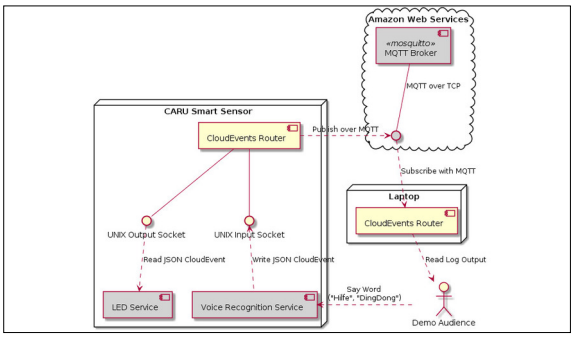

Fabrizio Lazzaretti


Linus Basig

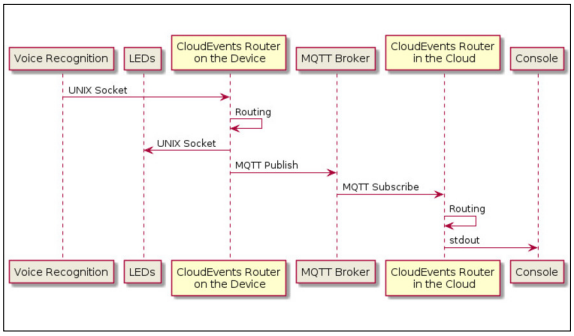
Students	Fabrizio Lazzaretti, Linus Basig
Examiner	Prof. Dr. Olaf Zimmermann
Subject Area	Networks, Security & Cloud Infrastructure
Project Partner	CARU AG, Zürich, Zürich

CloudEvents Router

The CARU Smart Sensor placed in the home of a senior citizen
CARU AG



Example deployment of the CloudEvents Router
Own presentment



The flow of a CloudEvent through the example deployment of the CloudEvents Router
Own presentment

Problem: CARU AG is an AgeTech startup with the mission to help the elderly live independently for longer. Its internet-connected device allows calling for help by voice command. In addition to this base functionality, the device is equipped with multiple sensors that enable a variety of data-driven functionalities.
The software architecture used by CARU AG is heavily event-driven. The different software components publish events whenever something notable happens. In turn, other components react to these events and publish new events themselves. The events are structured according to the CloudEvents specification of the Cloud Native Computing Foundation.
CARU AG wants to create a unified event plane where events can flow between all connected systems. These include the ones on the device, in the cloud, and potentially even from third parties. However, always broadcasting each event to all systems would be inefficient and cause security risks. We address this shortcoming by introducing the CloudEvents Router. As an implementation of the Content-Based Router pattern described in "Enterprise Integration Patterns" by Hohpe and Wolf, the router is responsible for routing the events between the different systems according to the event fields defined in the CloudEvents specification. Because of the diversity of hardware and limited resources available in the target environments, the architecture has special requirements regarding portability, resource usage, and extensibility.

Approach: As a starting point, we conducted extensive market research to assess already existing products that can support this use case. We found one solution with much potential: Apache Camel. Unfortunately, Camel does not meet all the resource requirements and misses support for one of the desired runtime environments.
To create a solution that fulfills all these requirements, we iterated between creating comprehensive architecture documentation and prototyping different aspects of the problem. After the design of the software architecture, we implemented our solution and applied pair programming along the way.

Result: We created and open-sourced a CloudEvents Router proof of concept implementation in the Rust programming language (https://github.com/ce-rust/cerk). To achieve the desired portability, we based the architecture on an event-driven Microkernel, which supports the platform-specific implementation of individual components.
The second deliverable is a report on our experience using the Rust programming language for the first time. After a frustrating beginning, we started to enjoy using it. Rust brings all the advantages of strongly typed languages plus some novel additions like its ownership model and safe memory management. However, these new concepts are also the reason for the steep learning curve. Fortunately, the exceptionally helpful error messages provided by the compiler make it easy to find one's mistakes. Finally, the tooling provided by the Rust Team is thoughtfully integrated. The creators of Rust brought together the best practices from many other programming languages and made them the default.