Evaluation of Microservice Architecture Designs in an IoT-Context

Abramov Sviatoslav
svyatabram@gmail.com

Supervisor: Prof. Dr.-Ing Günter Schäfer

Supervisor: Associate Prof. Dr. Igor Anikin

TECHNISCHE UNIVERSITÄT
ILMENAU

navimatix
IT-Projekte mitgedacht

Content

- ►Introduction
- ►Requirements
- ►State of the Art
- ►The system overview
- ►Results
- ►Conclusion
- ►Future Work

navimatix
IT-Projekte mitgedacht

TECHNISCHE UNIVERSITÄT
ILMENAU

Introduction

The thesis overview

Introduction

The main goal is to find the best MSA architecture design of in an IoT context.

We compared:

- ► Interconnection methods
- ► Database management systems
- ► Load balancing strategies

What we have done

1. Analyzed most demanded technologies in MSA context.
2. Built the MSA application, satisfying most common requirements.
3. Developed a load generation, simulating IoT devices.
4. Implemented a monitoring system
5. Processed measurements.

Requirements

The system requirements

Functional requirements

1. Provide connectivity for IoT devices.
2. Provide basic authentication system.
3. Transform IoT device data model to the system data model.

Non-functional Requirements

Qualitative

- Testable
- Reproducible
- Deployable

Quantitative

- Response time
- Scalable

State of the art

Literature overview

Basic articles

M. S. Hatem Hamad and R. Abed, “Performance evaluation of restful web services for mobile devices,” *Computer Engineering Department, Islamic University of Gaza, Palestine, International Arab Journal of e-Technology*, 2010.

In the article advantages of RESTful web services before SOAP web services are shown:

- RESTful web services provide less message size.
- RESTful web services provide less response time.

Basic articles

P. J. Amaral M. and C. D., “Performance evaluation of microservices architectures using containers.,” *IEEE 14th International Symposium on Network Computing and Applications*, 2015.

In this article Server Virtualization provides performance improvement is shown.

- SV increases a server throughput
- SV decreases a server latency

Basic articles

- J. F. Kunhua Zhu and Y. Li, *Research the performance testing and performance improvement strategy in web application", 2nd international Conference on Education Technology and Computer.* 2010.

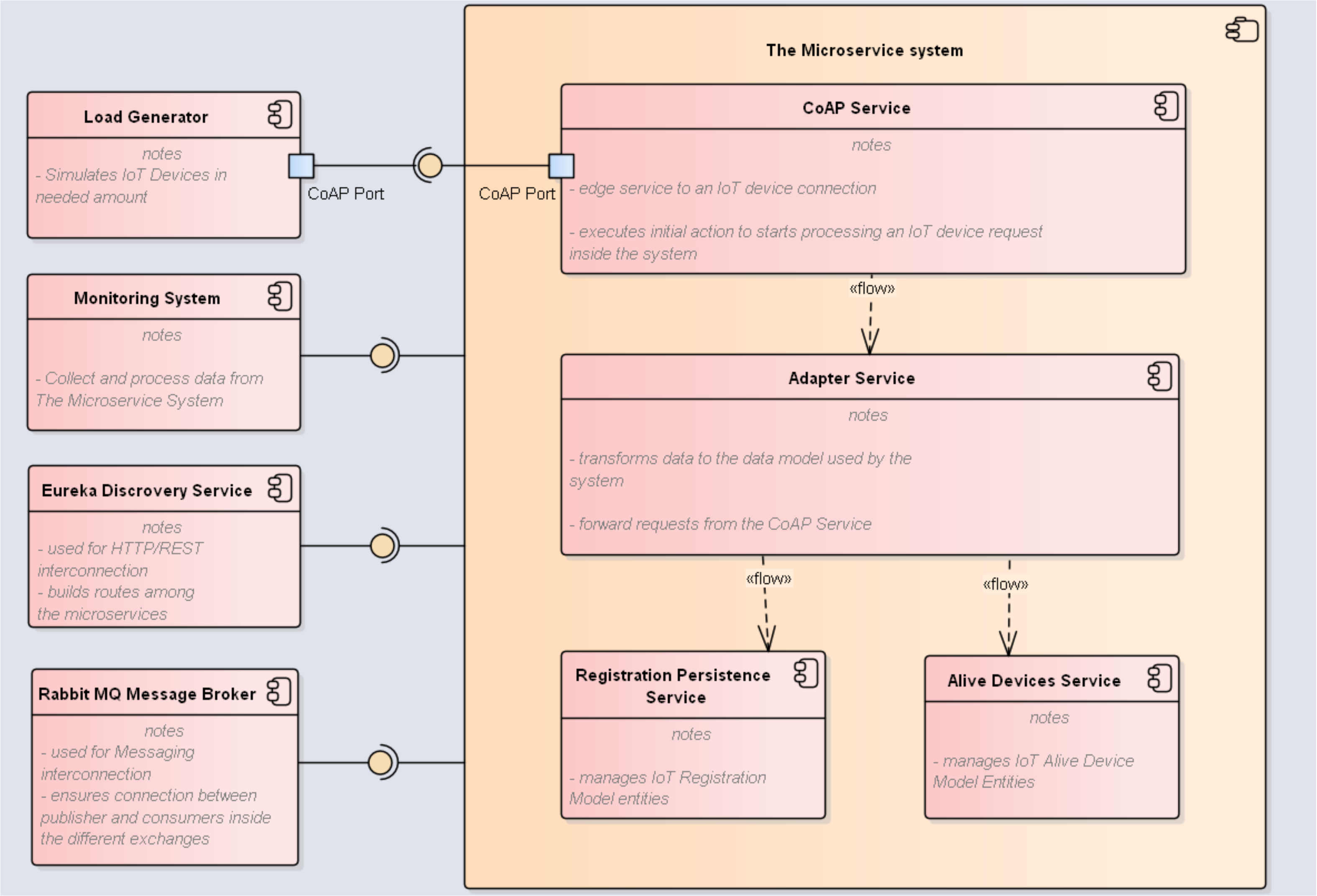
This article provides a survey about overall MSA based application testing :

- Functional and load testing of web MSA based web applications basis.
- Prediction of an application response time changing with an increasing user load.
- User load impact on application throughput.
- Possible bottlenecks caused by system utilization.

The system overview

Explaining the system components

Component diagram



The Microservice system

Load Generator

notes

- Simulates IoT Devices in needed amount

CoAP Port

CoAP Port

CoAP Service

notes

- edge service to an IoT device connection

- executes initial action to starts processing an IoT device request inside the system

«flow»

Monitoring System

notes

- Collect and process data from The Microservice System

Adapter Service

notes

- transforms data to the data model used by the system

- forward requests from the CoAP Service

Eureka Discrovery Service

notes

- used for HTTP/REST interconnection
- builds routes among the microservices

«flow»

«flow»

Registration Persistence Service

notes

- manages IoT Registration Model entities

Alive Devices Service

notes

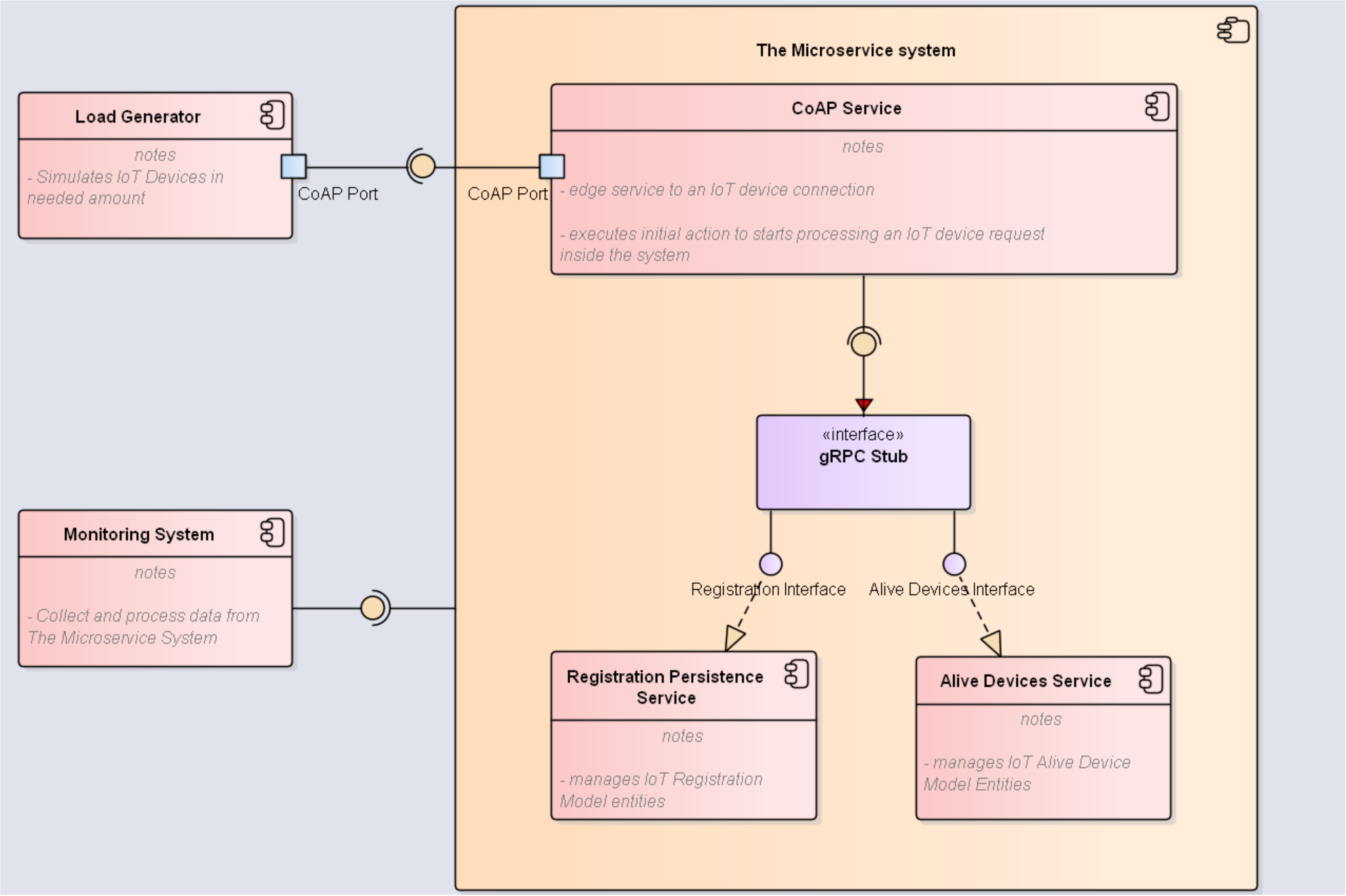
- manages IoT Alive Device Model Entities

Rabbit MQ Message Broker

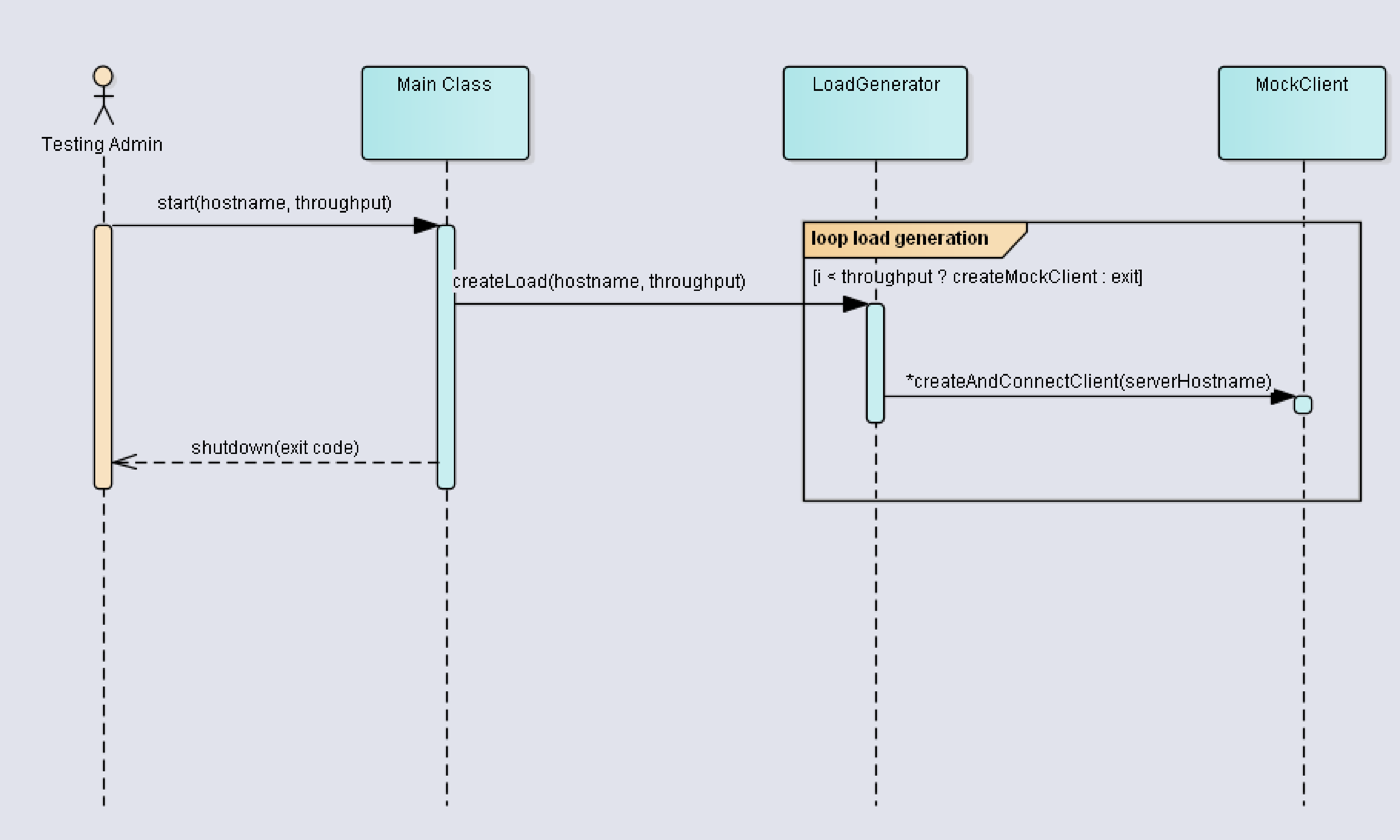
notes

- used for Messaging interconnection
- ensures connection between publisher and consumers inside the different exchanges

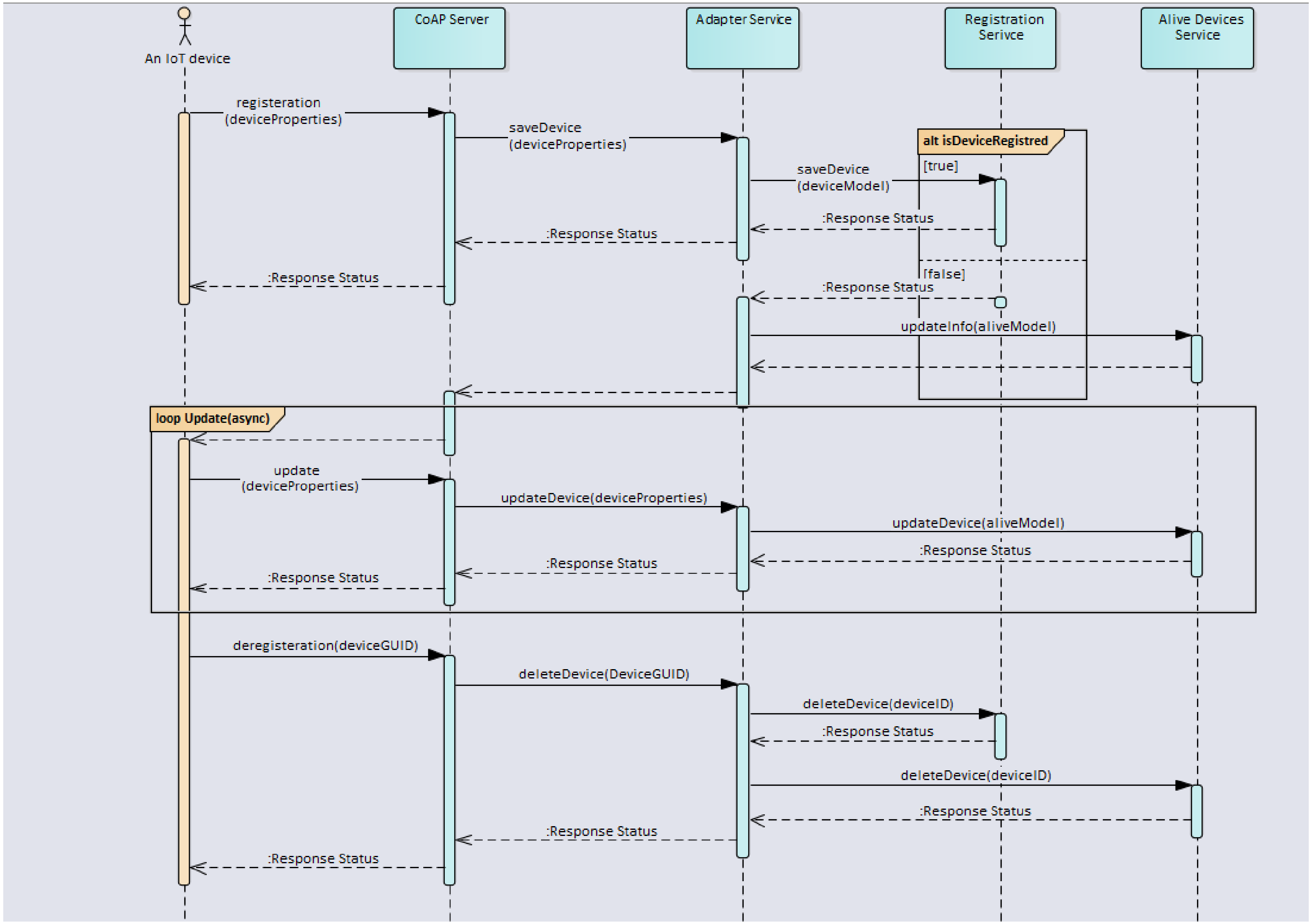
Component diagram (gRPC)


The Microservice system
Load Generator
notes
- Simulates IoT Devices in needed amount
CoAP Port
CoAP Port
CoAP Service
notes
- edge service to an IoT device connection
- executes initial action to starts processing an IoT device request inside the system
«interface»
gRPC Stub
Monitoring System
notes
- Collect and process data from The Microservice System
Registration Interface
Alive Devices Interface
Registration Persistence Service
notes
- manages IoT Registration Model entities
Alive Devices Service
notes
- manages IoT Alive Device Model Entities

Load generation sequence diagram


Testing Admin
Main Class
LoadGenerator
MockClient
start(hostname, throughput)
createLoad(hostname, throughput)
loop load generation
[i < throughput ? createMockClient : exit]
*createAndConnectClient(serverHostname)
shutdown(exit code)

Test scenario sequence diagram



Results

The Results discussion

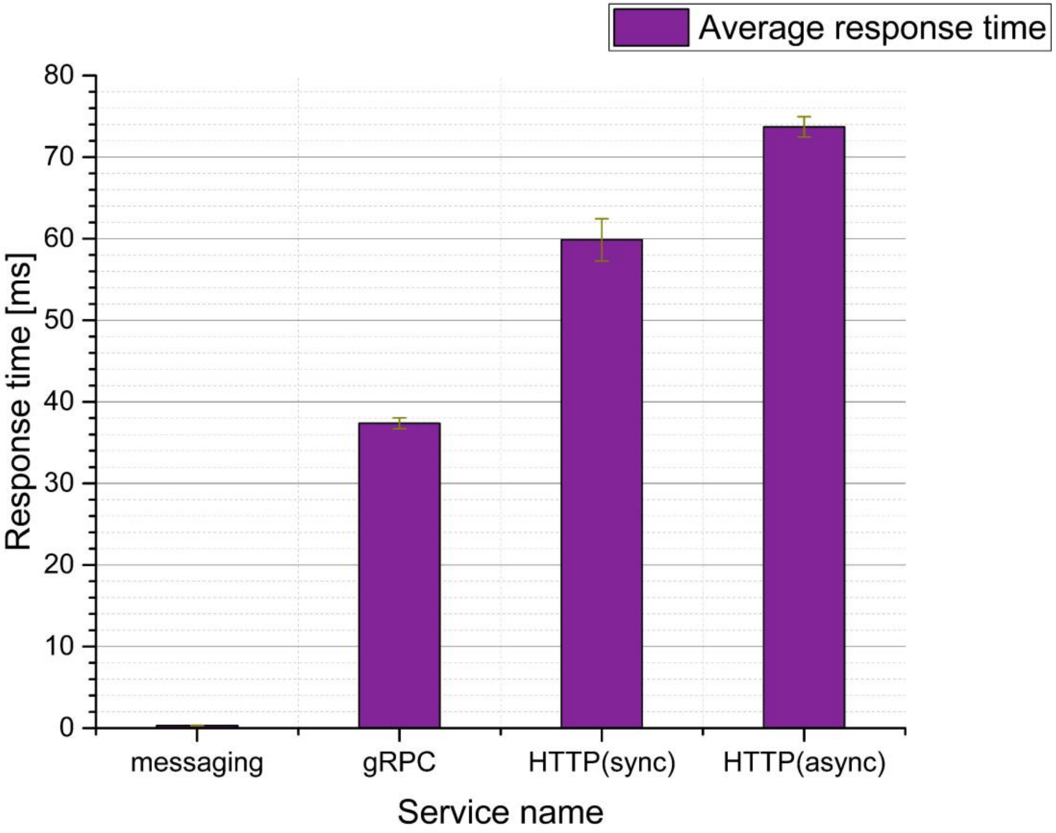
The measuring system

- To measure response we used *distributed tracing* system.
- We stored trace logs in *document-oriented* DB.
- We used search engine to retrieve and analyze logs

Interconnection comparison

- Messaging provides the lowest response time less than 1 millisecond.

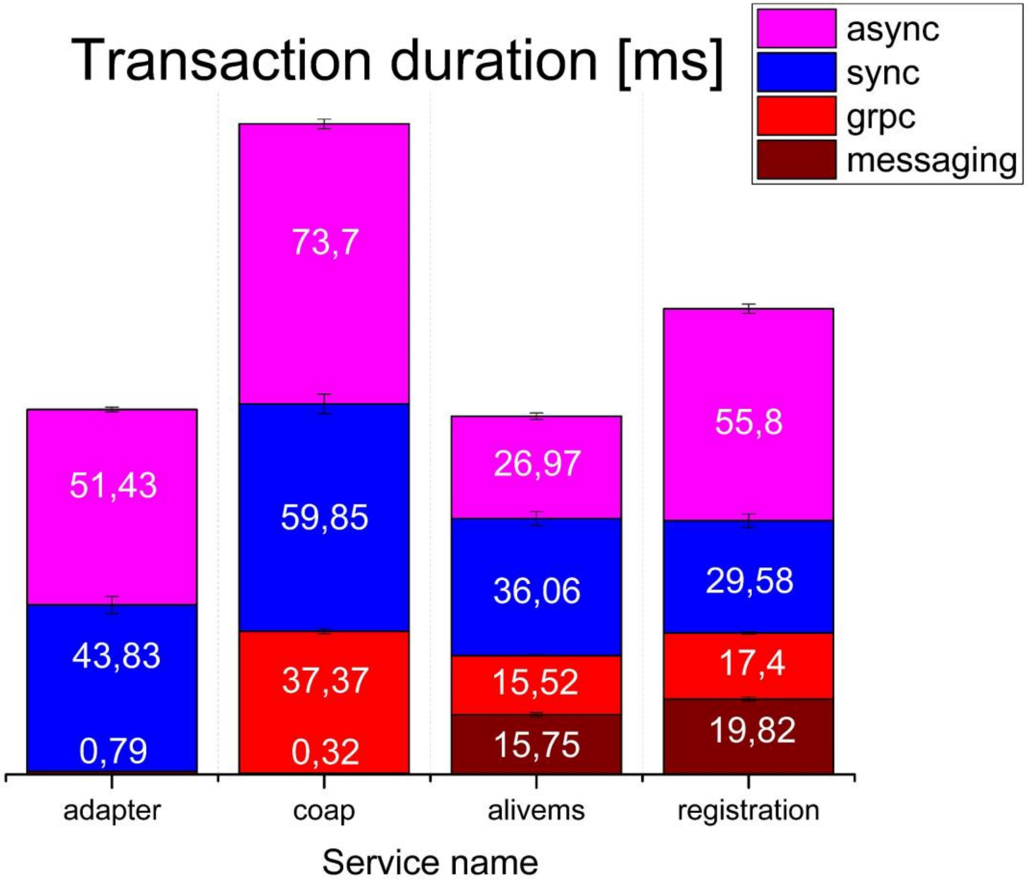
- Async. HTTP provides the highest response time, a bit higher 70 milliseconds.



Transaction duration by service

- gRPC interconnection provides lowest transaction duration time of persistence services.

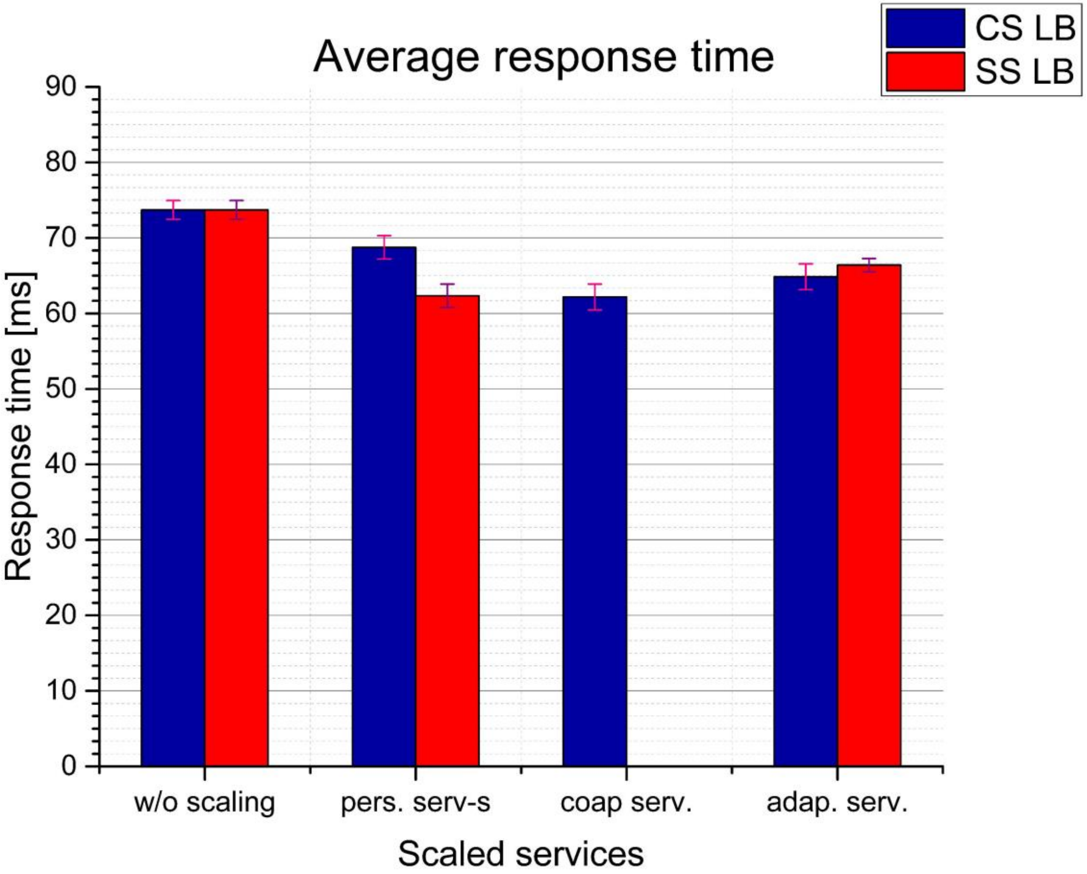
- The messaging one provides the lowest transaction duration time of non-persistence services.



Load balancing strategies comparison

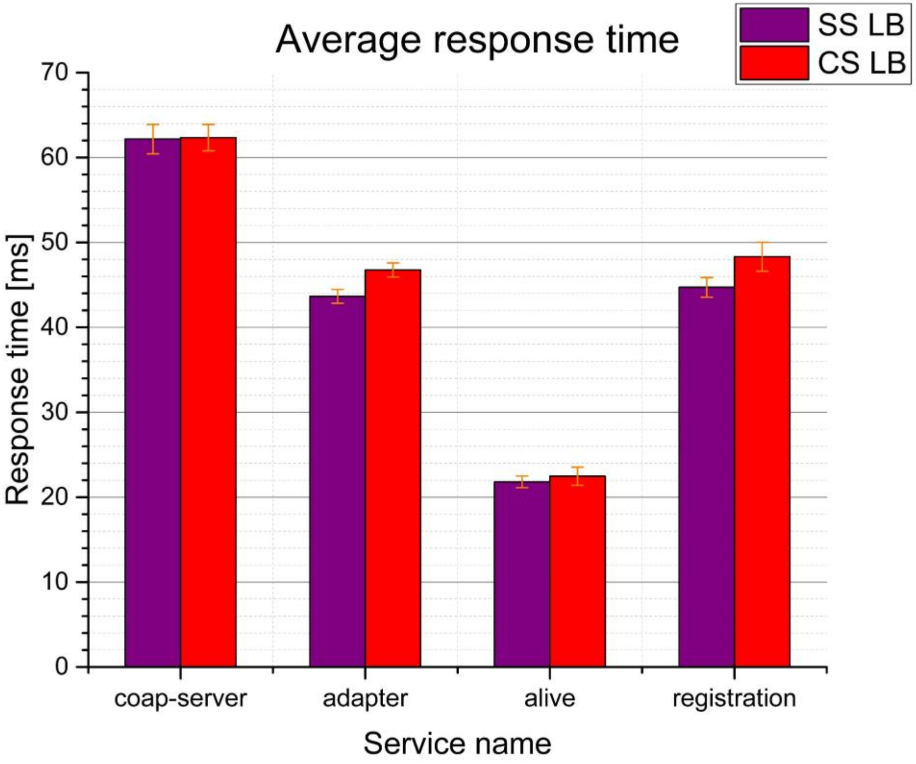
- Client-side LB strategy provides about 10% less the system response time, scaling persistency services.

- Server-side LB strategy improves the system response by almost the same amount but by scaling edge service.



Impact of LB on services

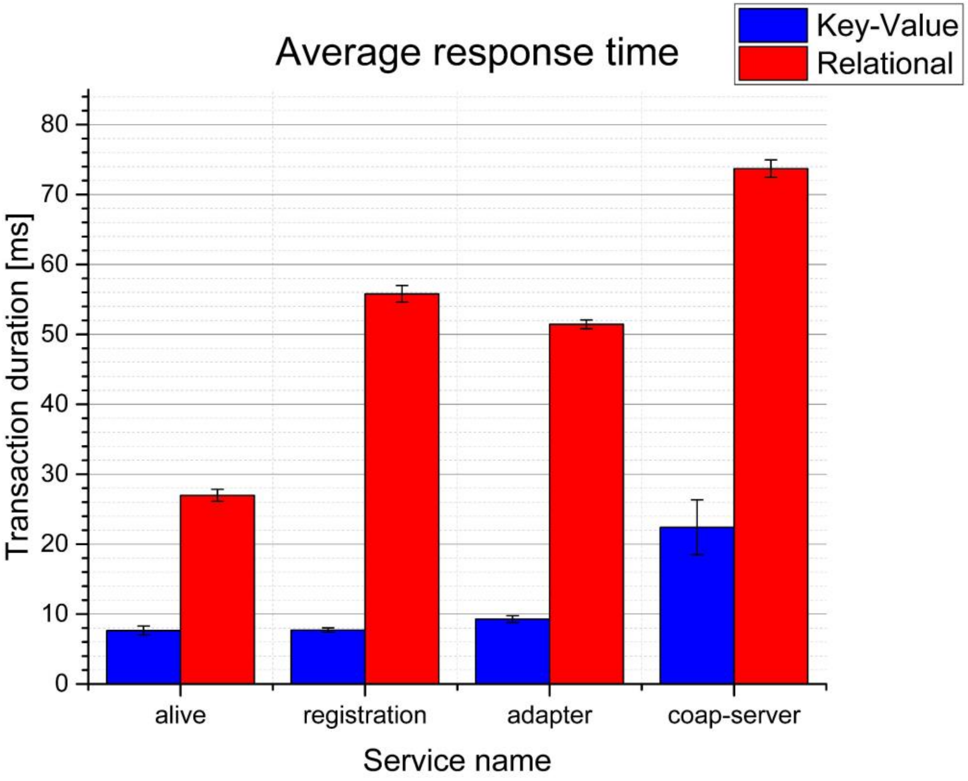
- Here need to add additional info on the plot



DBMS comparison

- Key-Value DBMS requires significantly less time to execute database operations.

- Despite a data structure it needs about 8 milliseconds to process an operation.



Conclusion

Summing up the thesis

Conclusion

- The fastest an IoT device request handling can provide a microservice system with messaging middleware interconnection to non-persistency services and gRPC interconnection to persistency services.
- Document-oriented key-value DBMS can ensure the lowest transaction duration of persistency services.
- Load balancing strategy choice depends on where a microservice system has a bottleneck.

Future work

- To test the microservice system with some additional services satisfying more realistic requirements.
- Improve the load generation.
- Survey about how caching might affect the system response time.
- Prove our research on a production made application.

Thank you for attention