# Virtual Environment for Evaluating the QoS of Distributed Mobile Applications

Sergio Laso*, Javier Berrocal*, Pablo Fernández†, Antonio Ruiz-Cortés† and Juan M. Murillo*
*University of Extremadura
Cáceres, Spain
Email:{slasom, jberolm, juanmamu}@unex.es
†I3US Institute, SCORE Lab. Universidad de Sevilla
Sevilla, Spain
Email:{pablofm, aruiz}@us.es

***Abstract*—The increasing capabilities of end devices has led to a wider distribution of the computation and the massive deployment of distributed mobile applications. The success of these applications is highly dependent on the Quality of Service they provide. This quality is especially difficult to assess due to the large number of entities involved and their heterogeneity. Current tools are usually focused on evaluating the QoS provided by a single entity. Nevertheless, the QoS of distributed applications not only depend on the QoS of each entity, the interactions among entities has also to be evaluated. Therefore, new techniques are required to perform a comprehensive evaluation of the expected QoS of these applications before their production deployment. This paper presents a framework, called Perses, for launching virtual environments to simulate and test the execution of distributed mobile applications. This simulation provides results of the QoS achieved. Moreover, the framework has been integrated into a DevOps methodology in order to automate its execution.**

***Video showcase*— https://youtu.be/wpIApe_sPFE**

***Index Terms*—Distributed Mobile Applications; Quality of Service; DevOps; Virtual Environment**

## I. Introduction

The success or failure of any mobile application depends on many dimensions such as advertising, virality or the Quality of Service (QoS) provided [1]. The QoS is a key dimension that can be controlled and evaluated during the development and operation of the application [2]. A poor quality will lead users to quickly reject the application.

These applications have been usually developed using a client-server architectural style to achieve a maximum QoS. The most demanding computing components were usually deployed in cloud environments due to their large computing and storage capabilities. The basic components were deployed at the client-side, especially the user interface, as their computing requirements are usually low and can be executed by almost any mobile device with a minimum set of functionalities.

During the last few years, mobile devices computing and storage capabilities have significantly increased [3], leading to the emergence of new paradigms focused on exploiting them, such as Human Microservices [4], Mist Computing [5] or Federated Learning [6]. They allow developers to on-load some computing functionalities on mobile devices to further improve the QoS and be more competitive at the enterprise level. For environment, such as Healthcare or Industry 4.0, with stringent QoS requirements (such as response time or latency) this on-load of functionalities is crucial [7]. These paradigms allow developers to locally store and compute the sensed information in order to be consumed by local applications or nearby devices.

In client-server applications, QoS-sensitive functionalities are deployed in cloud environments, in which cloud providers can guarantee a minimum quality of the infrastructure and also scale it up if more resources are required. In such a context, QoS is mainly evaluated at the server-side, whilst the tests at the mobile-side are typically focused on the user experience (functionality and adaptability of graphic elements, navigation through different screens, etc.).

In distributed applications, both the cloud and the mobile-side have computing and storage components. Moreover, QoS-sensitive functionalities can be deployed at both sides. Therefore, the QoS evaluation must combine the integrated assessment of the server-side, the mobile-side, and the interaction among devices. Only evaluating the three dimensions in an integrated would allow one to predict with adequate accuracy the expected QoS. In addition, these are usually highly heterogeneous environments (different mobile devices, with different capabilities, etc.), being even more difficult to predict the expected QoS. Therefore, technologies and methods are required to evaluate the QoS of distributed applications in heterogeneous environments.

Currently, there are different environments that can be used to evaluate the behaviour of non-distributed mobile applications in heterogeneous environments, such as AWS Device Farm [8] and Azure App Center Test [9]. They allow developers to create customized environments composed of devices with different hardware, versions of the operating system, configuration, etc. However, these platforms focus on evaluating and launching user interface tests (using Appium,[1] Espresso,[2] etc.). Therefore, they are not designed to measure the QoS attributes integrating the cloud and the mobile-side,

[1]http://appium.io/
[2]https://developer.android.com/training/testing/espresso

and their interactions.

Likewise, there are some research efforts towards the evaluation of the QoS of distributed applications. In [10] the authors propose a framework for the development and evaluation of distributed systems based on components for heterogeneous scenarios, taking into account mobile and fixed networks. However, this evaluation is limited to applications developed using the proposed framework. Therefore, approaches are needed that can also evaluate any application that can be deployed in the targeted devices. In [11], the authors propose a system for testing the usability and performance of Android mobile applications with Virtual Reality. This system simulates different network and mobility behaviours in order to evaluate applications in close to real environments. However, this approach is focused on manually testing the application using a single device, which hinder the evaluation of distributed applications deployed on heterogeneous devices. Furthermore, the tests carried out are focused on user experience.

In this paper, we present a framework, called Perses [12], to evaluate the QoS of distributed mobile applications. To that end, it allows the definition and deployment of customized virtual environments, with multiple heterogeneous virtual devices, in which developers can simulate the deployment of distributed mobile application and launch performance tests to evaluate different QoS attributes, namely computing time, response time or latency.

Moreover, this framework has been fully integrated into a DevOps methodology [13] integrating the virtual environment deployment and the tests execution in a continuous integration pipeline, reducing the effort of evaluating the QoS before deploying the application into production.

The rest of the paper is structured as follows. Section II explains the characteristics of Perses. Subsection II-A details its architecture and Subsection II-B describes a demo use case. Finally, the conclusions are presented in Section III.

## II. Perses

In distributed mobile applications, the holistic evaluation of the QoS is crucial to predict their success. In these applications, some components are deployed on the server and others on the mobile-side. If the QoS is evaluated independently in each entity and, then, aggregated, the results can be far from reality. The interactions among entities, and devices heterogeneity must also be taken into account.

Perses is a framework that allows developers to easily deploy a virtual environment with multiple heterogeneous virtual devices to evaluate the QoS of a distributed application. To use it, one only has to define a *configuration file* where the desired QoS and the characteristics of the environment where it is to be tested are indicated.

Perses is fully integrated into a DevOps methodology, being able to automate all the different steps that should be executed during the evaluation process, from the deployment of the environment to the collection and checking of the results obtained to determine whether or not the application achieves the desired quality. This is an important feature for development companies since they can easily integrate Perses in their continuous integration pipeline.

Perses is fully scalable, allowing one to evaluate applications with a large number of virtualised devices. It also allows customizing their characteristics such as hardware, OS, etc, being able to easily test different configurations.

### A. Architecture

Perses is a framework based on different tools to facilitate the creation and deployment of the virtual environments with heterogeneous virtual mobile devices simulating a real deployment environment. This section explains the different modules of this framework, how they have been developed using existing tools, and how they have been integrated.

Figure 1 shows a general diagram of Perses's architecture. Perses consists of two components: *Perses Launcher*, focused on the definition and configuration of the virtual environment, and *Perses Virtual Environment*, responsible for creating the heterogeneous environment and executing the defined tests. Each component has different modules that are explained below:

- **Setup:** this module allows developers to create a *configuration file* that is used for defining the characteristics of the virtual environment to be deployed, the tests to be launched and the desired QoS attributes to be evaluated during the test execution.
- **Deployment:** this module takes as input the *configuration file* and is responsible for creating and deploying the whole virtual environment. To that end, an abstraction layer has been defined that encapsulates Terraform [14] –a framework with a high-level language that can be used to define the deployment infrastructure of an application for cloud providers. This module, in addition to deploying the cloud infrastructure, orchestrates the virtual environment obtaining and installing the necessary resources, creating the virtual mobile devices and deploying the distributed mobile application.
- **Tests Execution:** this module is in charge of launching and analysing the tests defined in the *configuration file*. Two different kinds of tests can be executed: performance and user interface tests. The performance tests evaluates the QoS of the application. To that end, APIPecker [15] – a simple API performance tester in which different attributes can be defined (concurrent users, iterations and delay) – is used. For the user interface tests, Perses allows developers to execute UI tests developed with Espresso. After launching the tests, this module collects, analyses, and aggregates the system log of every virtual devices and the general performance metrics to determine if the desired QoS is achieved.
- **CI Manager:** this module is in charge of the integration with DevOps. It automates the whole deployment process of Perses and the execution of the different modules. This automation is carried out through a workflow defined with GitHub Actions [16]. Once the process is completed, the test results are provided to the developers.

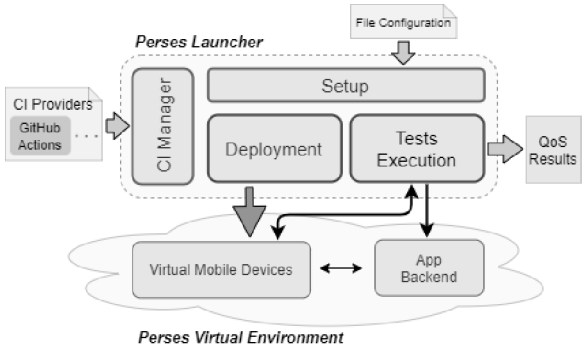


Fig. 1. General diagram of Perses.

### B. Demo Use Case

In this demonstration, we will show attendees how the Perses framework works by evaluating the QoS of a distributed mobile application. For this purpose, we will evaluate an application to help curb the Covid-19 pandemic. This application provides the contagion risk of a person.

The application is made up of a server and a mobile-side. The server-side is the access point for users to obtain the risk percentage and, therefore, is also responsible for calculating this risk. The mobile-side is in charge of saving the traces of each user and providing on-demand to the cloud the places/regions in which they have been on.

To know the contagion risk, the user provides the regions where he has been on during the last three days to the server. Then, the server compares them with the regions visited by recently diagnosed positive users in order to calculate the infection risk.

To make the application suitable for the production deployment, an average *response time* of no more than two seconds is required. For this purpose, Perses will create a virtual environment with a set of devices simulating a real situation to evaluate the QoS. The environment will be composed of virtual mobile devices that already have a location history stored. Some of them will act as users with Covid-19 to evaluate the most adverse cases where a user has moved through areas where there have been several positive people.

## III. Conclusion

QoS is one of the key dimensions for the success or failure of any application. In recent years, the capabilities of end devices have increased considerably, leading to the emergence of new paradigms focused on the exploitation of these capabilities. This gives rise to distributed applications where both the cloud and the mobile-side can have computing and storage components with the aim of further improving the QoS. For environments with strict QoS attributes this distribution of functionalities is crucial. Moreover, with the advance of IoT, these applications are going to be more and more common. Nevertheless, this quality depends on different entities and their interactions, which should be carefully analyzed.

In this paper, we presented a framework for the deployment of a customized virtual environment for assessing the QoS of distributed mobile applications. We currently work on analysing the impact of the use of this framework. We are also working on using Perses to evaluate different architectural styles and which one achieves a better QoS.

## Acknowledgment

This work has been partially funded by the projects RTI2018-101204-B-C21 and TIN2015-70560-R (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 RCIS network (TIN2016-81978-REDT), by the Department of Economy and Infrastructure of the Government of Extremadura (GR18112, IB18030), by the Government of Andalusian (US-1264651) and by the European Regional Development Fund.

## References

[1] H. J. Kim, D. H. Lee, J. M. Lee, K. H. Lee, W. Lyu, and S. G. Choi, "The qoe evaluation method through the qos-qoe correlation model," in *2008 Fourth International Conference on Networked Computing and Advanced Information Management*, vol. 2, 2008, pp. 719–725.

[2] W. N. Picoto, R. Duarte, and I. Pinto, "Uncovering top-ranking factors for mobile apps through a multimethod approach," *Journal of Business Research*, vol. 101, pp. 668–674, 2019.

[3] J. Berrocal, J. Garcia-Alonso, C. Vicente-Chicote, J. Hernández, T. Mikkonen, C. Canal, and J. M. Murillo, "Early analysis of resource consumption patterns in mobile applications," *Pervasive and Mobile Computing*, vol. 35, pp. 32 – 50, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574119216300797

[4] S. Laso, M. Linaje, J. Garcia-Alonso, J. M. Murillo, and J. Berrocal, "Artifact abstract: Deployment of apis on android mobile devices and microcontrollers," in *2020 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2020, pp. 1–2.

[5] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, 2019.

[6] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[7] S. Shukla, M. F. Hassan, L. T. Jung, and A. Awang, "Architecture for latency reduction in healthcare internet-of-things using reinforcement learning and fuzzy based fog computing," in *International Conference of Reliable Information and Communication Technology*. Springer, 2018, pp. 372–383.

[8] A. W. Service, "Aws device farm," https://aws.amazon.com/device-farm/, (Accessed on 28/10/2020).

[9] M. Azure, "App center test," https://docs.microsoft.com/en-us/appcenter/test-cloud/, (Accessed on 28/10/2020).

[10] B. Richerzhagen, D. Stingl, J. Ruckert, and R. Steinmetz, "Simonstrator: Simulation and prototyping platform for distributed mobile applications," in *The 8th EAI International Conference on Simulation Tools and Techniques (ACM SIMUTOOLS 2015)*. IMDEA Networks Institute Publications Repository, 2015.

[11] T. Amano, S. Kajita, H. Yamaguchi, T. Higashino, and M. Takai, "Smartphone applications testbed using virtual reality," in *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2018, pp. 422–431.

[12] Perses-org, "Perses," https://github.com/perses-org/perses, note = (Accessed on 28/10/2020).

[13] L. Bass, I. Weber, and L. Zhu, *DevOps: A software architect's perspective*. Addison-Wesley Professional, 2015.

[14] HashiCorp, "Terraform," https://www.terraform.io/, (Accessed on 28/10/2020).

[15] P. Fernandez, "API Pecker," https://www.npmjs.com/package/apipecker, (Accessed on 28/10/2020).

[16] Github, "Github actions," https://github.com/features/actions, (Accessed on 28/10/2020).