Continuously Testing Distributed IoT Systems: An Introduction to the Marvis Framework

Arne Boockmeyer[1], Jossekin Beilharz[1], Philipp Wiesner[2], Dirk Friedenberger[1+], and Lauritz Thamsen[2]

[1] Hasso Plattner Institute, University of Potsdam, Germany
{firstname.lastname}@hpi.de
[+]dirk.friedenberger@guests.hpi.de
[2] Technische Universität Berlin, Germany
{wiesner, lauritz.thamsen}@tu-berlin.de

Abstract. Testing applications in the area of the Internet of Things (IoT) is a difficult task due to the variety of devices and their complex communication behavior. Especially the communication capabilities are heavily influenced by the physical environment of the system.
To continuously test IoT applications, a staging environment providing representative conditions of a production environment is necessary. Marvis is a testing framework that orchestrates hybrid testbeds and co-simulated domain environments to provide such a staging environment. It features a central network simulation based on ns-3 to connect virtual and physical nodes executing the application under test.
This tutorial is recommended for all researchers, developers, and testers in the area of IoT or distributed applications as well as developers of simulation systems. For this target audience, Marvis provides a flexible and comprehensive framework to evaluate these applications.

Keywords: Internet of Things · Co-Simulation · Testing · Distributed Applications · Cyber-Physical Systems · Edge Computing

1 Introduction

Software development for widely distributed systems is becoming increasingly important with the rise of the Internet of Things (IoT) and fog and edge computing. At the same time, testing these systems is still given little consideration despite the critical nature of many IoT use cases.

In cluster and cloud software, on the other hand, continuous software testing is fairly well understood and widely used. Best practices of testing cloud software include testing in staging environments that replicate the production environment as close as possible. To model such environments for the testing of IoT-Apps is difficult due to two groups of reasons: the complex hardware landscape and the dependence of software behavior on the environment, due to the very nature of many IoT-Apps that integrate with the environment through sensors and actors.

This tutorial covers Marvis, a testing framework for IoT applications that works towards solving these issues through the integration of virtual and physical IT resources with domain-specific environment simulators and testbeds. The concept behind Marvis was first introduced at IEEE PerCom 2021, where it was awarded as the best work-in-progress paper [2]. It builds on the previously developed concepts of Hatebefi [3] and Héctor [1].

The tutorial explains and guides the audience through the different aspects and functionalities of Marvis. Therefore a mixture of theoretical input and hands-on sessions are planned. In the end, the audience will be able to use Marvis for their experiments in the area of testing IoT or distributed applications in general.

1.1 Target Audience

The target audience of this tutorial includes but are not limited to:

- Developers and testers of IoT applications or distributed systems in general
- Researchers in the area of IoT or distributed systems in general
- Developers of simulation systems

Participants should:

- Be familiar with the challenges of testing IoT applications or distributed applications in general
- Have a basic understanding of virtualization and simulation techniques
- Have basic proficiency with Docker or LXD containers

Besides these theoretical requirements, there are a few technical requirements to participate in the hands-on sessions:

- Python 3.7 and a text editor/IDE
- Capability to run local Docker containers
- An Ubuntu-based system would be ideal
- Completed Marvis setup according to the documentation[3]

1.2 Length

We aim for a tutorial length of around 2 hours and 30 minutes, including a 10 minute break. Around 40% of that time will be used for presenting content and providing an overview of each chapter. This is necessary to explain the context and goals of Marvis and lowers the level for understanding the practical exercises.

The focus of the tutorial is on the practical exercises of each chapter, which in total take up around 60% of the tutorial time. During this time, the audience develops their own tests in Marvis, guided by the presenters.

[3] https://github.com/diselab/marvis#readme

2 Tutorial Outline

The tutorials consist of four chapters. The first chapter covers fundamentals for understanding the concept behind Marvis and its usage. Chapters two and three point on different functionalities of Marvis. The last chapter provides a summary, an outlook over further readings and future work, and explains how participants can contribute to Marvis.

2.1 Chapter 1: Introduction (45 minutes)

The introductory chapter briefly explains the background and motivation behind Marvis as well as the problems it solves. Furthermore, the audience learns the terminology used in the tutorial and gets a first overview of the test scenarios. They also run the first experiment in a hands-on session, which verifies that all have a working installation of Marvis.

Content:

- Challenges in testing IoT scenarios
- Goals of Marvis
- Architectural overview
- Terminology
- Outline of the following tutorial
- Check the setup of the participants to prepare for the demo sessions
- Guide through the execution of the first example

2.2 Chapter 2: Node Orchestration (30 minutes)

This chapter will assist the audience in using the different kinds of nodes supported by Marvis. The first part of this chapter shows an overview, afterwards both node types, virtualized and hardware nodes, are used in a hands-on session to learn how to execute an application in Marvis. Virtualization techniques are not explained in detail but from a user perspective.

Content:

- Overview about supported node types
- Comparison of advantages and disadvantages of the different node types for the scenarios
- Performing scaleable experiments by using virtualized nodes (esp. Docker containers)
- Developing simple Hardware-In-The-Loop scenario through connecting network interfaces into the simulation

2.3 Chapter 3: Advanced Features of Marvis (50 minutes)

This chapter demonstrates advanced features of Marvis for more extensive scenarios. Each of the three features (advanced networking, co-simulation, and fault injection) will be briefly presented to the audience. Afterward, the audience will use these features in a hands-on session to enrich scenarios.

Content:

- Usage of different types of simulated network connections (e.g. Wifi, Ethernet), their combinations and limitations
- Introduction to co-simulation with SUMO integrated in Marvis
  - Creating and loading scenarios in SUMO
  - Connecting containers or hardware nodes to SUMO elements
  - Running SUMO scenarios integrated in Marvis
- Capabilities in fault injection testing with Marvis
  - Introduction to fault injection testing
  - Usage of different targets for fault injection (network, node or simulation)

2.4 Chapter 4: Conclusions, Questions and Feedback (15 minutes)

The last chapter concludes the tutorial with a summary of the tutorial and an outlook about future work and further readings. In addition, the presenters explain how the audience can contribute to Marvis.

Content:

- Summary of the tutorial
- Questions
- Outlook for future perspectives and developments of Marvis
- Instructions on how to contribute
- Feedback on this tutorial and Marvis itself

3 Presenter Biography

Arne Boockmeyer is a research associate and Ph.D. student at the Hasso Plattner Institute, University of Potsdam. As part of his Ph.D. studies, he also teaches topics like Co-Simulation or testing of distributed applications to students and supervises projects and master theses in that area. He is one of the Marvis developers and contributed to the concepts in the early stages.

Jossekin Beilharz is a research associate and Ph.D. student at the Operating Systems and Middleware Group at Hasso Plattner Institute, University of Potsdam. He works on dependable widely distributed systems and initiated the work on Marvis.

Philipp Wiesner is a research associate and Ph.D. student in the research group on Distributed and Operating Systems at Technische Universität Berlin, Germany. His research interests include co-simulation, the digitalization of critical urban infrastructures, energy consumption, and renewable-aware computing environments.

Dirk Friedenberger is an external Ph.D. student at the Hasso Plattner Institute, University of Potsdam, and graduated computer specialist employed by the DB Systel GmbH in Frankfurt. His research interests include co-simulation and dependable distributed systems.

Lauritz Thamsen is a senior researcher at TU Berlin, where he leads the work on adaptive resource management in the research group on Distributed and Operating Systems and lectures on cloud computing, data-intensive systems, and the IoT. His research interests include dependable and adaptive distributed systems, resource management and scheduling, cloud and fog computing, as well as distributed data processing.

References

Marvis:

- Marvis code and examples: https://github.com/diselab/marvis
- Marvis documentation: https://diselab.github.io/marvis

Used Technologies and Tools:

- ns-3: https://nsnam.org
- SUMO: https://sumo.dlr.de
- Docker: https://docker.com
- LXD: https://linuxcontainers.org/lxd/introduction/
- Wireshark: https://www.wireshark.org/

Literature:

1. Behnke, I., Thamsen, L., Kao, O.: Héctor: A Framework for Testing IoT Applications Across Heterogeneous Edge and Cloud Testbeds. In: 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion. pp. 15–20. UCC Companion'19, ACM (2019)
2. Beilharz, J., Wiesner, P., Boockmeyer, A., Brokhausen, F., Behnke, I., Schmid, R., Pirl, L., Thamsen, L.: Towards a Staging Environment for the Internet of Things. In: 2021 IEEE International Conference on Pervasive Computing and Communications Workshops. p. To appear. PerCom-W'21, IEEE (2021)
3. Boockmeyer, A., Beilharz, J., Pirl, L., Polze, A.: Hatebefi: Hybrid Applications Testbed for Fault Injection. In: 2019 IEEE 22nd International Symposium on Real-Time Distributed Computing. pp. 97–98. ISORC'19, IEEE (2019). https://doi.org/10.1109/ISORC.2019.00030