On-chip Tenant-Driven Monitoring on Multi-Tenant Microcontrollers

Bastien Buil*
Orange Research
Caen, France
bastien.buil@orange.com

Abstract

Multi-tenant microcontrollers enable the secure execution of code from mutually distrusting service providers through lightweight software containerization solutions such as WebAssembly. However, service providers may not receive sufficient service level to make their applications function properly. In this paper, we propose a model to allow service providers to verify by themselves that they receive a sufficient service level by testing the execution environment at different steps of the application lifecycle.

CCS Concepts

• **Software and its engineering** → *Runtime environments*; **Virtual machines**; • **Computer systems organization** → *Embedded systems.*

Keywords

IoT, Microcontrollers, Monitoring, Cloud-to-IoT Continuum, Containers

1 Introduction

The development of containerization on microcontrollers, such as FemtoContainers [6] and Aerogel [4], enables microcontrollers to securely host services coming from multiple mutually distrusting entities. This cohabitation of software on microcontrollers creates multi-tenant microcontrollers, which are shared microcontrollers that execute applications from different entities on a single device.

Company like MicroEJ [5] uses containerization to permit the servitization of microcontrollers which consists of enabling third party to develop services for the microcontroller of another entity. The servitization allows both to personalize the device, for example through appstore with third-party applications, and to monetize the device through commissions on premium features.

In this context, service providers are tenants of the device which is managed by the microcontroller host. As they might commit to provide their users a service, sometimes with paid features, service providers should have guarantees on the well functioning of their applications. While testing applications before deployment is a first step, it is not enough for cases when the host does not provide sufficient resources and quality of services for the applications to function well.

A first solution is having an agreement between service providers and the hosts on the provided level of service. Hosts can commit on service level as runtimes, like Aerogel [4] and Polyglot Cerberos [2], integrate resource contracts to define resources accessible by an application. This can be used to prevent interference by other applications by setting limits on dedicated memory size, percentage of dedicated time for execution, peripherals access, and network.

However, such mechanisms are not yet spread among all runtimes. Additionally, service providers are vulnerable to honest but greedy hosts, which generally fulfill their commitments but may occasionally neglect some obligations to reduce costs or out of convenience. This can provoke unpredicted incidents with applications failing unexpectedly.

In the Cloud, there exist mechanisms to allow tenants to verify by themselves if commitments are respected [3, 7], but these solutions are not adapted to the constraints of microcontrollers. Some solutions exist to monitor microcontrollers such as Memfault [1], but they are not adapted to applications in multi-tenant microcontrollers.

Implementing commitment verification on multi-tenant microcontrollers involves multiple challenges. First, multi-tenant microcontrollers use containerization to isolate applications both between them and with the host system. This reduces the room for maneuver for applications to implement monitoring mechanisms. Secondly, on multi-tenant microcontrollers, the resources of the system are shared between multiple actors. Thus, applications should especially be constrained to work on microcontrollers which have small amounts of memory and processing power. This problem is amplified by the containerization overhead.

This paper proposes a 3-phase model to allow service providers to monitor provided service level on multi-tenant microcontrollers.

2 Model overview

The three-phase model works by implementing tests at the different phases of application operation on multi-tenant microcontrollers.

2.1 Integrating tests in application lifecycles

Three testing phases. The three phases of the model correspond to the deployment, the operation and the suppression of the application. First, the check-in vetting consists of deploying a container before the deployment of the target application to assess the execution environment and ensure that the execution environment provides adequate resources. Second, in-operation monitoring integrate test in the application to continuously evaluate the device during application execution and verify that the service level established in the initial phase is maintained. Finally, the check-out inspection involves deploying a container after application suppression to confirm post-usage that the device remains functional and undamaged by the application.

Tests as logic block run across microcontrollers and server. In the three phases, tests are conceived similarly and are inspired by

*Also with Cnam, Cedric.

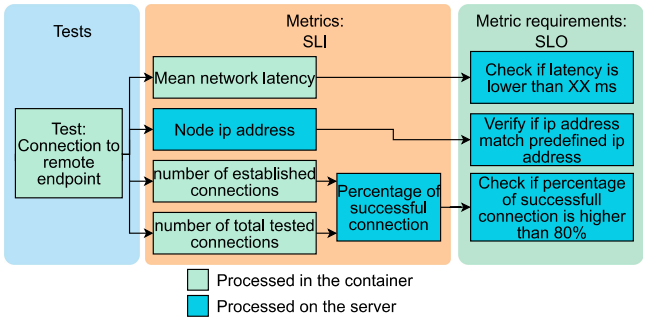

Figure 1: Example of test process and the processing of the test on both microcontrollers and server

WSLA [3]. The tests produce metrics, also called Service Level Indicators (SLI). These metrics are compared to metrics requirements, also called SLO (Service Level Objective). Tests, metrics and requirements are conceived as logic blocks split between the container and the monitoring server, as represented in Figure 1. The different blocks can run solely on microcontrollers, across both microcontrollers and server, or exclusively on the server. For example, RAM tests can be done from microcontrollers, connection tests can be done across both microcontrollers and server, and verifying availability of a service exposed on the device can be done only by the server.

Hook and actions. Tests can be triggered using hook. The test containers for check-in vetting and for check-out inspection only execute a succession of test hooks, while hooks for in-operation tests are inserted in the application code. When the metrics requirements are not met, actions are triggered on the client side or on the server side. During the vetting process, the server can stop the application deployment process. During application operation, the application can stop itself, alert its users, or contact a remote server to take proper action.

2.2 Automated instrumentation based on Service Level Agreement

To facilitate the creation of test containers for the vetting and check-out phases and the instrumentation for the in-operation phase, we propose to automatically compose tests in containers. The selection of tests can be automatic from a test database based on characteristics of the devices and the SLA required by the application.

3 Conclusion and next steps

We have proposed a three-phase model for continuously evaluating the provided service level on multi-tenant microcontrollers. The next steps for our work are to develop a proof of concept for evaluating the feasibility of the model on one type of multi-tenant microcontrollers. A major point for the evaluation will be to evaluate the overhead of having additional containers deployed for check-in vetting and check-out inspection, as well as the overhead of integrating tests in application containers.

Acknowledgments

The research leading to these results is funded by ANRT Convention Cifre n°2024/0426. This paper reflects only the authors' views.

References

[1] [n. d.]. Memfault - IoT Device Monitoring. https://memfault.com/iot-device-monitoring/.

[2] Sven Akkermans, Bruno Crispo, Wouter Joosen, and Danny Hughes. 2018. Polyglot CerberOS: Resource Security, Interoperability and Multi-Tenancy for IoT Services on a Multilingual Platform. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. ACM, New York NY USA, 59–68. https://doi.org/10.1145/3286978.3286997

[3] Alexander Keller and Heiko Ludwig. 2003. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management* 11, 1 (March 2003), 57–81. https://doi.org/10.1023/A:1022445108617

[4] Renju Liu, Luis Garcia, and Mani Srivastava. 2021. Aerogel: Lightweight Access Control Framework for WebAssembly-Based Bare-Metal IoT Devices. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*. 94–105. https://doi.org/10.1145/3453142.3491282

[5] microEJ. [n. d.]. MicroEJ Embrace IoT-Enabled Servitization. https://www.microej.com.

[6] Koen Zandberg, Emmanuel Baccelli, Shenghao Yuan, Frédéric Besson, and Jean-Pierre Talpin. 2022. Femto-Containers: Lightweight Virtualization and Fault Isolation for Small Software Functions on Low-Power IoT Microcontrollers. In *Proceedings of the 23rd ACM/IFIP International Middleware Conference (Middleware '22)*. Association for Computing Machinery, New York, NY, USA, 161–173. https://doi.org/10.1145/3528535.3565242

[7] Yinqian Zhang, Ari Juels, Alina Oprea, and Michael K. Reiter. 2011. HomeAlone: Co-residency Detection in the Cloud via Side-Channel Analysis. In *2011 IEEE Symposium on Security and Privacy*. 313–328. https://doi.org/10.1109/SP.2011.31