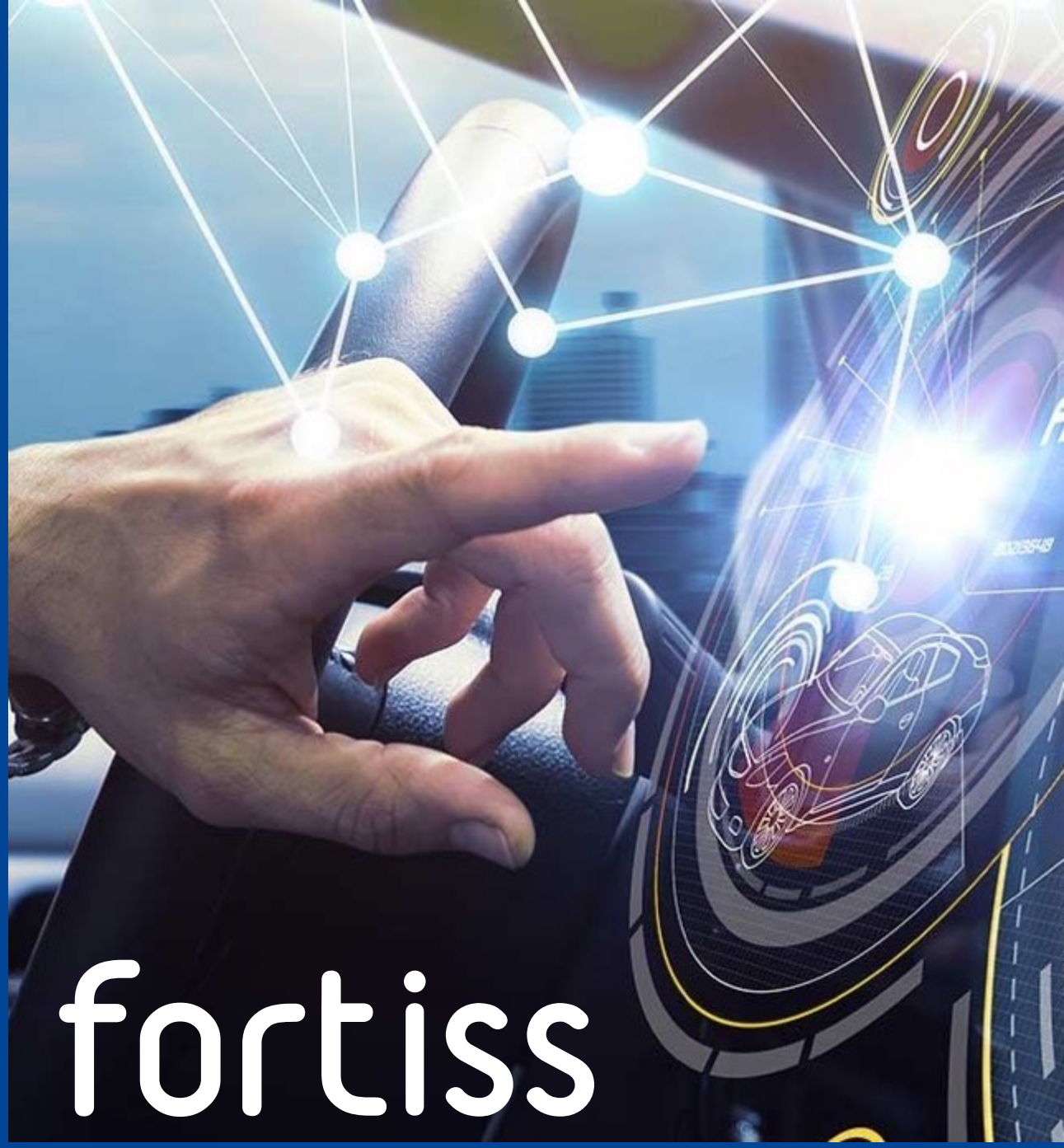
MArk8s

A Management Approach for Automotive Real-Time Kubernetes Containers in the Mobile Edge Cloud



Bernhard Blieninger

- Scientist (Automotive System Software and Architecture)
- fortiss GmbH
  Guerickestr. 25
  80805 Munich
  GERMANY
- www.fortiss.org
  blieninger@fortiss.org

Aaron Dietz

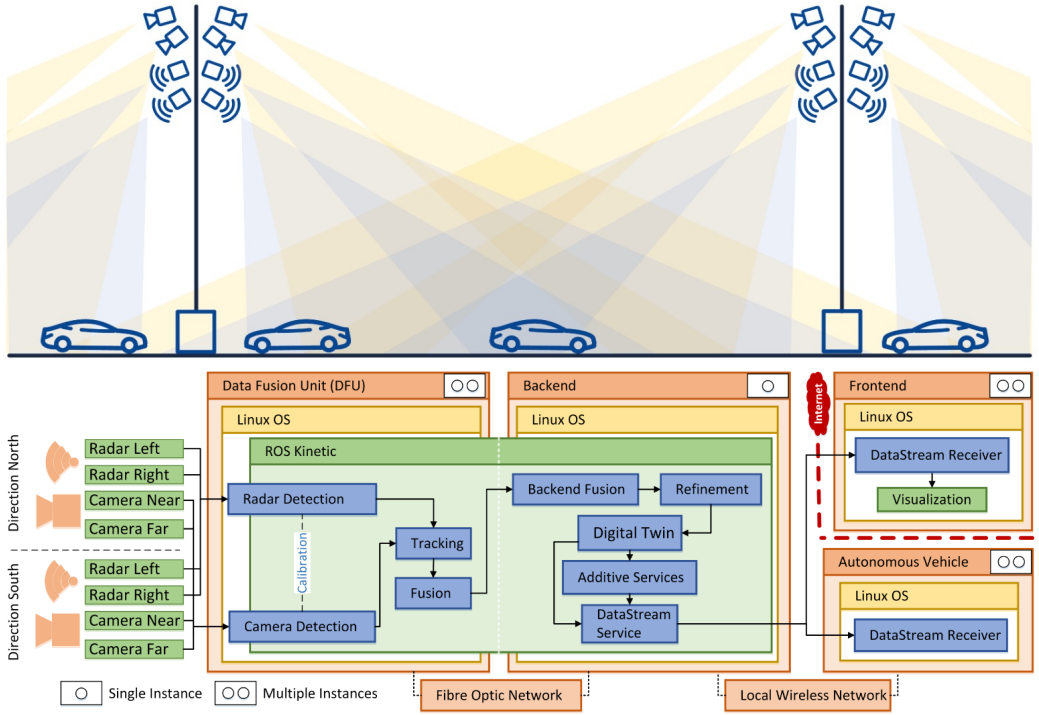
- Former student at Technial University of Munich
- Now at Rubean AG
  Kistlerhofstr. 168
  81379 Munich
  GERMANY
- www.rubean.com
  aaron.dietz@rubean.com

Introduction

- Providentia++ testbed:
  - MEC is enhanced with sensor systems
  - Sensor data is fused at the RSU
  - Digital Twin of the traffic scenario is created
  - Additional and extended information is provided to the vehicles

    => MEC can be a full substitute of vehicle driving tasks?

    1. MEC is not always available.
    2. MEC may not be reliable.



From: https://doi.org/10.48550/arXiv.1906.06789

Mark8s - General Idea

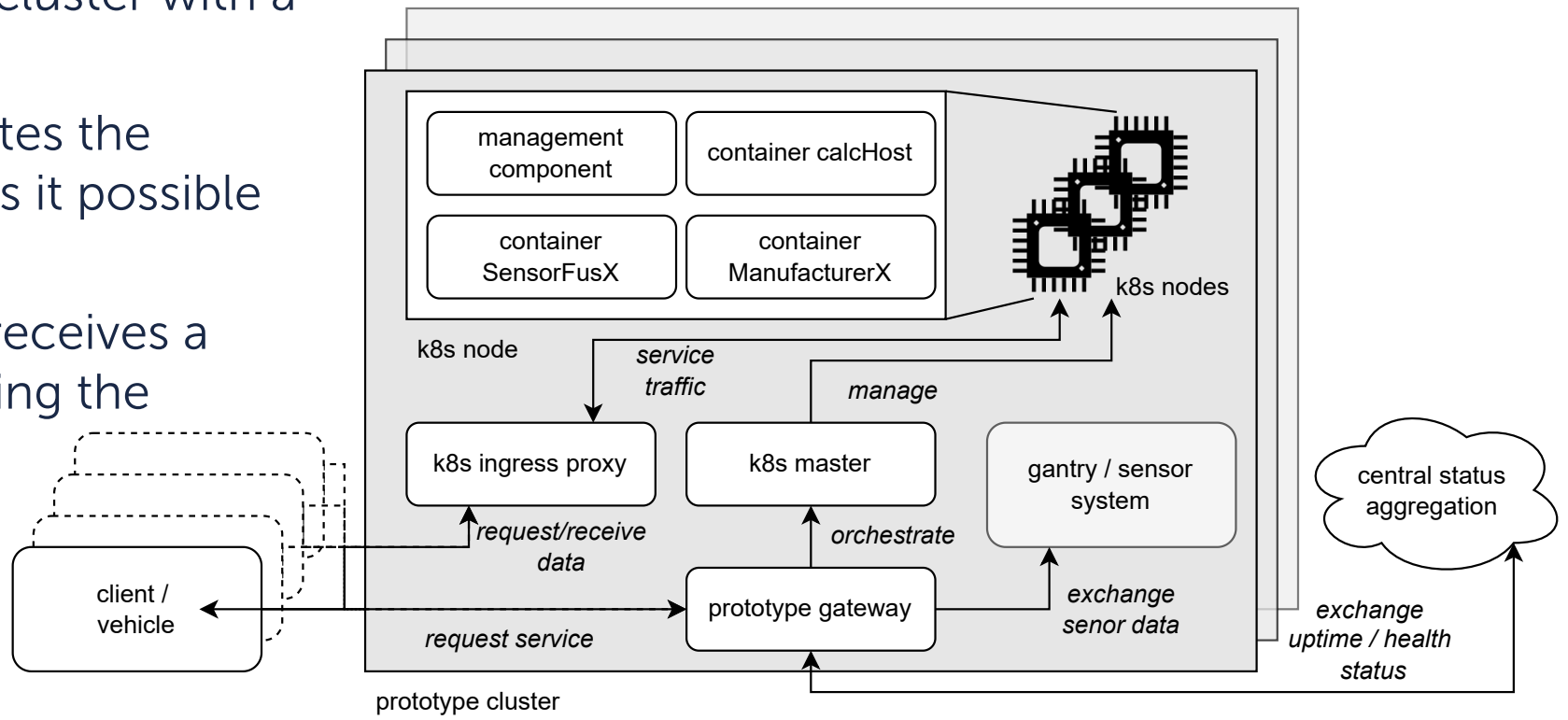
- Further exploit similarities of MEC and vehicle
  - Modularize applications
  - Apply containerization
  - Enable Real-Time capability
  - And reusability of applications

- Possibility use MEC as testbed for vehicle software

- One-Shot Containers
  - e.g. lane assistant
- Multi-Shot Containers
  - e.g. driving convoy
- Permanent Containers
  - e.g. extended sensing

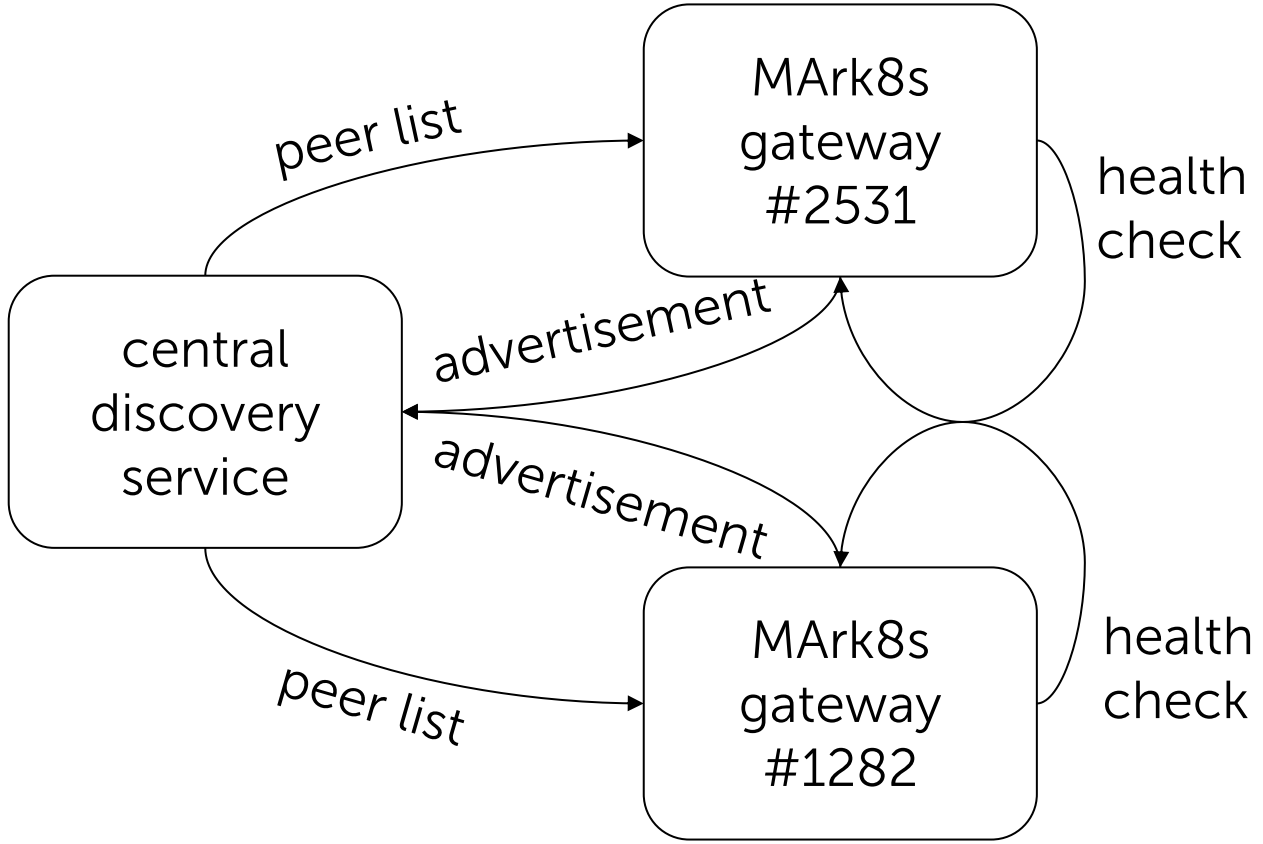
Mark8s - Setup

- Core is a default Kubernetes cluster with a RT-Kernel-patched Linux
- Prototype gateway orchestrates the Kubernetes cluster and makes it possible to define services
- Client requests services and receives a generated ingress URL for using the deployed services



Mark8s - Decentralized Units

- Only for initial advertisement a central discovery is needed
- MArk8s gateways automatically perform multiple health checks against their peers
- MArk8s gateways can redirect clients to healthy neighbours if they themself can not schedule a workload

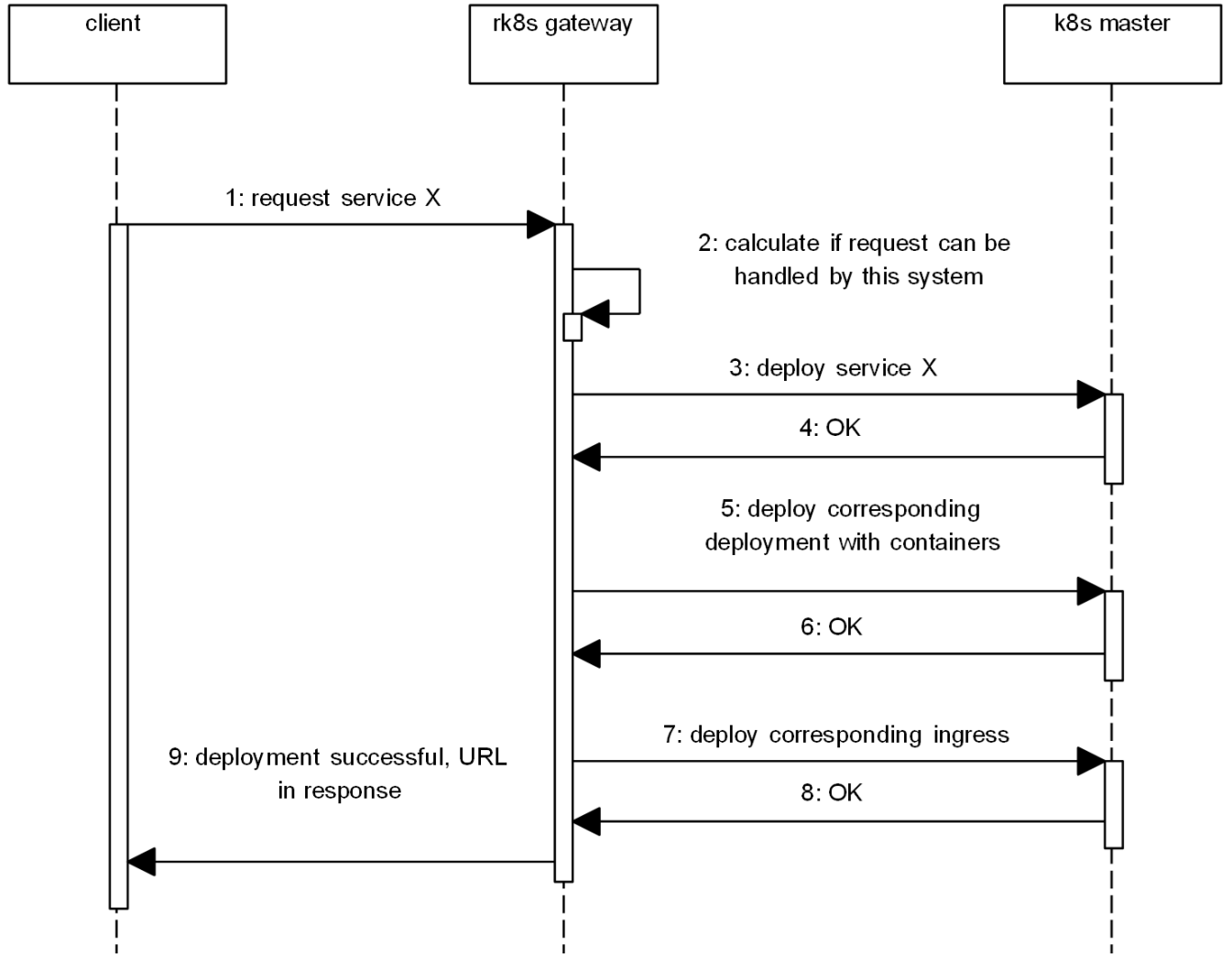


Mark8s - Client Request

- Simplified request for a service

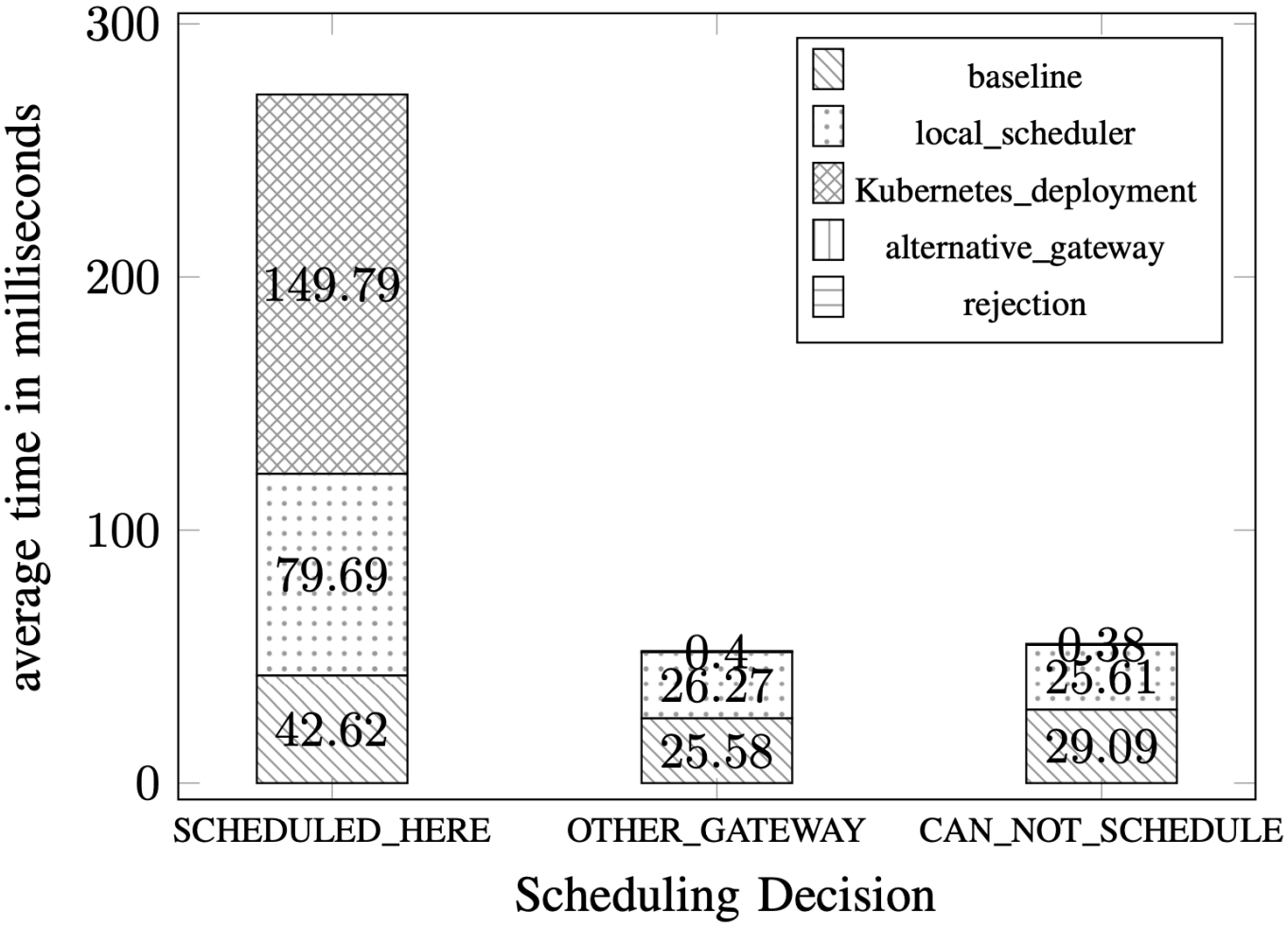
- Receive request for service
- Check if it is schedulable
- Deploy subcomponents
- Respond to the client

- Service will startup asynchronously (no waiting for readiness)

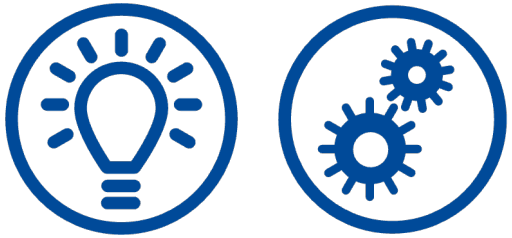


Mark8s - Tests and Evaluation

- Testing the request-response time for three scenarios:
  - Local gateway health: SCHEDULE_HERE
  - Local gateway unhealthy, but neighbour healthy: OTHER_GATEWAY
  - Local gateway unhealthy and neighbours unhealthy: CAN_NOT_SCHEDULE



Conclusions and Future Work



- Decentralization of MEC units with lightweight request-response framework Mark8s is feasable
- The first implementation of Mark8s introduces only a small overhead and can be easily tested, evaluated and extended (with e.g. RT-kubernetes, REACT)

- Future Work
  - Extension of Mark8s to enable dependent services and predictive requests
  - Further investigation of real-time workloads and exploitation of system similarity

Many thanks.

Any questions?

Tracing Tools

+