Multi-Objective Optimal Deployment of SDN-Fog Infrastructures and IoT Applications

Juan Luis Herrera*, Jaime Galán-Jiménez*, Paolo Bellavista†, Luca Foschini†, Jose Garcia-Alonso*, Juan M. Murillo* and Javier Berrocal*
*Department of Computer Systems and Telematics Engineering, University of Extremadura, Spain.
†Dipartamento di Informatica-Scienza e Ingegneria, University of Bologna, Italy.
[jlherrerag, jaime, jgaralo, juanmamu, jberolm]@unex.es, [paolo.bellavista, luca.foschini]@unibo.it

***Abstract*—The Internet of Things has brought digitalization to intensive domains through the automation of their real-world processes. However, the criticality of these processes is reflected in high Quality of Service (QoS) requirements for the application to work properly. Moreover, business-level QoS, such as the operational cost, are also key to the feasibility of these applications. This QoS depends on three, closely-related dimensions: the application software, the computing devices and the communication network, which provide high flexibility to obtain different performances at different costs. Thus, to achieve optimal QoS in these scenarios, the application, computing and networking dimensions must be optimized, considering their crucial interplay in a joint effort. Furthermore, this solution must allow multi-objective optimization, finding the optimal trade-off between operational cost and application performance. In this paper, we present Multi-Objective SDN Fog Optimization (MO-SFO), a holistic framework that allows for the optimization of both the response time and the deployment cost. MO-SFO is evaluated over an emulated smart city case study, showing the cost and performance trade-off achieved in different topologies.**

I. Introduction

The advent of the Internet of Things (IoT) paradigm has enabled for the bridging of computing applications and networks with real-world processes [1]. The potential of such computerization has attracted the interest of the research community in intensive domains, such as smart cities [1], industry or healthcare [2]. Nonetheless, the criticality of these domains is translated to strict Quality of Service (QoS) requirements for their IoT applications, such as low response times [1], [2]. Therefore, the integration of IoT-based systems in intensive domains is contingent on their ability to provide a high enough QoS, such as smart surveillance systems whose performance depends on the QoS [1], [2]. There are three dimensions that affect the QoS of IoT applications: application, computing, and networking [3]. All of them need to be considered in order to meet the strict QoS requirements of these intensive domains.

Regarding the application dimension, modern IoT applications require to be highly evolvable and interoperable, as well as allowing distributed deployments [4]. Thus, these applications are often designed following the Service-Oriented Computing paradigm, and, more concretely, the Microservices Architecture (MSA) design pattern [4]. In MSAs, applications are divided into small and loosely-coupled microservices, which perform simple tasks individually and collaborate to support more complex functionalities. Moreover, each of these microservices can be deployed individually, allowing the application to be deployed in a distributed manner and providing higher flexibility. Nonetheless, this is not a simplistic decision, as the application's QoS is affected by where each microservice is deployed. On the one hand, devices with higher computational power are able to provide lower execution times and, thus, lower response times. On the other hand, powerful devices tend to be further away from the data source, therefore having higher latencies, raising the response times. The problem of optimally deploying a set of microservices is labelled in literature as the Decentralized Computation Distribution Problem (DCDP) [5].

In the computer dimension, the QoS requirements, as well as the aforementioned focus on distribution, also motivate the used paradigms. Traditionally, cloud computing was used to deploy IoT applications [6]. Nonetheless, due to the complexity of guaranteeing a suitable QoS using cloud servers exclusively, which tend to be far away from IoT devices, intensive domains tend to move towards fog computing infrastructures [1]. In these infrastructures, servers that are closer to users (*fog nodes*, FNs) are used alongside with cloud servers [6]. Fog infrastructures generally enhance the QoS of the applications deployed in them due to their lower latencies. Nonetheless, the microservices running on both the cloud and FNs need to communicate between themselves, and therefore, the placement of these FNs is crucial in the QoS enhancement experienced by the application [2], [3]. In order to place them optimally in the infrastructure, the Fog Node Placement Problem (FNPP) [2] needs to be solved.

The QoS of the communications between microservices and, therefore, between devices, depends on the networking dimension. The network needs to be scalable and flexible, meeting the required QoS [1], [3]. Furthermore, in fog infrastructures, microservices are often replicated and deployed in different FNs [3]. Before consuming a certain microservice, the IoT device needs to learn where the nearest replica is deployed (*service discovery*) [7]. In order to perform this task separately from the IoT application's concerns, it would be desirable for the network to automatically route requests to the nearest replica. It is possible to perform service discovery at the network level while providing scalability and flexibility, by exploiting the Software-Defined Networking (SDN) paradigm [7],

A foundation of the SDN paradigm is to decouple the data plane, embodied in SDN switches, from the control plane, which is managed at the SDN controller. Therefore, the SDN switches and the SDN controller need to communicate. The communication QoS between them is crucial, as every time these planes need to interact, they will be subject to such QoS [8]. In order to optimize this QoS, the SDN Controller Placement Problem (CPP) [8] can be solved, placing the SDN controller wherever the QoS from the switches is best.

These three dimensions are inherently coupled, as each decision taken in a dimension affects the rest. Changing the placement of the controller affects the overall network QoS [8], which dictates the QoS between FNs and, consequently, the QoS between the microservices deployed in them [3]. Changing the placement of a FN affects the flow of traffic through the network, as a node that is the source and destination of communications is moved [2]. The flow of traffic is key to the optimal placement of the SDN controller [8], and thus, the network QoS will be affected, therefore also affecting the QoS of the microservices that are either deployed in the moved FN or communicating with microservices deployed in the moved FN [3].

However, companies usually have limited resources that limit the optimal deployment configurations that they can feasibly deploy [9]. In these cases, there are two QoS objectives in conflict: performance QoS, namely response time, and business QoS, in the form of deployment cost [9]. These two QoS objectives need to be traded off, performing a multi-objective optimization that suggests high-performance, low-cost deployment configurations that can be feasibly deployed.

In this work, we present Multi-Objective SDN Fog Optimization (MO-SFO), a multi-objective framework for the holistic solution of the CPP, the DCDP and the FNPP, optimizing response time and cost. In [3], some of the authors of this work proposed Umizatou, a framework able to optimize the response time by solving these three problems in a single effort. However, Umizatou only allows for the optimization of response time. This shortcoming complicates the use of Umizatou in scenarios in which the deployment cost is a QoS objective, as well as those in which cost and response time need to be traded off. MO-SFO extends Umizatou by considering multiple metrics, as well as multi-objective optimization. Furthermore, MO-SFO has been evaluated over an emulated SDN-fog testbed, whereas Umizatou's evaluation was not performed in an emulated environment [3]. The main contributions of this work are:

- The formalization of a problem that combines CPP, DCDP and FNPP for intensive IoT environments, aimed at optimizing the response time, the deployment cost, or both at the same time.
- The formulation of a multi-objective, Mixed Integer Linear Programming (MILP) optimization solution.
- The evaluation of MO-SFO over a realistic, emulated smart city case study.

The remainder of this paper is structured as follows. Sec. II explains the system model of MO-SFO. Sec. III details the multi-objective optimization formulation of MO-SFO making use of MILP. Sec. IV presents the evaluation of MO-SFO using an emulated smart city case study. Finally, Sec. V concludes the paper and highlights future challenges.

II. System model

To make it easier to understand the MO-SFO model, the problem is explained through a smart city case study. This case study is based on Intel's OpenVINO toolkit, an industrial framework for the execution of deep learning video and audio analysis models, designed for its use in fog environments and tailored towards smart cities [10]. Nonetheless, the MO-SFO framework is not exclusive to smart city scenarios, and could be used in other intensive IoT domains as well.

In the case study, the smart city has a set of cameras and microphones, which act as IoT devices and continuously send their information to a surveillance IoT application. This application allows for five different functionalities: detection of presence of people and vehicles, visual tracking of such people and vehicles, classification of the vehicles based in their attributes (e.g., the type of vehicle, their colors or their sizes), detection of the zone vehicles and people are on (e.g., whether they are on the sidewalk or on the road, on which lane of the road they are on), and environmental audio recognition (e.g., detection of car horns or car crashes through their sound) [10]. Each of these functionalities is implemented as a microservice. Based on their role (e.g., security surveillance cameras, traffic control cameras, emergency detection microphones), they can request these functionalities to be carried out with their video or audio inputs. This application is depicted in Fig. 1.

For the deployment of this surveillance application in the smart city, a SDN network, with the topology shown in Fig. 1, is envisioned. While this network connects the IoT devices, the placement of the SDN controller, as well as the fog nodes, is not yet decided. Moreover, both the application's response time and its deployment cost are considered crucial QoS metrics that need to be optimized [1]. Thus, the CPP, the FNPP, and the DCDP need to be solved to optimize the QoS of this application. The application operator defines the two QoS metrics to be optimized in order to find the best trade-off between response time and cost.

In this scenario, the objective of MO-SFO is to deploy SDN controllers, FNs and microservice instances within the FNs, in a manner that minimizes both the cost of the deployment and the application's response time. To do so, MO-SFO decides to use the deployment shown in Fig. 1: 2 FNs are deployed, one on the top side, with microservices for video processing, and one in the bottom, including audio recognition, and an SDN controller is deployed in the middle. MO-SFO deems this deployment optimal in terms of both objectives. While it would be possible to deploy more microservice replicas in each FN to further decrease response time, doing so would increase the energy consumption, and thus, the operational expenditures (OPEX) of the system. Similarly, deploying more FNs or SDN controllers would also decrease the response time, at the cost of higher capital expenditures (CAPEX). On the other hand, it would be possible to deploy a single FN,

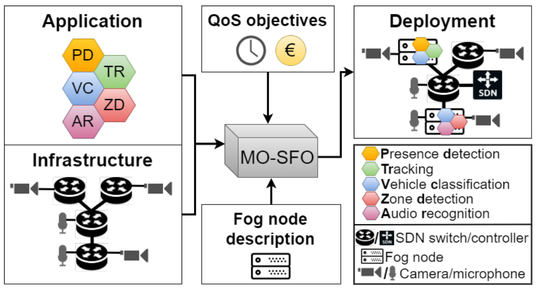

Figure 1: MO-SFO in the smart city case study.

running a replica of each microservice, to reduce the CAPEX and OPEX of the system. However, doing so would provoke an increase in the response time.

The decisions of MO-SFO are possible due to its multi-objective joint approach: in order to assess if deploying additional FNs is more cost-effective than deploying additional SDN controllers, MO-SFO needs control over both the computing and networking dimension. Furthermore, the cost-effectiveness of the equipment depends on the application dimension's communication patterns, computational workload and microservice placement. This sort of analysis can only be performed if both objectives are optimized with an holistic perspective that covers all three dimensions.

III. Problem formulation

In this section, the mathematical abstraction of the model of the joint CPP, DCDP and FNPP is presented as an MILP formulation. This formulation allows for the problem to be solved through the use of automatic MILP software solvers.

In this formulation, the infrastructure is represented by a directed graph $G = \{V, L\}$, where L represents the links and V represents the infrastructure's vertices. A link denoted as l_{ij} links together vertices i and j, with a propagation latency δ_{ij} and a maximum capacity θ_{ij}. The vertices of the infrastructure can be divided into two subsets: $V = H \cup S; S \cap H = \emptyset$, with H containing the vertices that are hosts, and S containing those that are SDN switches. Furthermore, every switch $s \in S$ has a certain processing latency δ_s. To simplify the understanding of the formulation, let $SW(v) \forall v \in V$ be a binary function, whose value is 1 if $v \in S$ and 0 otherwise. Moreover, each switch in the infrastructure has a given CAPEX ($CAPEX_s$) and OPEX per second due to energy consumption ($OPEX_s$).

Continuing with fog nodes, let r be the total RAM memory of an FN, which determines the number and kind of microservice replicas that can be deployed in it. In terms of cost, placing the FN has a certain CAPEX, defined as $CAPEX_{FN}$, and consumes an amount of energy, i.e., it has a certain OPEX. The energy consumption depends on the usage of the FN, as more intensive usage consumes more energy, and thus, we define $OPEX_{FN}$ as the OPEX per second for the corresponding FN. With respect to the SDN controller, we need to consider the size of control packets, which we label σ, to support multiple SDN protocols. Nonetheless, in this work, this size is taken from the OpenFlow 1.3 specification [11]. Similarly to the switches, each controller has a certain CAPEX ($CAPEX_C$) and OPEX per second due to energy consumption ($OPEX_C$).

To represent the application, we use a model based in the MSA pattern, i.e., the application is divided into a set of microservices we label M. Each microservice m consumes an amount of RAM r_m, and its input and output data have a size of I_m and O_m, respectively. Furthermore, in MSA-based applications, each client (i.e., IoT device) requests for functionalities in the form of *workflows* [4]. Each workflow can involve one or multiple collaborating microservices, which are executed in order, pipelining the output of a microservice to the input of the next one [9]. We represent the total set of workflows requested as W, and each workflow as an ordered set of microservices $w = \{c_1, c_2, ..., c_{|w|}\} \subseteq M$. It is important to note that each element is a microservice $c_i \in M$, and elements from different workflows can refer to the same microservice. For simplicity, we also define the binary function $WR(w, h) \forall w \in W, h \in H$, which takes the value of 1 if workflow w is requested by host h and 0 otherwise.

Finally, we introduce the parameter $\alpha \in [0, 1]$ as a means for the user of MO-SFO to select the objective or objectives of the optimization, as well as to decide the priority of each objective. If $\alpha = 0$, MO-SFO will only consider the deployment cost as the QoS objective, useful for scenarios with very stringent budgets and not very time critical. Conversely, $\alpha = 1$ only enforces the response time objective, which may be used whenever the budget is not a constraint. A value of $\alpha = 0.5$ indicates MO-SFO should find a trade-off between deployment cost and response time, which is meant for problems where the budget constrains an application that requires high performance.

After defining the parameters for the problem, the decision variables need to be defined as well. To solve the FNPP, FNs must be placed, replacing SDN switches: let $n_s \forall s \in S$ be a binary variable that will take a value of 1 if a FN is placed in switch s. Continuing the joint approach, to solve the DCDP, microservice replicas need to be deployed to FNs. We define $z^w_{sc_a} \forall w \in W, s \in S, a \in [1, |w|]$ a binary variable, representing whether the replica of microservice c_a in workflow w is deployed to the FN in switch s or not. Moreover, the communication between microservices generates traffic flows, which are represented by the binary variables $f^{vwc_a}_{ij} \forall l_{ij} \in L, v \in V, w \in W, a \in [1, |w|]$. These variables take a value of 1 if the traffic generated by vertex v as a consequence of requesting microservice c_a of workflow w is routed through the link l_{ij}. It is important to note that the response from the last microservice in each workflow to the requesting IoT device must also be considered: let the binary variables $f'^{vw}_{ij} \forall l_{ij} \in L, v \in V, w \in W$ represent them with a value of 1 if the traffic generated by vertex v due to the response of workflow w is routed through the link l_{ij}.

Finally, the CPP needs to be solved as well. SDN controllers are also placed in switches, and thus, $x_s \forall s \in S$ is defined as a binary variable whose value is 1 if an SDN controller is placed on SDN switch s. Moreover, if multiple controllers are placed, each switch needs to know which controller they should communicate with: let $y_{ss'} \forall s, s' \in S$ be a binary variable that will be 1 if the SDN switch s is assigned to communicate with the controller from switch s'. These communications also generate traffic flows, which are represented through the binary variables $cf_{ij}^{s} \forall l_{ij} \in L, s \in S$. These variables take the value of 1 if the flow of control communications for SDN switch s is routed through the link l_{ij}. Then, the problem can be formalized as follows:

$$\min \alpha RT + (1-\alpha)(CAPEX + OPEX) \tag{1}$$

subject to:

$$\begin{aligned} RT =& \frac{1}{|W|} \sum_{w \in W} \sum_{s \in S} \sum_{l_{ij} \in L} (\\ & \sum_{a=1}^{|w|} (f_{ij}^{swc_a}) + f_{ij}'^{sw})(\delta_{ij} + SW(j)\delta_i) \\ & + SW(j) \sum_{l_{km} \in L} cf_{km}^{j}(\delta_{km} + SW(m)\delta_k) \end{aligned} \tag{2}$$

$$CAPEX = \sum_{s \in S}(CAPEX_{FN} n_s + CAPEX_C x_s + CAPEX_s u_s) \tag{3}$$

$$\begin{aligned} OPEX = \sum s \in S((\sum_{w \in W} \sum a = 1^{|w|} \frac{r_{c_a}}{r} OPEX_{FN} z_{sc_a}^{w}) \\ + OPEX_C x_s + OPEX_s u_s) \end{aligned} \tag{4}$$

$$\begin{aligned} u_s = \max(& \max_{l_{is} \in L, v \in V, w \in W, a \in [0,|w|]} (\max(f_{is}^{vwc_a}, f_{is}'^{vw})) \\ & , \max_{l_{is} \in L, s' \in S} (cf_{is}^{s'})) \end{aligned} \tag{5}$$

$$\forall w \in W, a \in [1, |w|] : \sum_{s \in S} z_{sc_a}^{w} = 1 \tag{6}$$

$$\forall w \in W, a \in [1, |w|], s \in S : z_{sc_a}^{w} \leq n_s \tag{7}$$

$$\forall s \in S : \sum_{w \in W} \sum_{a=1}^{|w|} z_{sc_a}^{w} r_{c_a} \leq r \tag{8}$$

$$\begin{aligned} \forall l_{ij} \in L : \sum_{s \in S} [cf_{ij}^{s} \sigma + \sum_{w \in W} [(\sum_{a=1}^{|w|} f_{ij}^{swc_a} I_{c_a}) + (f_{ij}'^{sw} O_{c_{|w|}})]] \\ \leq \theta_{ij} \end{aligned} \tag{9}$$

$$\forall i, v \in V, w \in W : \\ \sum_{j \in V} f_{ij}^{vwc_1} - f_{ji}^{vwc_1} = \begin{cases} WR(w,v) & \text{if } i = v \\ -WR(w,v) z_{ic_1}^{w} & \text{if } v \in H \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

$$\forall i, v \in V, w \in W : \\ \sum_{j \in V} f_{ij}'^{vw} - f_{ji}'^{vw} = \begin{cases} z_{vc_{|w|}}^{w} & \text{if } i = v \\ -WR(w,i) & \text{otherwise.} \end{cases} \tag{11}$$

$$\forall i, v \in V, w \in W, a \in [0, |w|] : -z_{vc_{a-1}}^{w} + z_{vc_a}'^{iw} \leq 0 \tag{12}$$

$$\forall i, v \in V, w \in W, a \in [0, |w|] : -1 + z_{ic_a}^{w} + z_{vc_a}'^{iw} \leq 0 \tag{13}$$

$$\forall i, v \in V, w \in W, a \in [0, |w|] : z_{vc_{a-1}}^{w} + 1 - z_{ic_a}^{w} - z_{vc_a}'^{iw} \leq 1 \tag{14}$$

$$\forall i, v \in V, w \in W, a \in [0, |w|] : -z_{ic_a}^{w} + z_{vc_a}''^{iw} \leq 0 \tag{15}$$

$$\forall i, v \in V, w \in W, a \in [0, |w|] : -z_{vc_{a-1}}^{w} + z_{vc_a}''^{iw} \leq 0 \tag{16}$$

$$\forall i, v \in V, w \in W, a \in [0, |w|] : z_{vc_{a-1}}^{w} + z_{ic_a}^{w} - z_{vc_a}''^{iw} \leq 1 \tag{17}$$

$$\forall i, v \in V, w \in W, a \in [2, |w|] : \\ \sum_{j \in V} f_{ij}^{vwc_a} - f_{ji}^{vwc_a} = \begin{cases} z_{vc_a}'^{iw} & \text{if } i = v \\ 0 & \text{if } v \in H \\ -z_{vc_a}''^{iw} & \text{otherwise.} \end{cases} \tag{18}$$

$$\forall s \in S : \sum_{s' \in S} y_{ss'} = 1 \tag{19}$$

$$\forall s, s' \in S : y_{ss'} \leq x_{s'} \tag{20}$$

$$\forall i \in V, s \in S : \\ \sum_{j \in V} f_{ij}^{vwc_a} - f_{ji}^{vwc_a} = \begin{cases} 0 & \text{if } i \in H \\ 1 - y_{si} & \text{if } i = s \\ -y_{si} & \text{otherwise.} \end{cases} \tag{21}$$

Eq. 1 expresses the optimization objective: to minimize the weighted sum of the average response time of workflows and the deployment cost, integrated by CAPEX and OPEX. Starting with response time, since all FNs are equal, each microservice has a constant execution time, and hence the objective is to minimize workflow latency, as defined in Eq. 2. Workflow latency includes both the latency of application traffic flows (i.e., traffic generated by requesting microservice execution) and the control latency of switches traversed by the application flows related to the workflow. With respect to the cost, CAPEX is defined in Eq. 3 as the sum of the CAPEX of used switches, FNs and SDN controllers. OPEX, defined at Eq. 4, integrates the energy consumption of used switches and controllers as well, while the energy consumption of FNs is integrated as dependent on the resource consumption of the microservices deployed in it. Nonetheless, to define them both, we need to know if a switch is being used or not, which is defined in Eq. 5.

Moving to constraints, Eq. 6 states that each microservice request in a workflow can only be fulfilled once. Eq. 7 ensures that only FNs that are actually placed can execute microservices. Eq. 8 states that the total RAM consumed by the microservices executed in an FN cannot be higher than the available RAM of the FN. Eq. 9 enforces link capacity. Eqs. 10-18 adapt the classic flow constraints to the traffic flows generated by workflows. Eq. 19 makes sure only one controller is assigned to each switch. Eq. 20 states that only placed controllers can be assigned to switches. Finally, Eq. 21 adapts the flow constraints to control flows.

This formulation, along with a defined scenario, can be used as an input to an software MILP solver to obtain a solution with optimal placements for SDN controllers, FNs and microservices, as well as optimal information routing, according to the optimization objective or objectives selected.

IV. Performance evaluation

The objective of this section is the evaluation of MO-SFO over a smart city case study using different topologies.

A. *Evaluation environment*

The evaluation of MO-SFO has been performed over emulated testbeds based on the case study presented in Sec. II. These emulated environments have been created using the Kathará emulation framework [12], which allows for the creation of emulated SDN networks using Docker containers. Regarding the networking dimension, the switches of the emulated networks are based on OpenVSwitch [13], while the SDN controller software is Faucet, a controller made for enterprise usage [14]. Moreover, the network conditions have been emulated using Linux's traffic scheduler through the *tc* command. The evaluation has been performed on topologies created using the Erdös-Rényi model for graph generation [15], transforming the nodes with a degree of one into IoT devices, while the rest are left as SDN switches. The tests have been performed over two topologies: the *Small* topology (7 SDN switches, 6 IoT devices), and the *Large* topology (20 SDN switches, 18 IoT devices), based on the topology sizes from [3]. On the application dimension, all the five microservices defined in the case study have been emulated using the official OpenVINO Docker image from Intel [1], which includes real implementations for all of them, requiring approximately 1.25 GB of RAM each. IoT devices are emulated through Alpine Linux-based containers that stream the videos provided by Intel for their use with OpenVINO [2]. Each IoT device requested the execution of a single workflow 10 times, which are used on the Docker image to measure the average response time. On the computing dimension, each FN has the specifications of a PICO-TGU4, a device for intensive domains with 32 GB of RAM. The emulated environments have been executed in an AWS $c1.xlarge$ instance.

The version of MO-SFO used for the evaluation applies the formulation from Sec. III, through the use of the MILP solving software Gurobi, to the smart city scenario. MO-SFO has been executed in a computer with an Intel i7-8565U CPU and 16 GB of RAM. Three values of α were tested in each topology: 1, 0.5 and 0, which are labeled under the objective they represent (*Response time*, *Trade-off* and *Cost*, respectively). In these six scenarios, different analyses have been performed. First, an analysis to evaluate the cost of the deployment proposed by MO-SFO in each scenario is performed. In this analysis, the overall cost of each scenario, including the cost of each of the elements (e.g., SDN controllers, FNs) is evaluated, including the CAPEX and the OPEX over 5 years, which is a common amortization period [1]. Moreover, the distributions of the response time obtained in the emulated environments are also analyzed, assessing both the performance of MO-SFO solutions and allowing for the evaluation of the cost-effectiveness achieved by each objective.

B. *Performance analysis*

In the first analysis, depicted by Fig. 2, the cost of the deployment during 5 years on each of the six scenarios is compared, as well as the contribution of each element to the total cost. Moreover, four elements are considered part of the costs: the SDN controllers (including their CAPEX and OPEX), the CAPEX of FNs, the OPEX due to the microservice execution, and the SDN switches. The first conclusion that can be drawn from the figure is that the main source of cost is the energy consumption of the microservices, accounting for 11,845€ (65-77% of the total cost, 6.49€/day) in the small topology and 34,123€ (67-79%, 18.69€/day) in the large topology, 2.88× as much as in the small topology. Moreover, the cost due to microservice execution is the same across all objectives, due to the fact that the same number of microservice replicas were deployed in all cases. The deployment of the network infrastructure, i.e., SDN switches, is also a very important element in terms of cost, representing a total of 16-30% of the cost in the small topology, and a 13-35% of the large topology costs. Furthermore, the large topology has an overall higher cost than the small topology, and the response time objective requires for more costly deployments. The cost and trade-off objectives place a single controller in the small topology, while the response time objective places two. Nonetheless, three are placed in the large topology, regardless of the objective. Finally, the optimal cost objective achieves the best cost, while the multi-objective trade-off achieves a slightly (3-6%) higher cost. The response time objective, however, has the highest costs, 20-23% higher than the cost objective.

Fig. 3 details the distribution of the response time in each of the scenarios using a box plot, in which the medians are drawn as white lines in each box, and the means are depicted

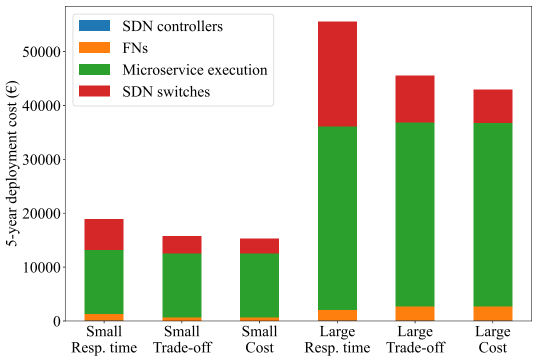

Figure 2: Deployment cost of each scenario, including CAPEX and OPEX during 5 years.

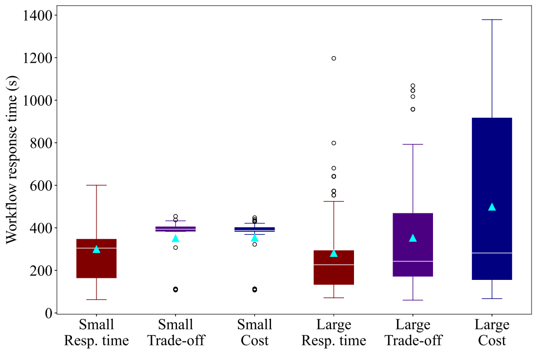

Figure 3: Distribution of response time in each scenario

[1]https://hub.docker.com/r/intel/video-analytics-serving
[2]https://github.com/intel-iot-devkit/sample-videos

by cyan triangles. These response times are the result of 10 requests of analysis over a 1-minute-long video, using the multiple microservices of the application (e.g., audio recognition, vehicle classification), and thus, the response time cannot be less than 60 seconds. Moreover, due to incompatibilities, GPU acceleration was also disabled in the emulated testbed. Starting with the small topology, the response time objective achieves an average response time of 299 seconds, a median response time of 304 seconds and a standard deviation of 163 seconds. These are the overall lowest response times for the topology, as the trade-off and cost objectives have similar mean response times, of 351 and 353 seconds (17% and 18% higher), respectively. While the median response time in the trade-off objective is slightly higher than in the cost objective, it is important to note the interquartile range is lower in the former (277 to 425 seconds) than in the latter (282 to 424 seconds), i.e., a *normal* IoT device experiences response times 1% lower if the trade-off multi-objective version is selected. Moving to the large topology, the response time objective exhibits the best response times again, with an average of 281 seconds, a median of 226 seconds, and a standard deviation of 224 seconds. Overall, the response times are lower than in the small topology, albeit the higher standard deviation indicates a larger spread. Similarly, the trade-off objective achieves an average of 352 seconds (57% higher) and a standard deviation of 276 (22% higher), whereas the cost objective exhibits an average of 499 seconds (78% higher) and an standard deviation of 413 (84% higher). On average, the trade-off multi-objective version exhibits 11% lower response times than the single-objective cost alternative, while they are a 37% higher than the response time-focused alternative.

Finally, if the results in terms of both cost and response time are considered, the multi-objective trade-off version is the most cost-effective, as it achieves a very similar deployment cost as the single-objective version, and lower response times. This phenomenon is due to the fact that the cost objective does not consider response time at all, and simply makes sure the application is able to function, whereas the trade-off objective arranges the application to optimize the response time while constraining the deployment to minimize its cost.

V. Conclusions and future work

The interactions between the physical and computing world brought by IoT make it interesting to apply its potential to intensive domains. However, the digitalization of critical processes comes with the cost of high QoS requirements for associated IoT applications. Furthermore, the QoS of intensive IoT applications depends on the application, computing and networking dimensions, as well as in the interplay between them, and is often multi-objective in nature. Optimizing these dimensions to achieve the required QoS calls for solutions able to consider the coupling between all three dimensions and trade off multiple QoS metrics to cater to specific IoT applications. In this paper, we presented MO-SFO, a multi-objective solution able to optimally place SDN controllers, FNs and microservices for intensive IoT environments. MO-SFO has been evaluated in an emulated smart city case study, focusing on the trade-off between the response time and deployment cost objectives. In the future, we expect to allow MO-SFO to use alternative multi-objective optimization algorithms, such as genetic algorithms, to improve the optimization times and provide a Pareto front. Finally, we also expect to integrate MO-SFO with state-of-the-art orchestrators, such as Kubernetes.

Acknowledgements

This work has been partially funded by MCIN/AEI/10.13039/501100011033 and by the European Union "Next GenerationEU /PRTR", by the Ministry of Science, Innovation and Universities (projects TED2021-130913B-I00, PDC2022-133465-I00), by the by the Cap4IE project (0786_CAP4ie_4_P) funded by the Interreg V-A España-Portugal (POCTEP) 2014-2020 program, by the Regional Ministry of Economy, Science and Digital Agenda of the Regional Government of Extremadura (GR21133) and the European Regional Development Fund, and by the Valhondo Calaff Institution.

References

[1] S. Ketu and P. K. Mishra, "A contemporary survey on iot based smart cities: Architecture, applications, and open issues," *Wireless Personal Communications*, pp. 1–49, 2022.

[2] J. L. Herrera, P. Bellavista, L. Foschini, J. Galán-Jiménez, J. M. Murillo, and J. Berrocal, "Meeting stringent qos requirements in iiot-based scenarios," in *IEEE Global Communications Conference*, 2020, pp. 1–6.

[3] J. L. Herrera, J. Galán-Jiménez, P. Bellavista, L. Foschini, J. Garcia-Alonso, J. M. Murillo, and J. Berrocal, "Optimal deployment of fog nodes, microservices and sdn controllers in time-sensitive iot scenarios," in *IEEE Global Communications Conference*, 2021, pp. 1–6.

[4] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: State of the art and research challenges," *Computer*, vol. 40, no. 11, pp. 38–45, 2007.

[5] B. Choudhury, S. Choudhury, and A. Dutta, "A Proactive Context-Aware Service Replication Scheme for Adhoc IoT Scenarios," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1797–1811, dec 2019.

[6] P. Bellavista, J. Berrocal, A. Corradi, S. K. Das, L. Foschini, and A. Zanni, "A survey on fog computing for the Internet of Things," *Pervasive and Mobile Computing*, vol. 52, pp. 71 – 99, 2019.

[7] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How Can Edge Computing Benefit from Software-Defined Networking: A Survey, Use Cases, and Future Directions," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4, pp. 2359–2391, 2017.

[8] T. Das, V. Sridharan, and M. Gurusamy, "A Survey on Controller Placement in SDN," *IEEE Communications Surveys & Tutorials*, pp. 472–503, 2019.

[9] J. L. Herrera, J. Galán-Jiménez, J. García-Alonso, J. Berrocal, and J. M. Murillo, "Joint optimization of response time and deployment cost in next-gen iot applications," *IEEE Internet of Things Journal*, 2022.

[10] Intel, "OpenVINO Toolkit," 2019. [Online]. Available: https://software.intel.com/en-us/openvino-toolkit

[11] Open Networking Foundation, "OpenFlow Switch Specification 1.3.0," 2013. [Online]. Available: https://tinyurl.com/openflow-v13-spec

[12] M. Scazzariello, L. Ariemma, and T. Caiazzi, "Kathará: A lightweight network emulation system," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–2.

[13] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar *et al.*, "The design and implementation of openvswitch," in *12th USENIX symposium on networked systems design and implementation (NSDI 15)*, 2015, pp. 117–130.

[14] J. Bailey and S. Stuart, "Faucet: Deploying sdn in the enterprise," *Communications of the ACM*, vol. 60, no. 1, pp. 45–49, 2016.

[15] P. Erdös and A. Rényi, "On random graphs i," *Publ. math. debrecen*, vol. 6, no. 290-297, p. 18, 1959.