Software Design Document
The AHC Project

Prof Dr. Ertan Onur
Deniz Koluaçık
Osman Ufuk Yağmur
Yiğitcan Uçan
Ozan Akın
Cenk Arda Yılmaz

Contents

List of Figures

List of Tables

Revision History

Revision	Revision Date	Notes
v1.0	March 14, 2023	Final version.
v0.1	October 12, 2022	Initial draft.

Table 1: Revision history table.

1 Introduction

1.1 Purpose

The overall goal of the AHC project is to provide an open-source education and research software framework that facilitates the development of distributed algorithms on wireless networks considering the impairments of wireless channels. The framework is being used as a learning and prompt-prototyping tool. As of the preparation of this document, the users of the AHC framework include graduate students, lecturers, and researchers working in the fields of digital communication, networking and distributed computing. Furthermore, the framework is expected to be used by a wider audience including engineers. The developed framework is available to all these user groups as permissive open-source software.

1.2 Scope

- The outcome of the AHC project is the AHC framework.
- The AHC framework consists of
    1. the free and open source (FOSS) software services that provide the functionality of the framework,
    2. the reproducible infrastructure that the software services run on and/or make use of,
    3. the system administrator documentation for the setup, the operation and the maintenance of the required infrastructure along with the FOSS third party services, and
    4. the user documentation for the general usage of the framework.
- The AHC framework software is divided into three services. These are the frontend user interface, the backend service and the AHC Runner service for conducting sandboxed experiments using real world Universal Software Radio Peripheral (USRP) devices.
- The backend service provides a REST API to expose the functionality of a deployed AHC framework instance (the instance). Moreover, the service also exposes and admin panel for the system administrators.
- The registered users of the instance use the system to schedule experiments. They can inspect already running or concluded experiments from the experimentation logs. The frontend service consumes the backend service's REST API to provide an user interface.
- The AHC Runner configures and orchestrates the USRP devices for the experimentation. It also reports the configured metrics throughout the experiment.

1.3 Stakeholders and Their Concerns

The stakeholders within the system can be divided into three groups:

1.3.1 Ordinary Users

Ordinary users who use the instance to conduct their experiments. Ordinary users use the GitHub integration to fetch their experimentation configuration and then they schedule experiments based on the selected configurations. In a graduate course scenario, ordinary users are the students who are registered to the course.

1.3.2 Super Users

Super users are the users with access to the admin panel. They can activate the accounts of newly registered initially-inactive users. They can also halt stuck experiments or cancel scheduled experiments. In a class, the professor would be the super user.

1.3.3 System Administrators

System admins set up and maintain the deployed AHC instance. The prototype instance is deployed as separate Docker container instances and depend on a running MinIO instance (a FOSS Amazon S3 compatible object storage). A system admin can build upon the prototype instance to tailor an AHC framework instance that suits their organization's needs. The software services having a permissive free and open source software license allows the system administrators to modify the software powering the prototype instance to this end.

2 References

This document was prepared with respected to the specifications described by the file below.

"ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering," in ISO/IEC/IEEE 29148:2018(E) , vol., no., pp.1-104, 30 Nov. 2018, doi: 10.1109/IEEESTD.2018.8559686.

3 Glossary

No	Term	Description
1	USRP	Universal Software Radio Peripheral (USRP)
2	REST API	**Re**presentational **S**tate **T**ransfer **A**pplication **P**rogramming **I**nterface
3	HTTP	Hypertext Transfer Protocol (HTTP) is an application-layer protocol for transmitting hypermedia documents, such as HTML.
4	Flask	Flask is a micro web framework written in Python.
5	API	API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other.
6	Python	Python is an interpreted high-level general-purpose programming language.
7	mDNS	In computer networking, the multicast DNS protocol resolves hostnames to IP addresses within small networks that do not include a local name server.
8	RPI	Raspberry Pi
9	OS	Operating System
10	Time-critical	Something that the performance is crucial for the success.
11	PostgreSQL	PostgreSQL, also known as Postgres, is a free and open-source relational database management system emphasizing extensibility and SQL compliance.
12	Qt	Qt is a widget toolkit for creating graphical user interfaces as well as cross-platform applications that run on various software and hardware platforms

Table 2: Connection use case

4 Architectural Views

4.1 Context View

This viewpoint provides a context of the system with all actors in general and detailed viewpoints. The Context Diagram provides a general explanation of the actors and their interactions with the system. In the following sections, Use Case Diagrams and detailed explanations for these diagrams can be found.

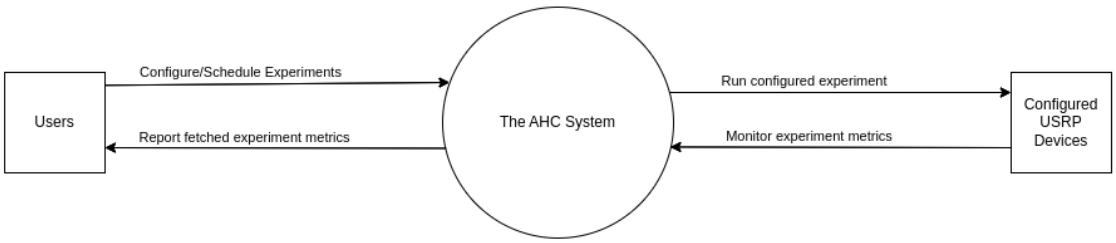

Figure 1: Context Diagram

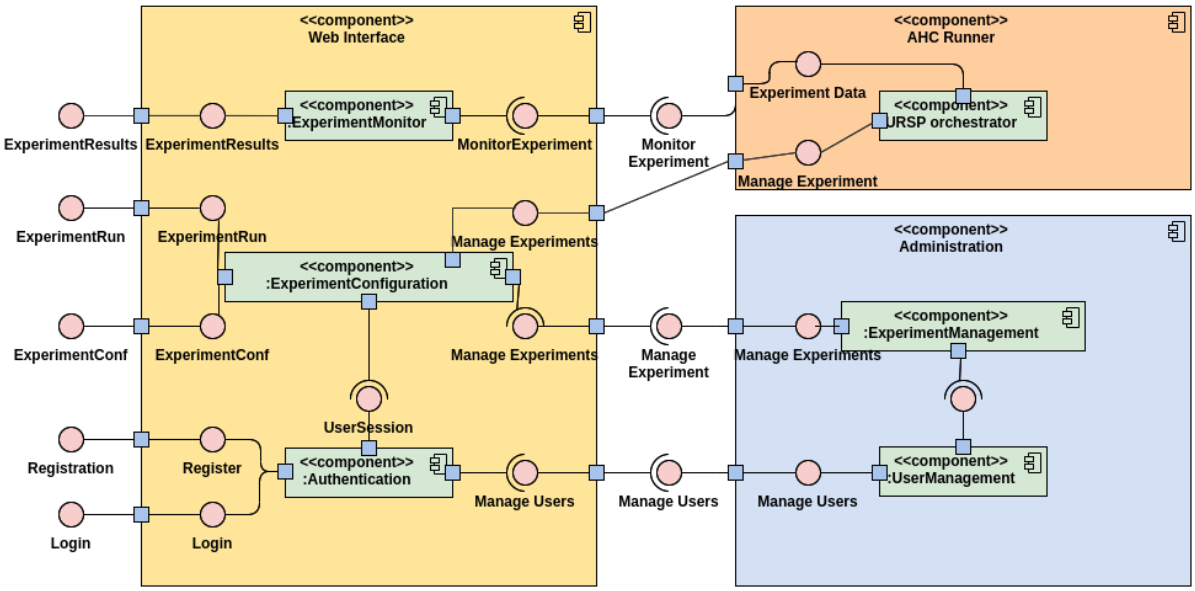

Figure 2: Component Diagram

4.2 Composition View

In this viewpoint, the components of the system will be identified. Details of the individual systems are available in the following sections.

4.2.1 Component Diagram

The component diagram can be seen in Figure 2.

Design Rationale

1. The user functionalities such as registration and logging in, experiment creation and scheduling and observing experiments are encapsulated in the Web Interface component.

2. ExperimentMonitor is responsible for monitoring experiments run by the AHC Runner.

3. ExperimentConfiguration allows experiment creation, configuration and scheduling as well as admin experiment operations such as stopping or postponing experiments by interfacing with the AHC Runner.

4. Authentication component provides ordinary user management functionality.

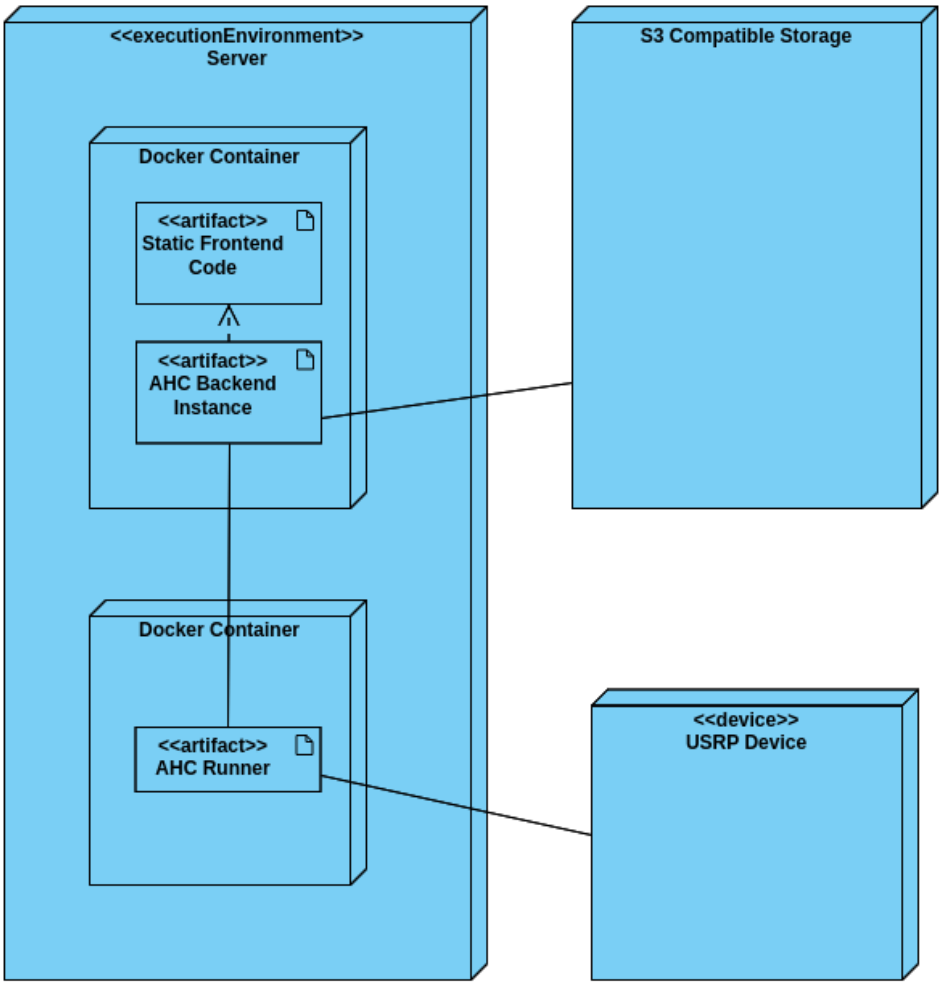

Figure 3: Deployment Diagram

5. Administration component allows the system administrators to manage users and ongoing/upcoming experiments.
6. AHC Runner orchestrates configured USRP devices according to the submitted experiment scenario and reports the experiment metrics.

4.2.2 Deployment Diagram

The deployment diagram can be seen in Figure 3.

Design Rationale

1. A docker container containing the AHC backend along with static frontend code is deployed to the production.

2. An S3 compatible object storage solution is required for storing experiment logs.
3. Another docker container containing the AHC Runner is also deployed to production server.
4. AHC Runner orchestrates the available USRP devices for the experiments.