Deploying Next Generation IoT Applications Through SDN-Enabled Fog Infrastructures

Deploying Next Generation IoT Applications Through SDN-Enabled Fog Infrastructures

Juan Luis Herrera*
Supervised by Javier Berrocal and Juan M. Murillo
*Department of Computer Systems and Telematics Engineering, University of Extremadura, Spain.
jlherrerag@unex.es

***Abstract*—The next generation of Internet of Things (IoT) applications automates critical, real-world processes from domains such as industry or healthcare. These applications have very strict Quality of Service (QoS) requirements. To meet these requirements, in recent years, the deployment of the application services can be performed not only in the cloud, but also through the fog to enhance the QoS, as well as for the use of programmable, Software-Defined Networks to optimize the QoS of their communications. However, the application and the network dimensions have been addressed separately, which leads to sub-optimal QoS in these critical applications. In this paper, we present the proposal of a framework to optimize the deployment of next-gen IoT applications through the fog that optimizes both, application and network, in a single effort.**

***Index Terms*—Fog computing, Internet of Things (IoT), Software-Defined Networking (SDN)**

I. Introduction

The advent of the Internet of Things (IoT) paradigm allows computer applications to automate and computerize real world processes. The next generation of IoT applications will apply this computerization to intensive domains, such as industry or healthcare [1]. Nonetheless, these applications require a high Quality of Service (QoS) [1], reflecting the criticality of the real-world processes they automate. These QoS requirements can span over different, or even multiple, QoS attributes, such as reliability or performance.

Traditionally, the most popular paradigm for IoT application deployment is cloud computing [2]. However, the large distance between IoT devices and cloud data centers is often reflected on its QoS (e.g., as a high latency), and hence it may be difficult to meet the QoS requirements of next-gen IoT applications in pure cloud deployments [2]. Therefore, paradigms such as fog computing, in which the application is deployed to servers closer to IoT devices, are more suitable for these applications [2]. Moreover, next-gen IoT applications are often implemented as a set of loosely-coupled *microservices* that collaborate to perform the application's functionalities [3]. Each of these microservices can be deployed independently or along with others, and it can also be replicated, allowing for the IoT application to be deployed through the complete cloud-to-thing infrastructure. Nonetheless, the placement of each microservice within the infrastructure affects the QoS of the application. The problem of placing these microservices to optimize the application's QoS is known as the Decentralized Computation Distribution Problem (DCDP) [3].

One of the main causes of the DCDP is the network fabric's QoS (e.g., latency), which directly impacts the communications between devices. Therefore, it is also desirable to optimize the network's QoS with techniques such as routing optimization [3]. Software-Defined Networking (SDN) is a paradigm that allows networks to be programmed by centralizing the control plane in the figure of SDN controllers. SDN enables, among other techniques, for the optimization of the communications' QoS through specific routing optimization. However, SDN switches must communicate with the SDN controller to retrieve their expected behavior in the form of rules [4]. These communications affect the QoS of the network fabric, but are themselves affected by the network fabric's QoS. Therefore, it is key to place SDN controllers in a manner that optimizes the network's QoS, i.e., to solve the Controller Placement Problem (CPP) [4].

Both the DCDP and the CPP are inherently related. On the one hand, the DCDP is partially caused by the QoS of the communications between microservices, which depends on where SDN controllers are placed, and thus, different controller placements may lead to different microservice distributions [3]. On the other hand, the CPP heavily depends on the sources and targets and of the traffic demands that are routed through the network [4], demands that are generated by the communications between microservices, and that may vary depending on the microservice distribution. Therefore, we conclude that both the DCDP and the CPP must be solved as part of a single optimization effort able to control all four, microservice replication, microservice placement, routing optimization and SDN controller placement. The objective of this project is the proposal of the Distributed Application Deployment Optimization (DADO) framework, able to solve both the DCDP and the CPP, optimizing the QoS of the next generation of IoT applications.

II. DADO framework

The DADO framework allows developers to optimize the QoS of their IoT applications in two key stages of their lifecycle, optimally deploying the application at design-time

This work has been partially funded by the project RTI2018-094591-B-I00 (MCI/AEI/FEDER,UE), the 4IE+ Project (0499-4IE-PLUS-4-E) funded by the Interreg V-A España-Portugal (POCTEP) 2014-2020 program, by the Department of Economy and Infrastructure of the Government of Extremadura (GR18112, IB18030), by the Valhondo Calaff institution, and by the European Regional Development Fund.

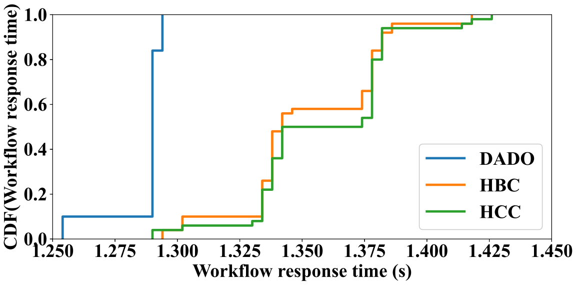

Figure 1: Empirical CDF of workflow response times.

and adapting the deployment to environmental changes at execution-time. In DADO, QoS is the *set of characteristics of a service that bear on its ability to satisfy the needs of its user* [5]. This definition includes technical and non-technical QoS, e.g., performance, availability or deployment cost. Therefore, DADO is envisioned as framework that provides an extensible model for QoS optimization. DADO provides multiple built-in QoS attributes to be optimized, and the possibility for users to define their own attributes.

DADO requires three inputs: the application architecture, the computing devices from the infrastructure and the network fabric that connects them. These inputs are plans and estimations if DADO is leveraged at design time, or their monitored versions at execution time. Along with these inputs, the developer must also specify the QoS attributes to be optimized. The output received by the developer consists on a deployment plan that optimizes the QoS.

Applications in DADO are modeled as sets of independent modules or *microservices*, that define their technical characteristics. It is important to note that each microservice should be defined only once, as DADO will automatically replicate the microservice if needed. These microservices are often not requested independently, e.g., after data has been processed by a microservice, it will probably be stored by another microservice. Therefore, the application is also defined by the interactions among microservices or *workflows*: pipelines of multiple microservices in which the output of a microservice is the input of the next one. The workflow definition includes the valid workflows for the application as well as the estimated or real number of requests per workflow, allowing DADO to adapt the application's deployment to its load.

Each of the microservices requested by these workflows and instanced by DADO needs to be deployed at a computing device (e.g., fog node, cloud server). DADO models all the computation-capable devices as a set of technical characteristics, normally analogous to those of microservices (e.g., microservices consume RAM, and devices have a total RAM). DADO does not directly model whether a device is a cloud, fog, edge or mist device, thus allowing the developer to define an infrastructure with arbitrary layers.

All these devices are connected to each other using a network fabric, modeled as a graph. Each of the vertices of the graph is either a computing device, or an SDN switch (and thus, a potential placement for an SDN controller [4]), while each edge represents a link between devices or switches. Furthermore, each of the switches and links also define their own technical characteristics (e.g., latency, capacity).

Using these inputs, DADO generates an optimal deployment plan for the application. This deployment plan details how many replicas of each microservice need to be deployed, where to deploy each of them, how many SDN controllers need to be set up, where to place each controller, which controller should each switch communicate with, and the paths followed by application and SDN control traffic. All these decisions are taken by DADO to optimize the QoS, considering the effects of each decision in the rest (e.g., deploying two microservices that communicate in different machines creates a traffic flow, which alters the optimal controller placement).

Currently, a version of DADO based on Mixed-Integer Linear Programming (MILP) is under development [3]. Fig. 1 depicts an empirical CDF of the preliminary results from this prototype version, comparing DADO with the usage of graph metrics such as Highest Closeness Centrality (HCC) or Highest Betweenness Centrality (HBC) to take the deployment decisions. As seen in Fig. 1, the slowest workflows deployed using DADO have similar response times to the fastest workflows deployed using HBC and HCC. However, MILP does not scale well on large infrastructures, taking over 10 hours to optimize the deployment plan [3].

III. Conclusions and future work

The next generation of the Internet of Things will computerize and automate critical real-world processes, and their criticality will be reflected as high QoS requirements for next-gen IoT applications. Meeting the QoS requirements will require the applications to be optimally deployed in a fog infrastructure, communicated by a SDN network fabric. The objective of this PhD thesis is to build DADO, a framework to optimize the application's deployment, and a possible enabler for the next generation of IoT.

DADO is currently a work in progress, and thus, there are limitations to what this paper currently presents. The current MILP-based prototype takes a long amount of time to optimize the deployment of large scenarios [3], and therefore is only useful at design time. Accordingly, we expect to develop heuristic solutions that allow for the optimization to be performed at execution time, allowing DADO to adapt the deployment to environmental changes.

References

[1] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A Survey on Industrial Internet of Things: A Cyber-Physical Systems Perspective," *IEEE Access*, vol. 6, pp. 78 238–78 259, 2018.

[2] 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.

[3] J. L. Herrera, J. Galán-Jiménez, J. Berrocal, and J. M. Murillo, "Optimizing the response time in sdn-fog environments for time-strict iot applications," *IEEE Internet of Things Journal*, 2021.

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

[5] ITU-T, *Definitions of terms related to quality of service*, International Telecommunication Union Telecommunication Standarization Sector Std. E.800, Rev. 09/2008.