionally, for applications of multiple users, the EC infrastructure provider should accurately estimate the dynamic workload demand and admit the number of requests that can be served with a guaranteed level of performance. Hence, we propose a two-step offloading decision that takes into account both the user and the provider's operating conditions and perspectives.

- The focus of this work is placed on the challenging scenarios where the edge computing infrastructure is receiving a great amount of incoming workload of offloaded tasks. This workload also fluctuates during the day, for example following the traffic of visitors at the place of interest. Hence, the scalability of EC infrastructure is a prerequisite for meeting the performance requirements and the optimal resource utilization. Our proposed dynamic resource allocation mechanism leverages the application's resource profile and allocates only the necessary resources in order to maximize the number of offloaded requests with response time guarantee.

In this study, the response time is defined as the sum of the transmission and processing time to serve an offloaded request. Figure 1 illustrates an overview of all levels of the proposed architecture. At the bottom level, the mobile device estimates the location of the user and originally decides the local or remote execution of the generated request. At the upper level, the Controller component realizes the intelligent functionalities of the proposed architecture, which includes the resource profiling, the workload prediction and the scaling mechanism along with the necessary monitoring service. Furthermore, the Virtual Machines (VMs) with specific computing resources host the image recognition

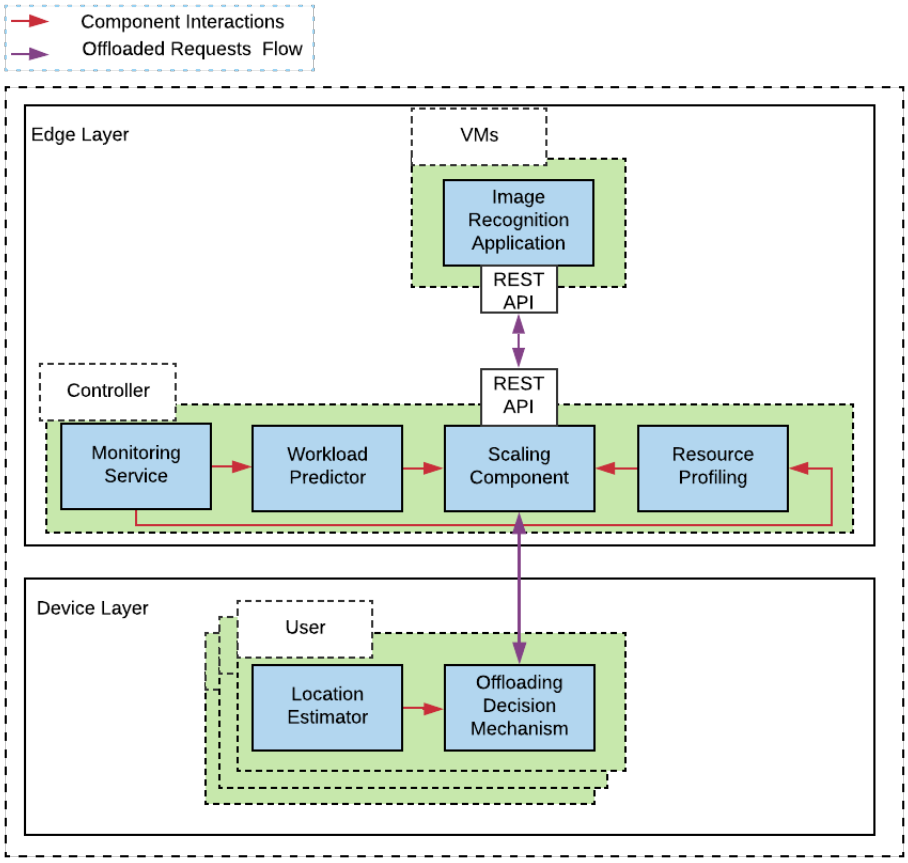


Figure 1: An overview of the EC architecture for LBS.

service. The proposed application is considered to be stateless with stringent performance requirements. In the following sections, the individual components of the architecture are described in more detail along with the intelligence of the mechanisms provided at each layer.

3.1. Device Layer

At the Device layer, two fundamental mechanisms co-exist, providing the user's contextual information and the initial offloading decision.

3.1.1. Location Estimator

To estimate the user's position, a step detection algorithm has been developed. The users' motion is measured by inertial measurement unit (IMU) sensors and the estimation is performed by dead reckoning technique. This mechanism is classified in the category of Inertial based Systems, where there is no need for external information by the physical infrastructure, and the position of the user is estimated relative to a known starting point. This location estimation method is applicable in any end-device equipped with the appropriate IMU sensor. As thoroughly examined in [33], a step detection algorithm is applied using an accelerometer based on the mechanics of human motion, while the heading of the user is calculated by a gyroscope sensor. Consequently, given a starting position, the step and heading mechanism grants users the capability to locate themselves in an indoor environment and share this information with the infrastructure for an optimal offloading decision. Hence, in the scope of this work, the Location Estimator plays a significant role in the workflow of the offloaded requests, as shown in the next subsection.

3.1.2. Offloading Decision Mechanism

One of the main objectives of this work is the development of a smart dynamic offloading mechanism that is based on contextual information and aims to satisfy both the QoS requirements of the users and optimal utilization of resources of the underlying infrastructure. Therefore, we propose a two-step offloading mechanism, involving both the Device and Edge layer of the architecture. More specifically, whenever a new request is generated, the offloading decision is based on two parameters; (a) the position of the user, and (b) the current measurement of the wireless signal strength. On the one hand, we assume that the signal coverage is fluctuating in the territory of this scenario. As a result, one's position affects the quality of the wireless connection. Moreover, to facilitate automated offloading, the touristic area is divided into sections where offloading is permitted based on offline metrics about the quality of the connection. Subsequently, a map is constructed based on these metrics, hereinafter

referred to as "offloading map". On the other hand, in crowded areas, signal interference is high due to the shared medium of the wireless connection among the users. Considering that the transmission power of the antenna is bounded, the shared bandwidth results in a maximum number of users, who can offload their requests with satisfying transmission rate. Hence, the measured signal strength is adopted as an indication of the quality of the wireless transmission rate, which heavily affects the performance of the system.

At the Device Layer, we assume a predefined signal strength threshold that the user must satisfy in order to offload a request. If both the position and signal strength constraints are met, then the device offloads its request for further processing at the Edge layer, where the final offloading decision is taken. As a result, the user's offloading decision is performed in both a proactive and reactive manner, exploiting contextual information of the user's status in terms of position and connection quality. In case of a rejected request during this offloading decision phase, the image recognition service is executed locally on the mobile device.

3.2. Edge Layer

The workload generated by the mobile users is directed to the Controller of the Edge layer through a Wireless Access Point. As mentioned previously, the Controller is responsible for the allocation of resources to the deployed VMs, which incorporates the second step of the offloading decision, and the load balancing of the requests to the VMs. Initially, the monitoring service collects the necessary data for the incoming requests, the performance of running VMs and the utilization of the EC resources. The monitoring data is used by the rest components of the Controller. In particular, the Resource Profiling component provides an offline training tool based on System Theory to extract specific resource profiles of running VMs. A resource profile interprets the QoS requirements of the service (*i.e.,* response time) to EC computing resources (*i.e.,* vCPUs, memory). These profiles enable the Scaling Component (SC) of our approach. To better manage and orchestrate the EC infrastructure, time is di-

vided into discrete time intervals. At the beginning of a time interval, based on Kalman Filtering approach and monitoring data, the Workload Predictor provides an estimation of the expected number of requests within the next time interval. This estimation and the computed resource profiles are used by the SC to determine the number and resources of running VMs that host the object recognition application. Hence, to guarantee the QoS of the the users, the SC accepts only up to a limit of incoming requests at each interval, derived from the profiles of the operating VMs, while rejecting those exceeding it. Thus, the SC takes the final offloading decision considering the overall performance of the EC infrastructure. Leveraging this mechanism, the SC assures that for each request that is accepted for remote execution, the users' QoS criteria are met, and for the remaining requests, users execute the object-recognition service locally on their devices. Last but not least, within each time interval, the SC distributes each accepted request to the corresponding operating VMs, via a REST-API operating at the latter. The activated VMs host the image recognition-based applications, which are trained off-line and identify the candidate PoIs.

3.2.1. Resource Profiling Component

In general, the resource profiling mechanism quantifies the relation between the application's requirements, the incoming requests and the computing resources at the EC side. Towards this direction, the Resource Profiling component build various resource profiles of an application that correspond to different operating conditions. In our scenario, the image processing service is CPU-intensive, thus, for the rest of the paper, the resource the profiling mechanism refers to vCPUs. To extract the resource profiles, we adopt switching linear systems that can describe the performance of an application under different operational conditions. For different numbers of allocated vCPUs to a VM and varying incoming requests, we define a discrete linear system of the following form,

$$x(t+1) = \alpha x(t) + \beta u(t)$$

where $x(t)$, named as state variable, is the average response time in time interval t and $u(t)$, named as input variable, is the admitted requests in time interval t. In such a way, we describe the performance of the image recognition service under dynamic workload conditions. The Recursive Least Square algorithm [34] is exploited to calculate an estimation for the parameters α and β. The main objective of the Resource Profiling component is to maximize the offloaded requests and optimally allocate them the available computing resources. Towards this objective, for each linear system and given the desired average response time x_e, we extract feasible resource profiles, which maximizes the admitted requests and satisfies input constraints, by solving the following optimization problem,

$$\max_{x_m, x_M, x_e, u_m, u_M} \quad u_e \tag{1a}$$

subject to

$$x_e = a x_e + b u_e \tag{1b}$$

$$x_m \leq x_e \leq x_M \tag{1c}$$

$$u_m \leq u_e \leq u_M. \tag{1d}$$

First of all, the resource profile must be an equilibrium point of the linear system as defined by the first constraint. The second constraint implies that there is a minimum (u_m) and the maximum value (u_M) for the offloaded requests. Similarly, the last constraint refers to the state variables and means that the average response time varies between the minimum (x_m) and the maximum values (x_M). In such a way, the resource profiles actually translate the QoS requirements of the object recognition-based service to EC computing resources. Within the scope of this work, the resource profiles actually correspond to a VM with specific computing resources that serves a predefined number (u_e) of offloaded requests with a predefined average response time per interval (x_e). The

value of x_e can be selected as a percentage of the maximum acceptable response time, which is implied by a Service Level Agreement (SLA) and it represents the strictness of the underlying QoS constraints. In our use case, the average response time of the resource profiles are set for demonstration only purposes, at the half of the maximum acceptable response time by the user.

The benefits of above resource profiles' design are manifold. Initially, it is a generic process that is applicable for any type of IoT-based application and edge/cloud infrastructure. The utilization of switched linear systems allows to model any operating condition and the design of the resource profiles according to the specific QoS requirements of the LBS. Contrary to queue models [11, 35] that are valid only in steady state, the linear systems can capture transient phenomena and do not provide only static information such as mean service and arrival rate. Additionally, although out of the scope of this article, the linear systems enable the design of feedback controllers, which can guarantee valuable system properties such as stability and reachability apart from the satisfaction of the QoS constraints. The latter cannot be accomplished with queue models. Finally, resource profiles enables the interoperability of heterogeneous infrastructure. In the case of edge-to-edge or edge-to-cloud collaboration, the computation of resource profiles for the different types of hardware simplifies the load distribution between sites, because the number of exchanged requests can be automatically translated into resources of each site through the resource profiles.

3.2.2. Workload Predictor

Due to the mobility of the users, the amount of offloaded requests changes significantly. In order to enable optimal scaling and resource allocation, an accurate workload prediction methodology is necessary to estimate future incoming requests. For this purpose, we adopt Kalman Filtering [36], which is a well-known estimation methodology for dynamic systems. We further assume that the offloaded requests can be modeled as a system with process and mea-

surement uncertainties, as follows,

$$L(t+1) = L(t) + w(t) \tag{2}$$

$$Z(t) = L(t) + v(t) \tag{3}$$

where L is the number of offloaded requests of the image recognition service at time interval t. At interval $t+1$, the value of the workload is defined as the sum of the current value and the process noise w, which models the generation of requests by the users. The covariance matrix of the process noise is defined as Q. The variable Z denotes the measurement of the offloaded request and equals the actual number of requests L plus a measurement noise value $v(t)$, which follows a normal distribution with zero mean, while R is its covariance matrix. The estimation of the request at the next time interval $\hat{L}^-$ can be computed by applying the following Kalman filter equations.

$$\hat{L}^-(k) = \hat{L}(k-1) \tag{4a}$$

$$P^-(k) = P^-(k-1) + Q \tag{4b}$$

$$K(k) = \frac{P^-(k)}{P^-(k) + R} \tag{4c}$$

$$\hat{L}(k) = \hat{L}^-(k) + K(k)(Z(k) - \hat{L}^-(k)) \tag{4d}$$

$$P(k) = (1 - K(k))P^-(k) \tag{4e}$$

The term $\hat{L}^-$ is the update state value based on the measured value of Z and the extrapolated state value of the previous step. The term K is called Kalman gain and is a number between zero and one which expresses the importance of the measurement. The term P is the error covariance. In every step, we use the first equation to estimate the forthcoming requests for the next interval based on the previous extrapolated values. The estimation of the incoming request is used by the SC for the scaling decision.

3.2.3. Scaling Component

Based on the resource profiles of the application and the estimation of the offloaded requests, the SC aims to meet the dynamic workload demands and

avoid over- or under-provisioning of EC resources. We formulate a mixed integer linear optimization problem (MILP), which minimizes the amount of the allocated EC resources. We define n binary variables $v_i, i = 1, \ldots, n$ that correspond to the n different resource profiles of a VM. Furthermore, for each v_i, we define the weight $w_i, i = 1, \ldots, n$, which is proportional to the number of the vCPUs, v, dedicated to a specific VM flavor. The optimization problem is formulated as follows:

$$\min_{v_i}\{\sum_{i=1}^{n}(w_i v_i)\} \tag{5a}$$

subject to:

$$\sum_{i=1}^{n}(v_i l_i) \geqslant \hat{L}^{-},\ i = 1, \ldots, n \tag{5b}$$

$$v_i = 0, 1,\ \forall i \in [0, \ldots, n], \tag{5c}$$

$$\sum_{i=1}^{n} v_i \leq TV \tag{5d}$$

where the first constraint indicates that the estimated offloaded requests $\hat{L}^{-}$ must be served by the activated VMs in the following time interval. The second constraint indicates that a VM with specific flavor can be either running or closed. This constraint implies that multiple instances of a VM with specific resource profile are not allowed in the same server. The last constraints guarantees that the resources of the running VMs do not exceed the total amount of available EC resources (TV) in terms of vCPUs. As a result, solving this optimization problem at each time interval, the SC ensures that the predicted workload will be served based on the corresponding resource demands with the minimum allocated resources. Nevertheless the overall cost of the flavors to be implemented is minimized. It is worth mentioning that the resource profiling mechanism allows to solve the problem for both cases of homogeneous and heterogeneous resources.

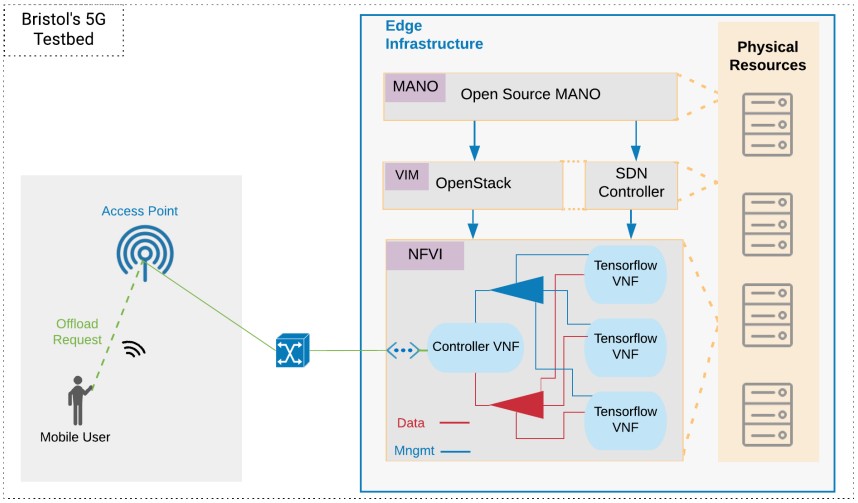


Figure 2: System Orchestration overview.

4. System Orchestration and Implementation

The aforementioned architecture was implemented in Bristol's University 5G testbed [37], which provides NFV orchestration and management solutions, bearing in mind that a real-world experience evaluation is of imperative importance when testing a system architecture. The System Orchestration overview is illustrated in Figure 2. Open Source MANO[1] (OSM) operates as an NFV management and orchestration (MANO) tool, in the context of the NFV MANO reference architecture, as defined by the ETSI NFV Standard. Thus, independently of the underlying Virtual Infrastructure Managers (VIM), OSM allows performing resource and service orchestration. In the proposed orchestration mechanism, OSM interacts with Openstack[2], which operates as the VIM. Openstack is a well-known open-source software for realizing Infrastructure as a Service (IaaS), structured by a stack of software tools used to build and man-

[1] https://osm.etsi.org/
[2] https://www.openstack.org/

age cloud computing platforms for public and private clouds. Moreover, it is responsible for controlling and managing the Network Function Virtualized Infrastructure (NFVI) resources.

The proposed system orchestration has a twofold aim; to meet the system architecture requirements, and to provide a scalable virtualization solution, enabling the basic aspects of a next-generation LBS-enabled architecture. Hence, the respective services are deployed as Virtual Network Functions (VNFs). Subsequently, two different VNFs have been deployed; namely the Controller, and the LBS-enabled object recognition application. The Controller VNF implements the intelligence mechanisms of the proposed Edge architecture as thoroughly presented in Section 3.2, while the Object Recognition application is used for the identification of PoIs in the Bristol's Millennium Square. Both VNFs are managed and orchestrated by OSM as depicted in Figure 2. Similar to the system's architecture, the NFV implementation follows a top-down design. At the top layer lies the Controller VNF, which dictates the decisions needed for the scaling of the operating application-VNFs and the load balancing of the incoming workload, while, at the bottom layer, three VNFs are facilitating with the object recognition application. Moreover, the Network Service (NS) of the architecture is defined, to deploy an automated service chain between those VNFs. Furthermore, the NS is broken down into two parts, namely; data and management network, for the connection of VNFs, following the standards of NFV and SDN .

In this section, we present the necessary procedure for the realization of the system orchestration. Initially, we created the appropriate system images comprised of an operating system and the corresponding services. The functionalities of the Controller are deployed as the first system image, while the second one involves the object recognition service alongside with the REST-API service. These two images have been onboarded in the Openstack of Bristol's 5G testbed. Furthermore, the appropriate VNF Descriptor (VNFD) files for both images are realized. These descriptors contain all the necessary information for the VIM to instantiate Virtual Deployment Units (VDUs), which are the VMs

that host the network functions. To be more specific, these descriptors define the flavors of virtual resources (vCPUs, RAM and storage) and virtual network connection points between the VDUs. Similarly, the Network Service Descriptor (NSD) specifies the Network Service (NS), which is the top-level structure that includes all the VNFs and implements the network topology of the experiment, associating respectively the connection points defined in VNFDs. Those descriptors have to be onboarded in OSM too. After this onboarding phase, OSM instantiates the NS and consequently the VNFs. Additionally, OSM dictates Openstack to realize the corresponding VDUs and the virtual network of the experiment. In our case, four VDUs are instantiated in Bristol's Openstack compute node. As explained in Section 3.2.1, three VDUs of different flavors were deployed hosting the object recognition service. Based on them, the SC decides at the beginning of each interval the running VDUs and directs the incoming traffic of offloaded requests accordingly, solving the optimization problem in Section 3.2.3.

5. Experimental Evaluation

In this section, the experimental set-up and corresponding results of the proposed EC framework realization and deployment are presented and discussed in detail. The proposed framework was deployed in a real large-scale experimental facility of University of Bristol 5G testbed [37], that exploited OpenStack and OSM for NFV orchestration. To highlight the validity of the proposed location-aware EC architecture, a smart touristic application for crowded indoor areas was deployed. The considered experimental set-up provided realistic conditions to evaluate the proposed EC architecture for LBS from the perspective of: (i) location estimation accuracy, (ii) success of the offloading decision and the application performance, and (iii) effectiveness of the scaling mechanism, (iv) a comparative evaluation with well established studies [10, 11]. Before proceeding with the illustration and analysis of the the obtained results in terms of the aforementioned directions, a detailed description of the experimental set-up and

considered scenario is provided.

5.1. Experimental Set-up and Scenario

In the experimental scenario under consideration, the visitors of Millennium Square of Bristol University testbed were scattered in a crowded touristic area, where several PoIs exist. Leveraging an object-recognition service, the visitors were able to retrieve interesting information about these PoIs. Google-powered TensorFlow [38] framework was deployed as the object-recognition service. Tensorflow is an open-source machine learning platform that can be easily retrained to classify the PoIs. For the retraining phase, a data set containing real pictures of the PoIs was used to build an object detection model for our LBS. At the Device layer, the visitors were emulated by Raspberry Pi devices equipped with the IMU sensor SparkFun MPU-9250 IMU Breakout[3], which contains a 3-axis gyroscope, a 3-axis accelerometer, and a 3-axis magnetometer. The Pi devices offloaded requests to the proximate edge computing infrastructure using a wireless local area network (WLAN) via an AC400 Nokia access point. Additionally, a laptop generated a large number of requests in order to create a dynamic realistic workload. As explained in Section 4, the Edge layer of the proposed architecture deployment was achieved exploiting the OpenStack and OSM for NFV orchestration, of the utilized tedbed facility. A powerful Dell PowerEdge T630 server with 32 vCPUs, 64GB RAM and 1TB storage was used to deploy all the necessary VDUs. Initially, in order to compute various resource profiles of the application we created a measurement data set and inferred three different types of VDUs. Table 1 provides the computing resources and the number of requests to be served per time interval for each resource profile. For every flavor, the average response time is set to $5sec$. The control time interval was set to 30 *sec*. At the beginning of each interval, a workload estimation was computed according to Section 3.2.2 and subsequently the SC determined the combination of the running VDUs.

[3]https://www.sparkfun.com/products/13762

Table 1: VDU flavors based on the Resource Profiling.

Flavor	vCPUs	RAM	Requests served per Interval	Average Response Time
Small	2	2GB	3	5sec
Medium	4	4GB	14	5sec
Large	8	8GB	27	5sec

5.2. Evaluation of User's Location Estimation

The purpose of this experiment is to demonstrate the accuracy of the location estimation technique. Figure 3 presents a sketch of Millennium Square topology. The wireless access point was located at point A(10,0) and enabled the users to offload their requests to the EC infrastructure. Furthermore, Figure 3 illustrates the "offloading" map of the Millennium Square. In the green-coloured area, the wireless signal strength was high and the users were encouraged to offload requests, while the signal strength was poor in the white-coloured areas. The strength of the wireless signal was measured using the Wavemon software[4], and the areas were divided by setting the threshold for a good quality signal to the value of $-70dB$. With respect to the differentiation in terms of signal strength in these areas, please note that in the specific square in the "white areas" physical obstacles exist (*e.g.,* metallic constructions), which cause high interference resulting in wireless "dead zones". However, these metrics were performed by a single user while in a real world scenario the users' signal interference may heavily affect signal strength. To address this issue the offloading decision considers both the location-based classification of a user according to the "offloading map" and the contemporary signal strength of a user to specify his transmission capability in an online manner. This procedure is fundamental for the proposed architecture, since the object recognition service is CPU-intensive. In such a way, we ensure that the transmission time is negligible relative to the processing time of an offloaded request.

[4]https://github.com/uoaerg/wavemon

Table 2: Maximum error of estimation at each point of the trajectory of the user.

Point	Maximum Error of Estimations (m)
O(0,0)	0
A(10,0)	0.08
B(10,10)	0.1131
C(10,20)	0.1789
D(18,20)	0.394
E(18,30)	0.4861

In order to evaluate the location estimation accuracy, the user repeatedly (*i.e.,* five times) followed a path of six already known points, starting from point O and finishing at point E, while holding a Raspberry PI device mounted with the aforesaid IMU sensor. In Figure 3, for each point, the real position is marked with a star, while the cross symbol refers to the estimated position. At each point, the drawn circle represents the maximum estimation error. Additionally, Table 2 summarizes the estimation maximum error at each predefined point. It is obvious that the location estimation approach is very precise and the error is less than 0.5 meters in all cases, which is acceptable in our scenario. It is also worthwhile mentioning that the estimation error is cumulative, thus, the error becomes higher at the last point.

5.3. Evaluation of Offloading Decision

Based on the experimental set-up described previously, an experiment of one hour duration, was conducted to validate the success of the two-step offloading decision mechanism and the performance of the LBS on the edge infrastructure. A set of photos captured in Millennium Square were used to generate requests, whose inter-arrival time follows a Poisson distribution. As depicted in Figure 4, the black-colored line represents the magnitude of requests that were offloaded by the mobile devices based on the first step of the offloading mechanism. At the beginning of each time interval, the estimation of these requests (green-colored line) was computed by the Kalman Filter and were forwarded for further

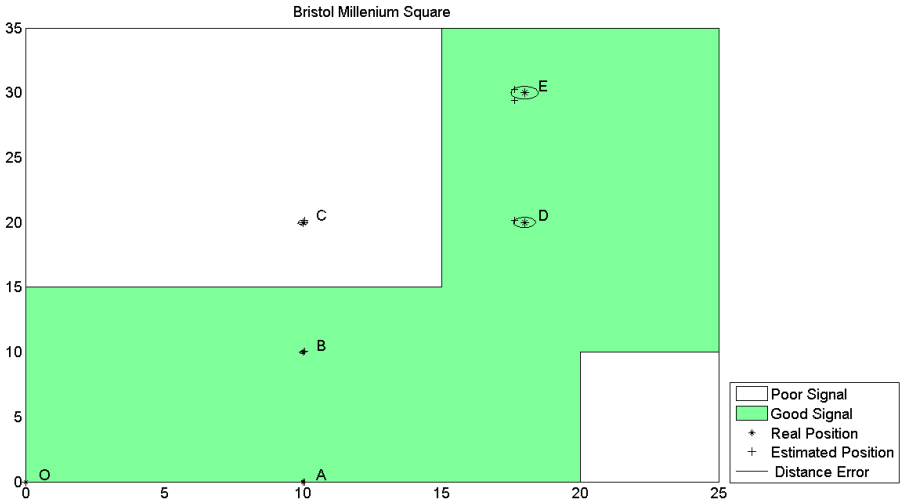


Figure 3: The "offloading map" and the user trajectory scenario at the Millennium Square of Bristol's 5G testbed.

processing by the EC infrastructure. The blue-colored line corresponds to the number of executed requests on the deployed VDUs after the final offloading decision on the EC side, while the red-colored line represents the requests that were rejected by the EC servers and were executed locally on the users' devices. Two useful remarks can be discussed here. First, the workload predictor can accurately follow the fluctuation of requests. It fails only when a sudden change occurs (e.g., at $200sec$). Therefore, the number of the rejected requests is high only at these intervals, and until the prediction is properly and timely adapted, as shown in this figure. Secondly, 91.37% of total requests were successfully offloaded and executed on the EC side, while only 8.63% were executed locally on the devices, which evinces the success of the two-step offloading strategy.

The proposed workload predictor is compared with an autoregressive integrated moving average (ARIMA) model [10] which is widely used for time-series forecasting. For comparison purposes, two well-known accuracy metrics are used, namely the percentage average error (PAE) and the Best Fit Rate (BFR)

[39], which are defined respecitvely,

$$PAE = 100|\frac{E[y_k - \hat{y}_k]}{E[y_k]}|$$

$$BFR = 100\%max(1 - \sqrt{\frac{E[(y_k - \hat{y}_k)^2]}{E[(y_k - \overline{y}_k)^2]}}, 0)$$

The lower the PAE score is the more accurate the predictor is. However the PAE metric is sensitive to the values of large absolute error. Thus, we utilize the BFR metric which is more robust to the effect of small set of values. Contrary to PAE, high score of BFR indicates precise estimation.Table 3 shows the score of Kalman-based and ARIMA-based workload predictors. As it shown, the PAE score is very low for both predictors and ARIMA-based predictor is slightly better than Kalman-based. On the other hand, the proposed workload predictors achieves significantly higher BFR score than the ARIMA-based. These remarks indicate that both predictors provide successful estimation, which is acceptable for the scaling and offloading decisions.

The key performance indicator of the object recognition service is the response time of the requests. The execution time of a request by the mobile device varies between 30 to 60 sec. Following the design principles of Section 3.2.1, the extracted resource profiles were selected to have an average response time of $5sec$, half of the maximum acceptance value by the users. As it is shown in Figure 5, the average response time of the offloaded requests throughout the experiment is $4.75sec$ (below the aforesaid threshold), subdivided into processing time (with an average of $4.02sec$) and transmission time (with an average of $0.73sec$). The average response time of the offloaded requests is at least five times lower than the local execution, which indicates that the resource

Table 3: Comparison of the Workload Predictors.

Accuracy Metric	Kalman-based Predictor (%)	ARIMA-based Predictor (%)
PAE	3.72	0.25
BFR	73.81	63.38

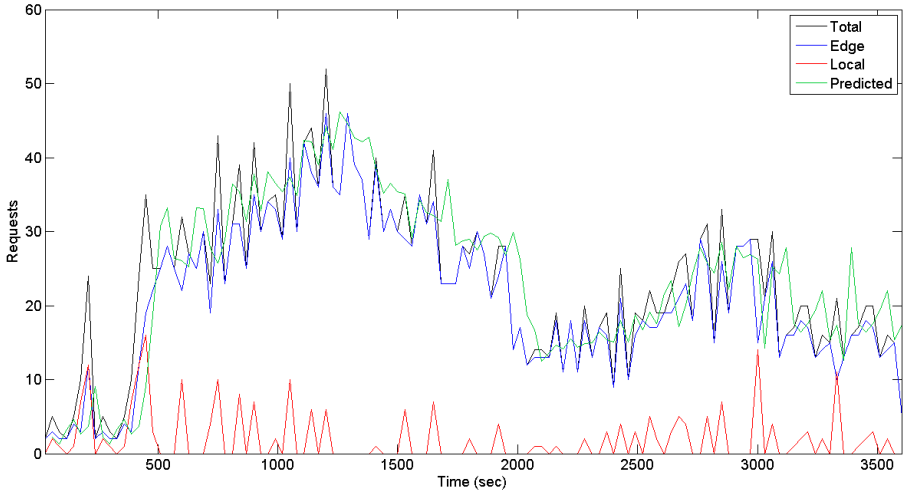


Figure 4: Number of total, predicted, remotely and locally executed requests per time interval

profiling models effectively the performance of the application and motivates its necessity for properly meeting the QoS requirements. This also demonstrates that the second step of the offloading decision on the EC side becomes more a requirement rather than a desire. Finally, the low transmission time implies that the first-step of the offloading decision benefits the fulfilment of the user's requirement.

5.4. Evaluation of Scaling Mechanism

The evaluation of the scaling mechanism was conducted based on the one-hour experiment described in the previous subsection. In particular, Figure 6 depicts the selected combination of operating VDUs at each time interval. Each combination of operating flavors is illustrated with a distinct color. Having in mind the prediction of requests in Figure 4 and corresponding trend, at the first intervals of the experiment only a few visitors offloaded their requests on the EC infrastructure. This workload was handled by a small or medium flavor VDUs. At the $600th$ simulation second, more visitors swarmed in the Square interested in PoIs, resulting in the gradual increase of the workload, which peaked at 50 requests per interval at the $1200th$ second. During this period, the

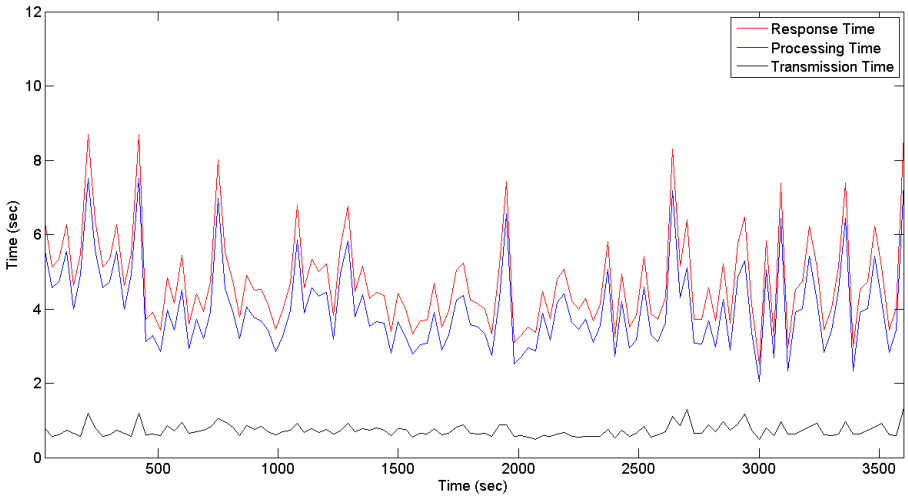


Figure 5: Average response, processing and transmission time of offloaded requests.

full functionality of the scaling mechanism is clearly demonstrated. Specifically, to deal with the rising traffic, the SC determined the appropriate combination of active VDUs in order to avoid performance degradation. Moreover, even when the maximum number of requests occurred, the average response time remained below $4secs$. When the number of users began to decrease again, the scaling mechanism adapted timely and released resources from the operating VDUs, to follow the course. After the $2000th$ second of the experiment, the amount of requests fluctuates between 20 and 30 per interval. As it is shown, the scaling mechanism efficiently adapted the resources of running VDUs towards the workload fluctuation. The success of this mechanism is further strengthened by the performance of each VDU, as explained in detail below.

In Figure 7, for each VDU, the average response time of the offloaded requests is depicted in the respective subfigure. The information derived from these subfigures complements the operation of the scaling mechanism, and elucidates the periods of time that each VDU is active or not. During the whole experiment, there is only one time interval that the user's maximum acceptable response time is violated in the medium VDU. This implies that the scaling deci-

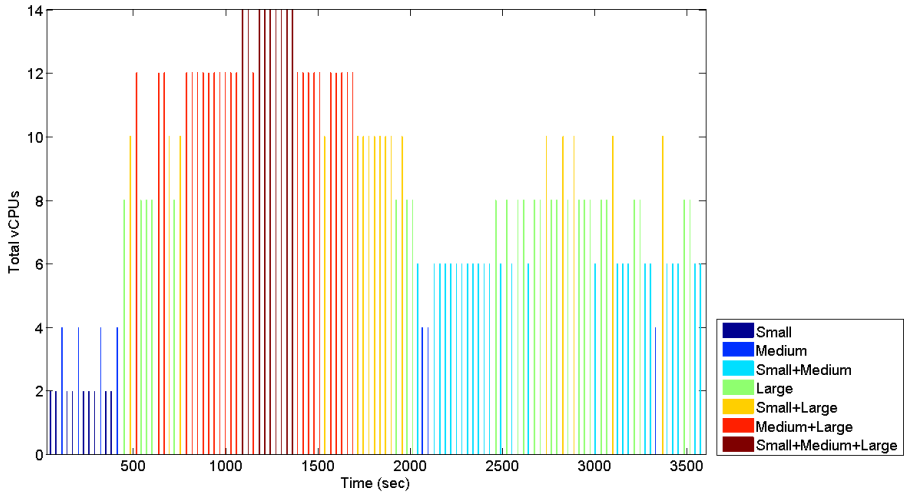


Figure 6: VDU flavors selected to operate at each interval according to the scaling mechanism.

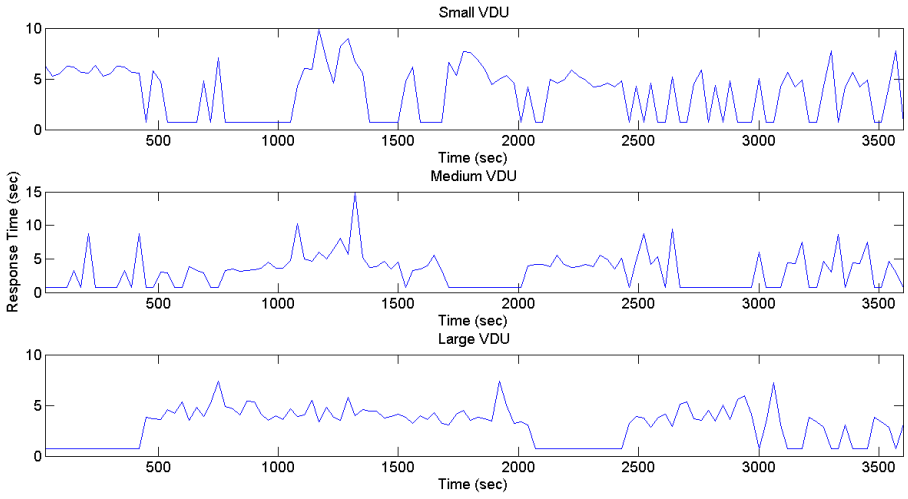
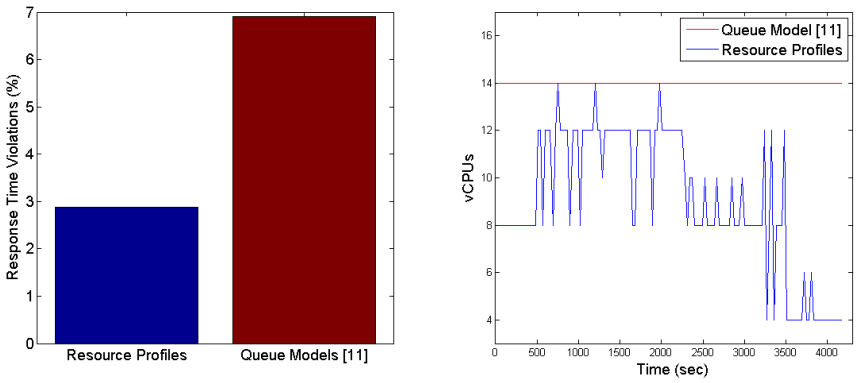


Figure 7: Processing time of requests at each VDU per time interval

sion based on the resource profiles and the estimation of requests can guarantee the QoS requirements and the success of the final offloading decision.

5.5. Comparison of the Scaling Mechanism

In this section, we present a comparative evaluation of the proposed framework with the work presented in [11]. Similarly to the proposed EC architecture, this work presented a setting of interconnected clusters, which form a wireless metropolitan area network. Each cluster of computers; namely cloudlet, has a static resource provisioning to serve the offloaded requests. The main goal of this study is to redirect part of the incoming traffic of the overutilized cloudlets towards the underutilized ones. In [11], the average response time for each cloudlet is calculated by a queue model. Utilizing the M/M/c queue model, the maximum offloaded workload served at each cloudlet is bounded by the corresponding service rate capabilities. The rest is considered rejected and redirected back to the user to perform local execution of the request. The service rate of a cloudlet is defined as the average number of requests that can be executed throughout a time interval. Following, the results of a comparative evaluation between the tho methods, are presented. Moreover, to ensure an unbiased comparison, we deployed, the method presented in [11] in the same manner to the system orchestration and experiment setup presented in Sections 4,5 and we used the exact same workload of requests for both setups. Following the design in subsection 3.2.1 the threshold of the QoS violations is set at $10sec$. In one hour of experimentation, as depicted in Figure 8a the resource profiles mechanism reported only 2.88% QoS violations contrary to 6.9% of the queuing theory modeling proposed in [11]. The static placement of the three flavors discussed in Section 5.1 has a total of 14 vCPUs. On the other hand, our approach utilized 35% less or an average of 9 vCPUs, throughout the experiment, as shown in Figure 8b. More specifically when the offloaded workload can be served the medium flavor VDU, as in the end of this experiment, the static placement results in a waste of resources that could be used for another application. Finally, to achieve all of the above, our framework had to reject 9.3% of the incoming



(a) QoS Violations.

(b) Utilization of Resources.

Figure 8: Comparative Experiment

requests, while in [11] 1.21% was rejected. To sum up, a dynamic resource allocation mechanism along with the resource profiles, are able to guarantee the QoS requirements while the resources utilized are minimized, sacrificing a small percent of requests rejected.

6. Conclusions and Future Work

This article focused on the introduction, design and experimental evaluation of a scalable two-tier Edge Computing architecture to realize location based services in a smart city context. The scenario that motivated this work and was used for validation purposes, considers several users located in the vicinity of a touristic place, moving around looking for PoIs, and holding portable devices. Exploiting media content they opt to get additional information for the PoIs by an object-recognition service. As such, their requests are offloaded via Wi-Fi to an EC infrastructure in order to reduce power consumption and improve response time.

In particular, we proposed and evaluated a location estimation technique to assist with a smart offloading decision at the Device Layer, using contextual information of the users' state (*i.e.,* position and wireless signal strength). Fur-

thermore, a mechanism for scaling and allocating the resources of the underlying infrastructure is realized to support and accompany the realization of the final offloading decision at the Edge layer, which depends on resource availability. This EC architecture is applicable in various MEC environments and mMTC application scenarios. The overall approach was implemented and evaluated in a real 5G testbed exploiting NFV orchestration and widely used software tools (*e.g.,* OpenStack, OSM) for the experiments execution. Based on the obtained detailed experimental evaluation results, the location estimation method proved to be very reliable for the smart touristic scenario under consideration. With this valuable estimation, the two-steps smart offloading decision had a dominant role in the performance of the overall system and architecture. The scaling mechanism and the resource profiles yielded remarkable performance gains when compared to relevant well-established research works in the literature, not only in terms of meeting the user QoS criteria, but also eliminating the under-utilization of resources.

Finally, as part of our current and future work, we aim to extend these novel methodologies for the position estimation, to further improve its accuracy, while including pattern recognition of users mobility in the overall designed architecture. This can be achieved by exploiting Machine Learning techniques and combining them with a more detailed modeling of the signal interference among the users, in order to ultimately offer a more effective and efficient offloading decision. Furthermore, for time-sensitive applications, e.g., augmented or virtual reality ones, the resource profiles can be used to design feedback controllers that enable scale up/down operations on the deployed VDUs and achieve fine-grained regulation of the performance.

Acknowledgement

This work was supported by European Commission H2020 5GinFIRE Project [Grant Number. 732497].

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work.

References

[1] Cisco, Internet of things at a glance, https://www.cisco.com/c/dam/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf, (accessed on 28 January 2020) (2017).

[2] Ericsson, Ericsson mobility report - Q3 2019, https://www.ericsson.com/4acd7e/assets/local/mobility-report/documents/2019/emr-november-2019.pdf, (accessed on 28 January 2020) (2019).

[3] European Commission, 5GPPP Vision: The 5G Infrastructure Public-Private Partnership: the next generation of communication networks and services, https://ec.europa.eu/research/press/2013/pdf/ppp/5g_factsheet.pdf, (accessed on 28 January 2020) (2013).

[4] J. Schleicher, M. Vögler, S. Dustdar, C. Inzinger, Enabling a smart city application ecosystem: requirements and architectural aspects, IEEE Internet Computing 20 (2) (2016) 58–65. doi:10.1109/MIC.2016.39.

[5] J. Herrera, J. Botero, Resource allocation in nfv: A comprehensive survey, IEEE Transactions on Network and Service Management 13 (3) (2016) 518–532. doi:10.1109/TNSM.2016.2598420.

[6] N. Feamster, J. Rexford, E. Zegura, The road to sdn: an intellectual history of programmable networks, ACM SIGCOMM Computer Communication Review 44 (2) (2014) 87–98. doi:10.1145/2602204.2602219.

[7] A. Correa, M. Barcelo, A. Morell, J. Vicario, A review of pedestrian indoor positioning systems for mass market applications, Sensors 17 (8) (2017) 1927. doi:10.3390/s17081927.

[8] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, J. Jue, All one needs to know about fog computing and related edge computing paradigms: A complete survey, Journal of Systems Architecturedoi:10.1016/j.sysarc.2019.02.009.

[9] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, D. Sabella, On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration, IEEE Communications Surveys & Tutorials 19 (3) (2017) 1657–1681. doi:10.1109/COMST.2017.2705720.

[10] R. Calheiros, E. Masoumi, R. Ranjan, R. Buyya, Workload prediction using arima model and its impact on cloud applications' qos, IEEE Transactions on Cloud Computing 3 (4) (2014) 449–458.

[11] M. Jia, W. Liang, Z. Xu, M. Huang, Cloudlet load balancing in wireless metropolitan area networks, in: IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE, 2016, pp. 1–9.

[12] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, O. Rana, Fog computing for the internet of things: A survey, ACM Transactions on Internet Technology (TOIT) 19 (2) (2019) 1–41. doi:10.1145/3301443.

[13] I. Afolabi, J. Prados, M. Bagaa, T. Taleb, P. Ameigeiras, Dynamic resource provisioning of a scalable e2e network slicing orchestration system, IEEE Transactions on Mobile Computingdoi:10.1109/TMC.2019.2930059.

[14] ETSI, Open Source NFV Management and Orchestration (MANO), https://osm.etsi.org/, (accessed on 28 January 2020) (2020).

[15] K. Katsaros, V. Glykantzis, P. Papadimitriou, G. Papathanail, D. Dechouniotis, S. Papavassiliou, MESON: Facilitating cross-slice communications for enhanced service delivery at the edge, in: 2019 European Conference on Networks and Communications (EuCNC) Poster Session, IEEE, 2019.

[16] A. Leivadeas, G. Kesidis, M. Ibnkahla, I. Lambadaris, Vnf placement optimization at the edge and cloud, Future Internet 11 (3) (2019) 69. doi:10.3390/fi11030069.

[17] C. Sonmez, A. Ozgovde, C. Ersoy, Fuzzy workload orchestration for edge computing, IEEE Transactions on Network and Service Management 16 (2) (2019) 769–782. doi:10.1109/TNSM.2019.2901346.

[18] B. Gao, Z. Zhou, F. Liu, F. Xu, Winning at the starting line: Joint network selection and service placement for mobile edge computing, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, 2019, pp. 1459–1467. doi:10.1109/INFOCOM.2019.8737543.

[19] J. Xu, B. Palanisamy, H. Ludwig, Q. Wang, Zenith: Utility-aware resource allocation for edge computing, in: 2017 IEEE International Conference on Edge Computing (EDGE), IEEE, 2017, pp. 47–54.

[20] W. Wang, Y. Zhao, M. Tornatore, A. Gupta, J. Zhang, B. Mukherjee, Virtual machine placement and workload assignment for mobile edge computing, in: 2017 IEEE 6th International Conference on Cloud Networking (CloudNet), IEEE, 2017, pp. 1–6.

[21] M. Bagaa, T. Taleb, A. Laghrissi, A. Ksentini, H. Flinck, Coalitional game for the creation of efficient virtual core network slices in 5g mobile systems, IEEE Journal on Selected Areas in Communications 36 (3) (2018) 469–484.

[22] K. Kumar, J. Liu, Y. Lu, B. Bhargava, A survey of computation offloading for mobile systems, Mobile Networks and Applications 18 (1) (2013) 129–140. doi:10.1007/s11036-012-0368-0.

[23] Y. Shi, S. Chen, X. Xu, Maga: A mobility-aware computation offloading decision for distributed mobile cloud computing, IEEE Internet of Things Journal 5 (1) (2017) 164–174. doi:10.1109/JIOT.2017.2776252.

[24] L. Tang, S. He, Multi-user computation offloading in mobile edge computing: A behavioral perspective, IEEE Network 32 (1) (2018) 48–53. doi:10.1109/MNET.2018.1700119.

[25] P. Apostolopoulos, E. Tsiropoulou, S. Papavassiliou, Risk-aware Social Cloud Computing based on Serverless Computing Model, in: IEEE Global Communications Conference (GLOBECOM), 2019., 2019.

[26] M. Chen, Y. Hao, Task offloading for mobile edge computing in software defined ultra-dense network, IEEE Journal on Selected Areas in Communications 36 (3) (2018) 587–597. doi:10.1109/JSAC.2018.2815360.

[27] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, Y. Zhang, Selective offloading in mobile edge computing for the green internet of things, IEEE Network 32 (1) (2018) 54–60. doi:10.1109/MNET.2018.1700101.

[28] H. Yang, Y. Zhang, Y. Huang, H. Fu, Z. Wang, Wknn indoor location algorithm based on zone partition by spatial features and restriction of former location, Pervasive and Mobile Computing 60 (2019) 101085. doi:10.1016/j.pmcj.2019.101085.

[29] W. Kang, Y. Han, Smartpdr: Smartphone-based pedestrian dead reckoning for indoor localization, IEEE Sensors journal 15 (5) (2014) 2906–2916. doi:10.1109/JSEN.2014.2382568.

[30] A. Correa, M. Llado, A. Morell, J. Vicario, Indoor pedestrian tracking by on-body multiple receivers, IEEE Sensors Journal 16 (8) (2016) 2545–2553. doi:10.1109/JSEN.2016.2518872.

[31] M. Avgeris, D. Spatharakis, N. Athanasopoulos, D. Dechouniotis, S. Papavassiliou, Single Vision-Based Self-Localization for Autonomous Robotic Agents, in: 7th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), 2019, pp. 123–129. doi:10.1109/FiCloudW.2019.00035.

[32] T. Zhou, M. Brown, N. Snavely, D. Lowe, Unsupervised learning of depth and ego-motion from video, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1851–1858.

[33] N. Ho, P. Truong, G. Jeong, Step-detection and adaptive step-length estimation for pedestrian dead-reckoning at various walking speeds using a smartphone, Sensors 16 (9) (2016) 1423. doi:10.3390/s16091423.

[34] L. Ljung, System Identification: Theory for the User, Prentice-hall, Inc., 1987. doi:10.1007/978-1-4612-1768-8_11.

[35] D. Ardagna, B. Panicucci, M. Trubian, L. Zhang, Energy-aware autonomic resource allocation in multitier virtualized environments, IEEE transactions on services computing 5 (1) (2010) 2–19.

[36] R. Kalman, A new approach to linear filtering and prediction problems, Journal of basic Engineering 82 (1) (1960) 35–45. doi:10.1115/1.3662552.

[37] 5GinFIRE, University of Bristol 5g Testbed, https://5ginfire.eu/university-of-bristol-5g-testbed/, (accessed on 28 January 2020) (2019).

[38] Google, TensorFlow Platform, https://www.tensorflow.org/, (accessed on 28 January 2020) (2020).

[39] D. Dechouniotis, N. Leontiou, N. Athanasopoulos, A. Christakidis, S. Denazis, A control-theoretic approach towards joint admission control and resource allocation of cloud computing services, International Journal of Network Management 25 (3) (2015) 159–180.