Mobile Cloud Computing

EINSTEIN CENTER Digital Future

Technische Universität Berlin

Fog Computing

Bermbach | Part 5: Testing and Benchmarking

Agenda

Technische Universität Berlin

Lectures

- **From Cloud to Fog Computing**
- **Data Distribution**
- **Data Management**
- **Platforms & Applications**
- **Testing & Benchmarking**
- **LEO Edge Computing**

Assignments

- Prototyping Assignment
- Reading Assignment

Wrap-up

- **Q&A**
- Final Test

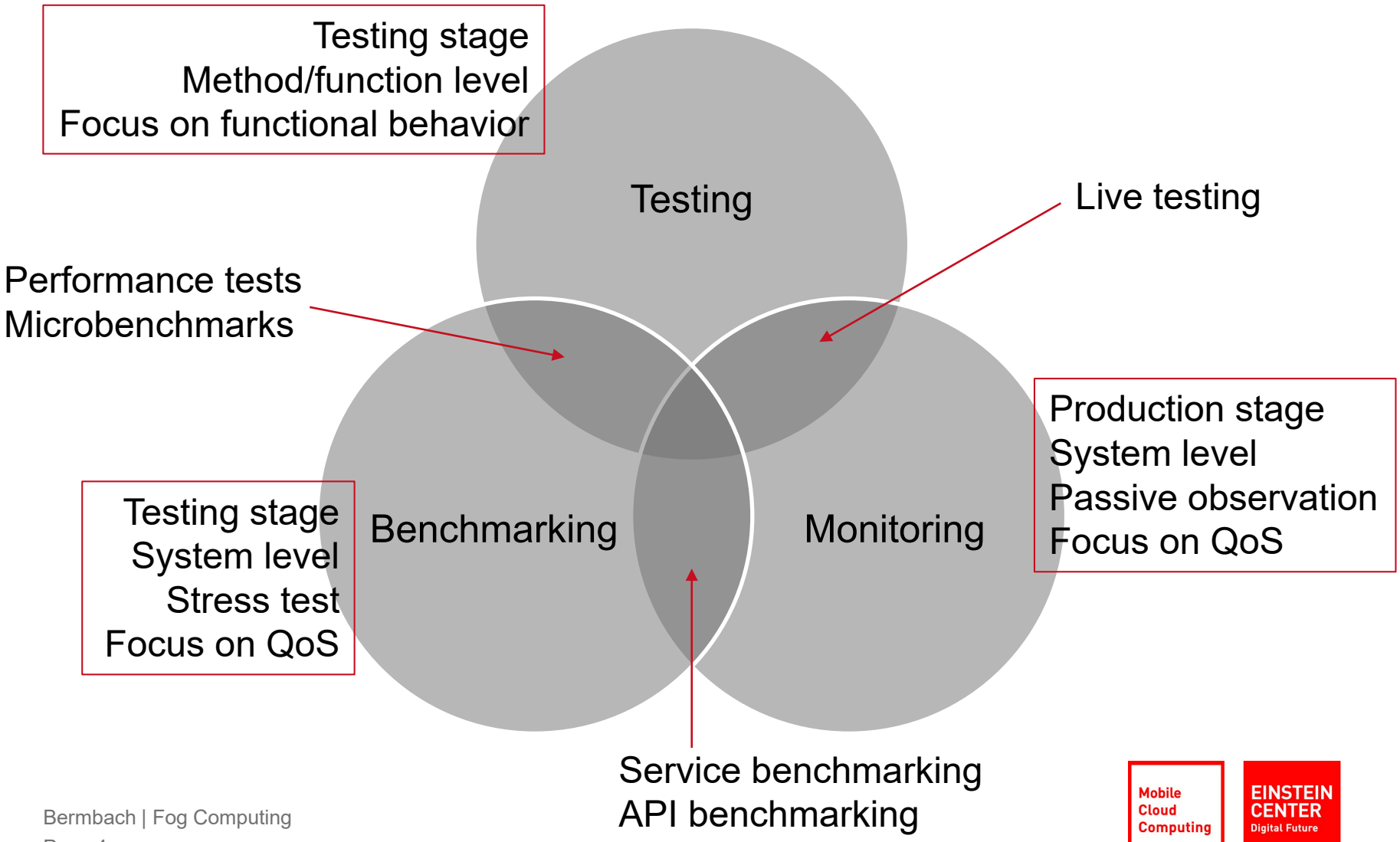
Mobile Cloud Computing

EINSTEIN CENTER Digital Future

Testing and Benchmarking

How do we can we test and benchmark fog-based systems, what are unique challenges in fog computing?

Definitions



Technische Universität Berlin

TESTING

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

Testing

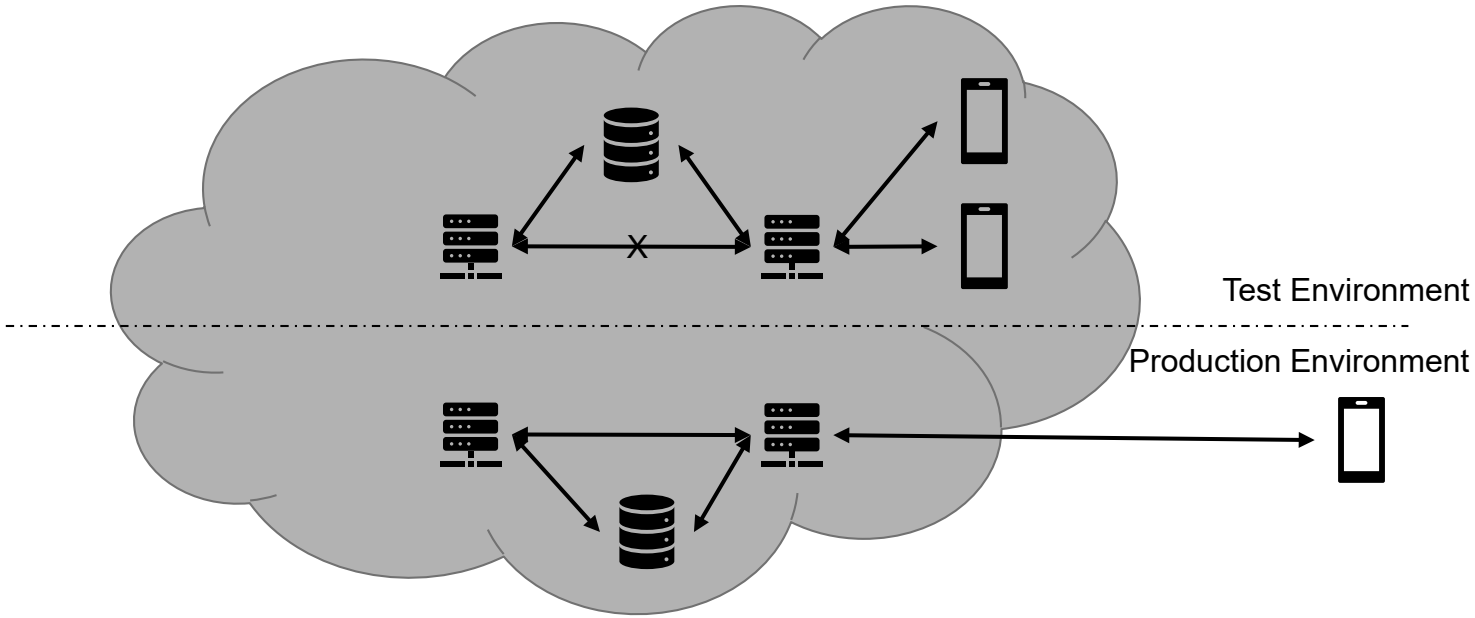
Important part of software engineering process: Make sure that the system works as intended

Phases:
(Unit Testing)
Integration & Live Testing

- Canary Testing
- Dark Launches
- A/B Testing
- …

Cloud Integration Tests

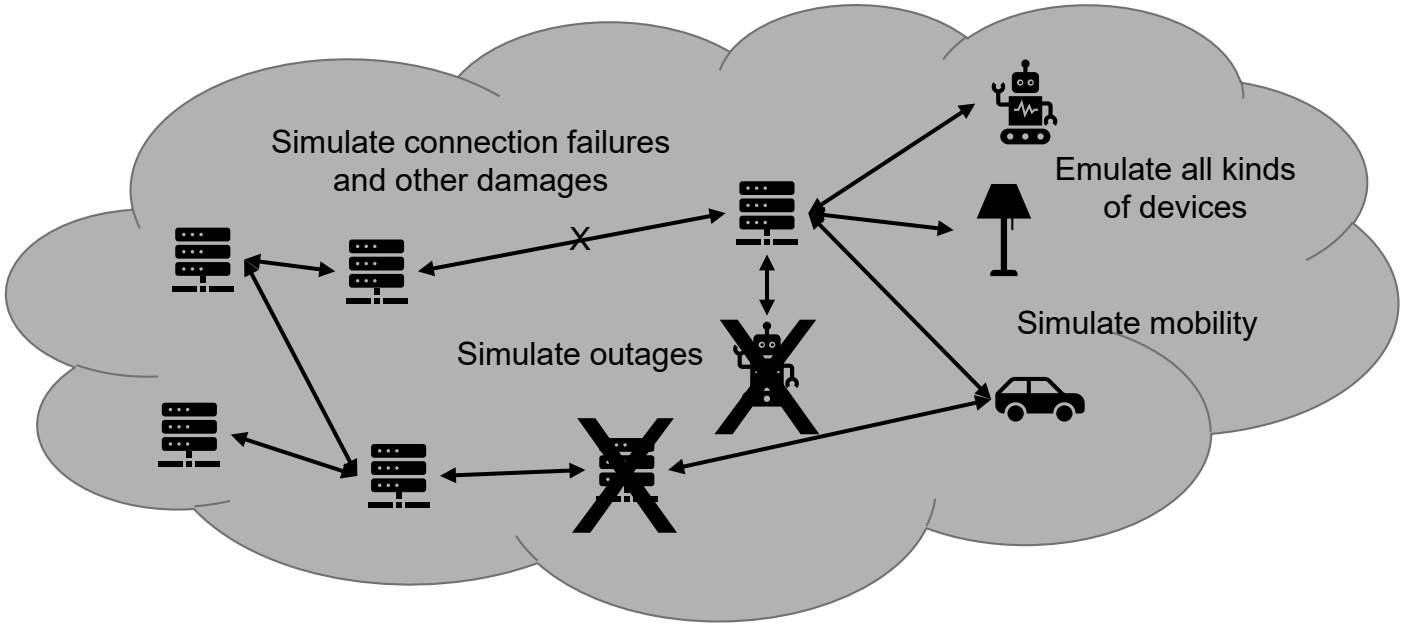
Setup (virtual) testbed and check whatever is required
Mock services, data, devices
Evaluate corner-cases which usually should not exist in production



Fog Integration Tests

Testbed creation is difficult because physical infrastructure and devices cannot be duplicated

(Partial) Solution: virtualize & emulate fog environment in the cloud



Live Testing

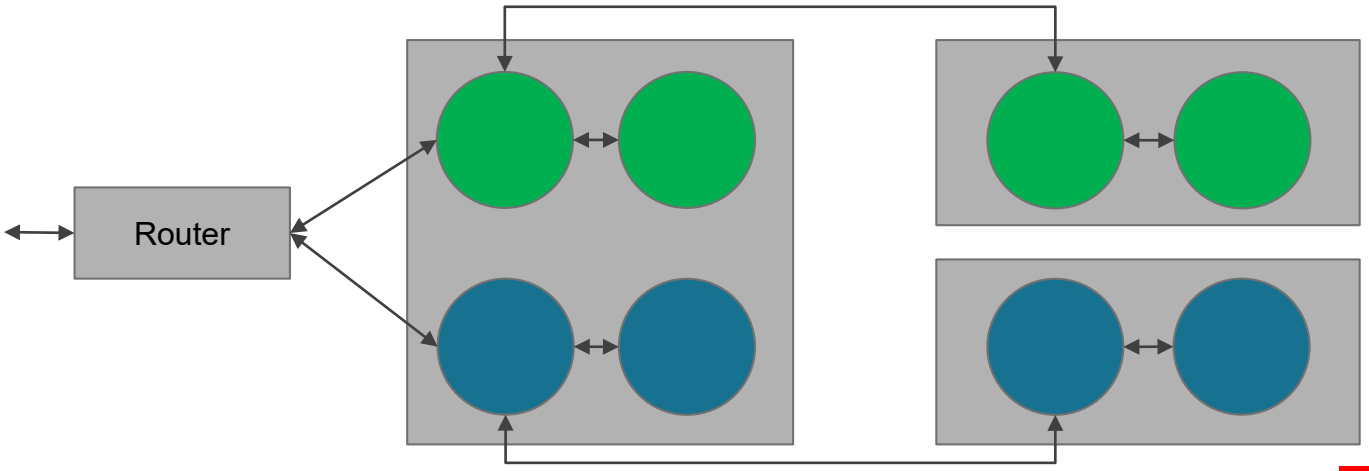
If possible, live testing techniques test new software versions in production

Basic principle: Monitor what happens…
…while rolling out an update gradually
…while directing part of the traffic to old and/or new version

Example: Blue-Green Deployments

There are two identical versions of the production environment, the blue and the green one

1. Deploy new version to blue environment
2. Smoke tests, etc., against the blue system
3. Switch the router configuration to the blue system
4. The blue environment becomes the production system
   (or switch back if there are errors)



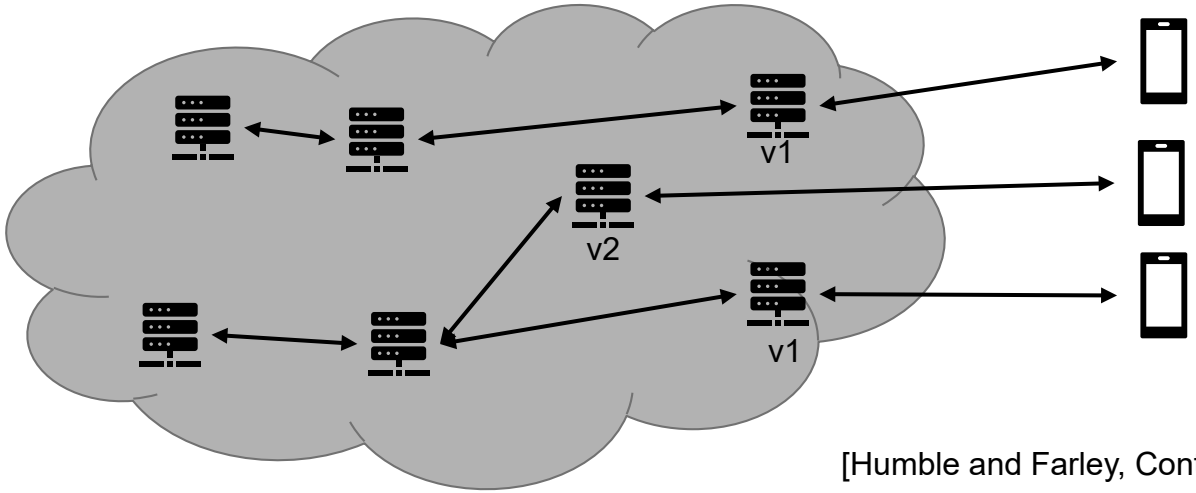
Example: Canary Releasing

Rollout of a new version only to a subset of production servers

Any problem with this new version will not affect the majority of users

Benefits:

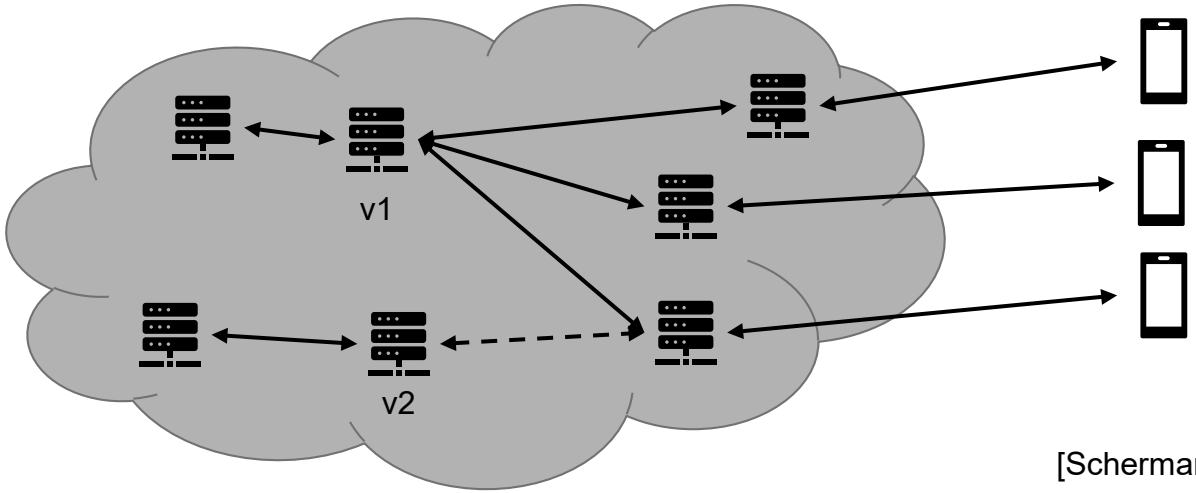
- Easy to revert: stop routing users to canary servers
- Use it for A/B-Testing: Decide on user-defined criteria whether to keep the old or new version (e.g., revenue, quality of results, …)
- Check capacity requirements by incrementally increasing the load



[Humble and Farley, Continuous Delivery, 2011]

Example: Dark/Shadow Launches

- Functionality is deployed in a production environment without being visible or activated for the users
- Some or all production traffic is duplicated and routed to the shadow versions as well
- Observing the shadow version without impacting any user



[Schermann et al., Bifrost, 2016]

Fog Live Testing

Technische Universität Berlin

Can continue to be used in the cloud part or on intermediary nodes (>= single rack)

Difficult on edge devices which

- may not have the capacity to run two versions in parallel
- may have safety requirements which make canary releases impossible

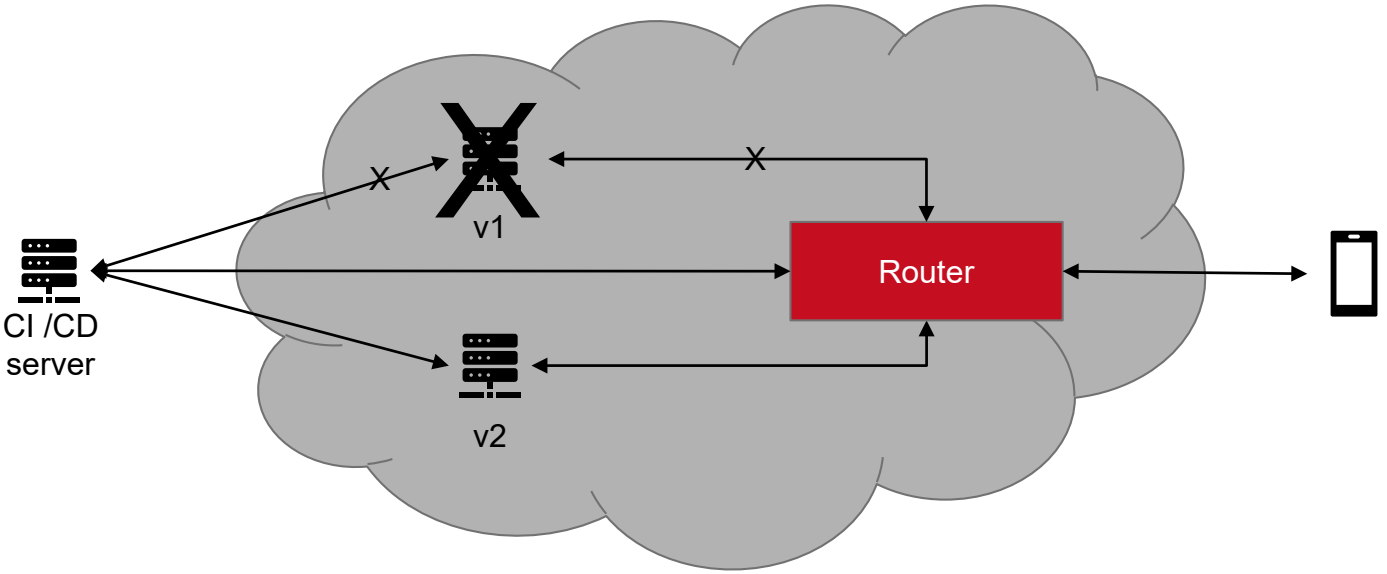
⇒ Find a separate solution for the edge part

- Mock edge devices in the cloud (see MockFog case study)
- Have a dedicated (physical!) testbed
- …

⇒ How about deployment?

Deploying Cloud Applications

- Changes are pushed to devices/instances through IaC
- New virtual devices are created, configured, and deployed with the new version; old instances are disconnected terminated



Deploying Fog Applications

For fog applications this only works for the cloud end (and possibly for smaller cloud-like data centers).

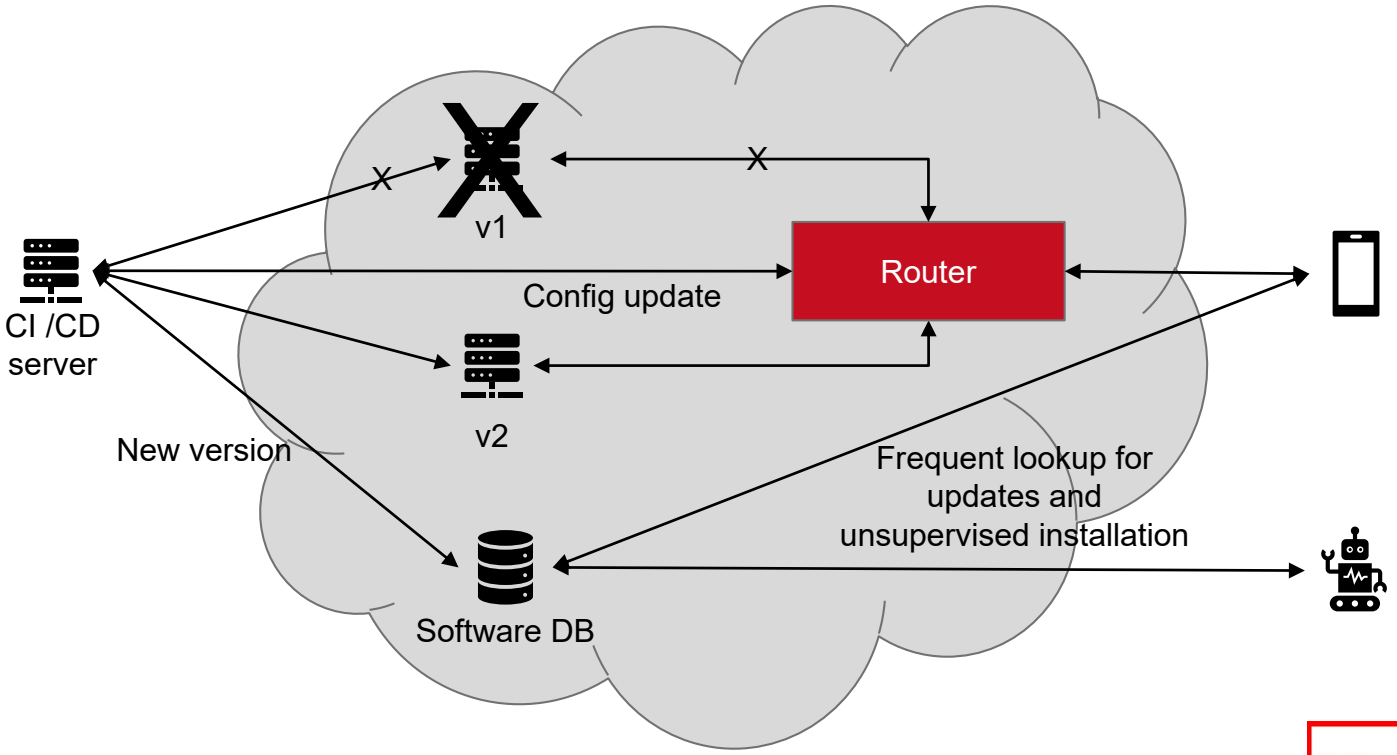
Edge devices often need to be physically connected *at least once* for deploying the first version.

One strategy (there may be others): Use an app store-like approach

- Update is sent to a central software repository
- Deployed application frequently checks for updates and self-updates if necessary => pull approach

Deploying Fog Applications

- Plan with incompatibilities and different versions on devices
- Use versioned interfaces!



BENCHMARKING

What is benchmarking?

Benchmarking is a way to systematically study the quality of cloud services based on experiments.

For this purpose, a benchmark tool creates an artificial load on the System under Test (SUT) while carefully tracking detailed quality metrics.

A key design goal of benchmarking is to mimic an application as close as possible to get meaningful results, however, benchmark runs also aim to extensively stress the SUT, e.g., through system load or even injected failures.

Adapted from:
Bermbach, Wittern, Tai. Cloud Service Benchmarking: Measuring Quality of Cloud Services from a Client Perspective. p.12. Springer 2017.

Design objectives

Relevance: benchmark the important parts, mimic real-world use

Repeatability: maximize determinism in the benchmark

Fairness: treat all SUTs the same

Portability: avoid assumptions about the SUT, make the benchmark broadly applicable

Understandability: have an intuitive benchmark specification

Implementation objectives

Correctness: assert adherence of implementation to specification

Distribution: build the benchmarking tool for distributed deployments, keep coordination pre-benchmark run, consider clock synchronization

Fine-grained logging: never discard information if not absolutely necessary

Reproducibility: use repeatable benchmarks, repeat often, run sufficiently long, document settings

Portability: use adapter designs, consider extensibility and evolvement, avoid assumptions on the SUT

Ease of use: document everything, provide instructions, release code

Fog-specific challenges

Technische Universität Berlin

Geo-distribution of experiments

Deployment of benchmarking clients for edge-based SUTs

Distributed measurements of QoS (e.g., end-to-end latency in an IoT data processing pipeline)

Multi-workload scenarios (e.g., event-driven at the edge, OLAP and OLTP in the cloud)

Complex analysis and results

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

Technische Universität Berlin

Benchmarking cloud VMs and edge devices

DeFog*

McChesney et al. DeFog: Fog Computing Benchmarks. ACM SEC 2019.

* Slides adapted from Blesson Varghese‘s original conference presentation and used with his kind permission.

Mobile Cloud Computing

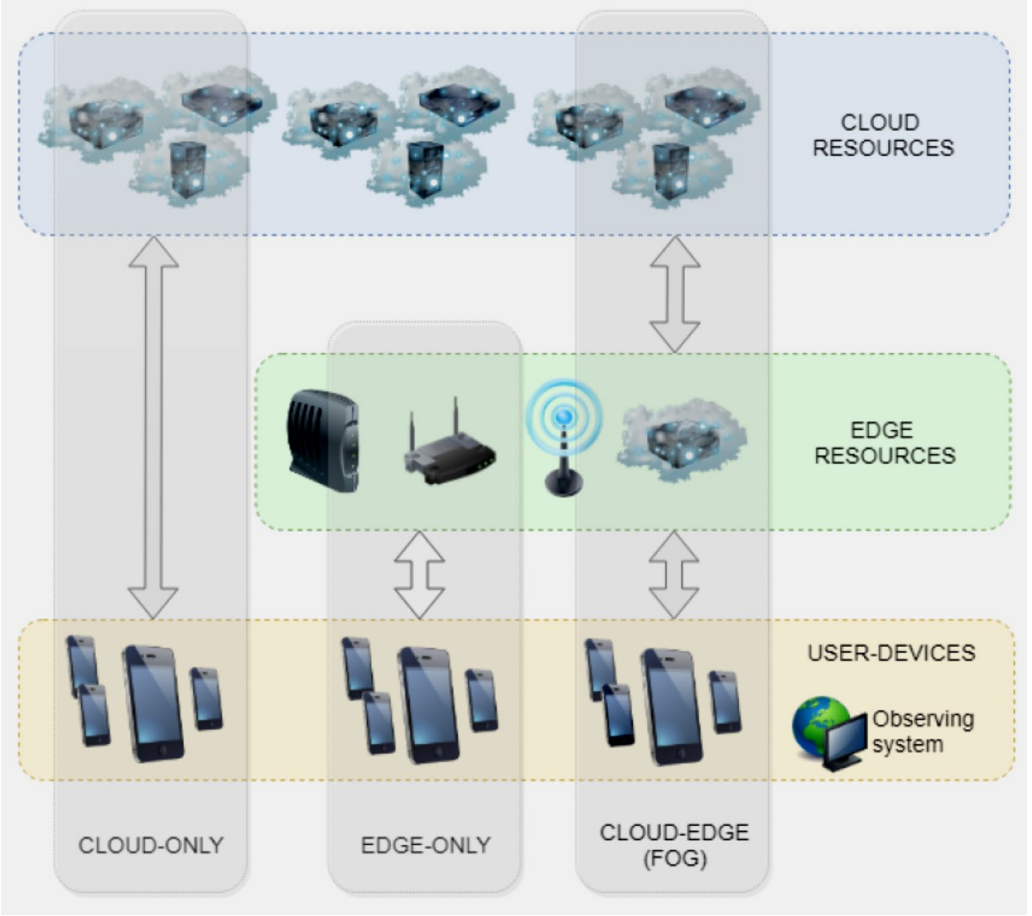
EINSTEIN CENTER Digital Future

Motivation

Applications can be deployed in different ways

Various hardware options exist on the edge.

How can we compare them?



Research questions

Q1: How can the relative performance of using a Fog platform over the cloud-only platform be understood and quantified?

Q2: If there are multiple services of an application that can be moved to the edge, then how can performance of using the Fog be understood?

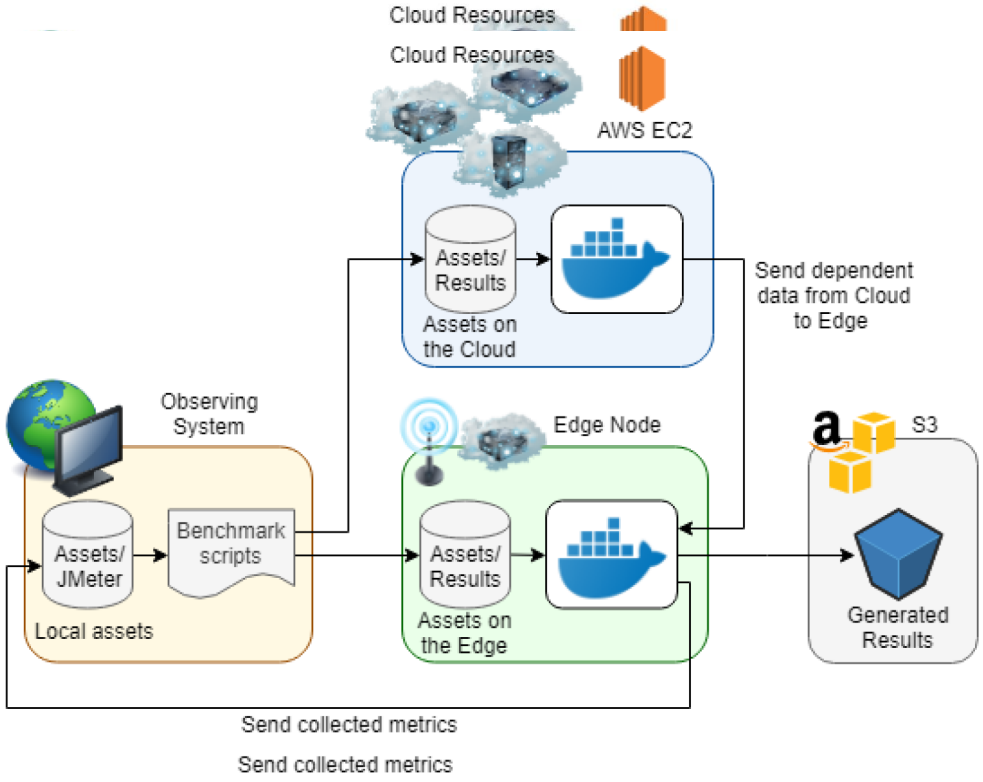
Q3: If there are multiple competing resources available at the edge suited for a specific service, then which resource should be selected for maximising the overall performance gain?

Deployment options

Three deployment modes:

- Cloud-only
- Edge-only
- Cloud-Edge (Fog)

Docker as deployment vehicle



Approach

Use a set of representative benchmark applications

Measure end-to-end performance as well as low-level metrics

6 applications: latency-critical, bandwidth-intensive, location-aware, or compute-intensive

Application	What it does
YOLO	Object classification using deep learning
PocketSphinx	Speech-to-text conversion
Aeneas	Text-audio forced alignment
iPokeMon	Geo-location based online mobile game
FogLAMP	IoT Edge gateway application
RealFD	Real-time face detection on video detection

Technische Universität Berlin

Benchmarking fog-based FaaS platforms

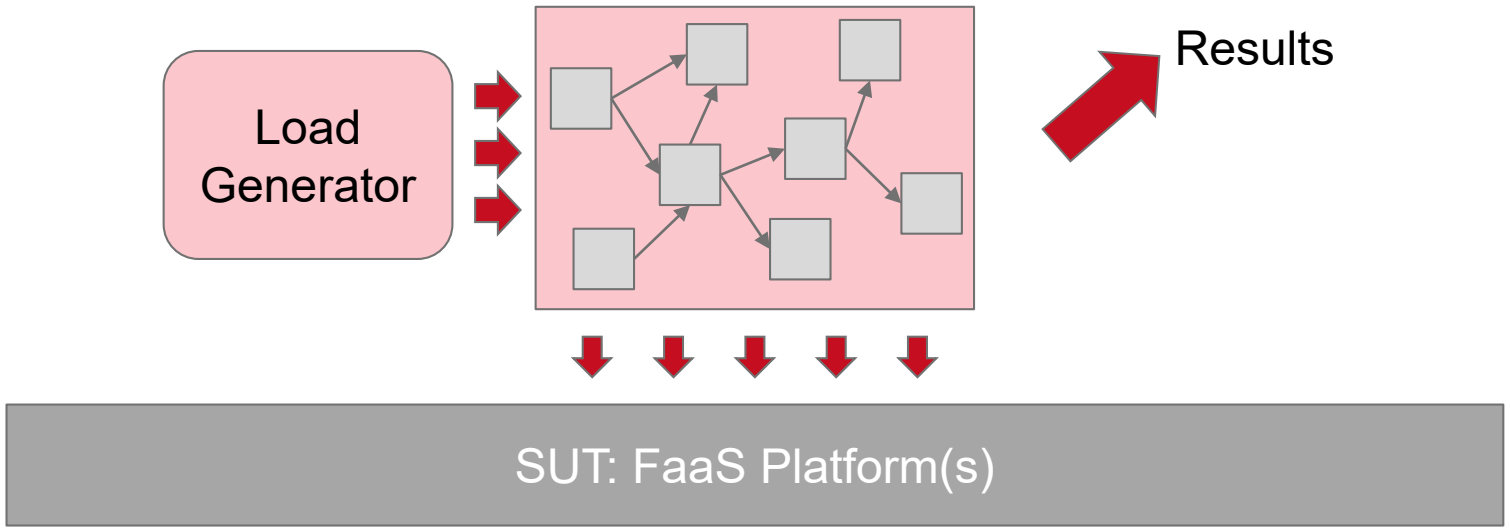
BeFaaS

Grambow et al. BeFaaS: An Application-Centric Benchmarking Framework for FaaS Environments. IEEE IC2E 2021.

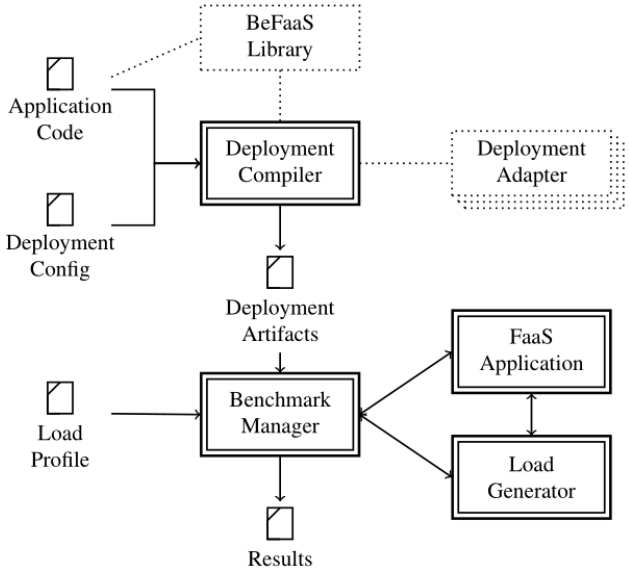
Mobile Cloud Computing

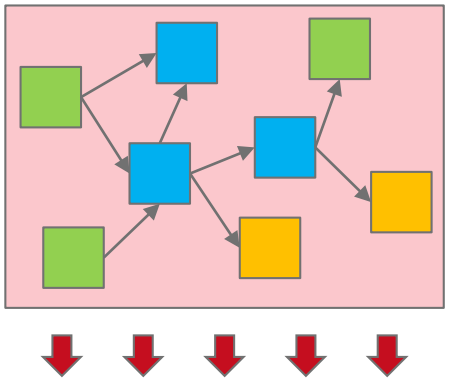
EINSTEIN CENTER Digital Future

BeFaaS architecture



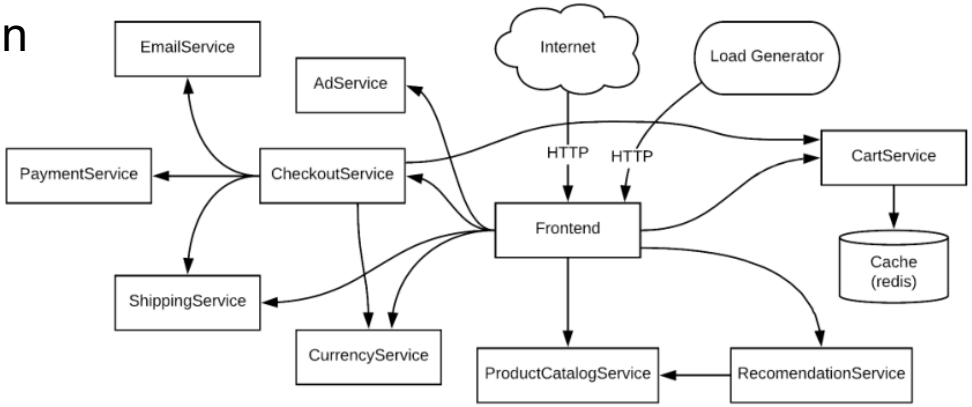
Federated deployments

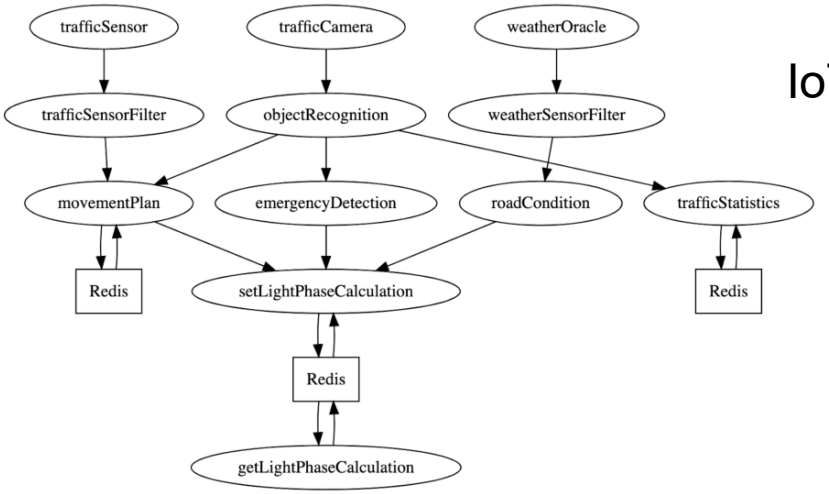




Workloads

E-commerce application (web shop)





IoT application (smart traffic lights)

Technische Universität Berlin

Testing and benchmarking infrastructure

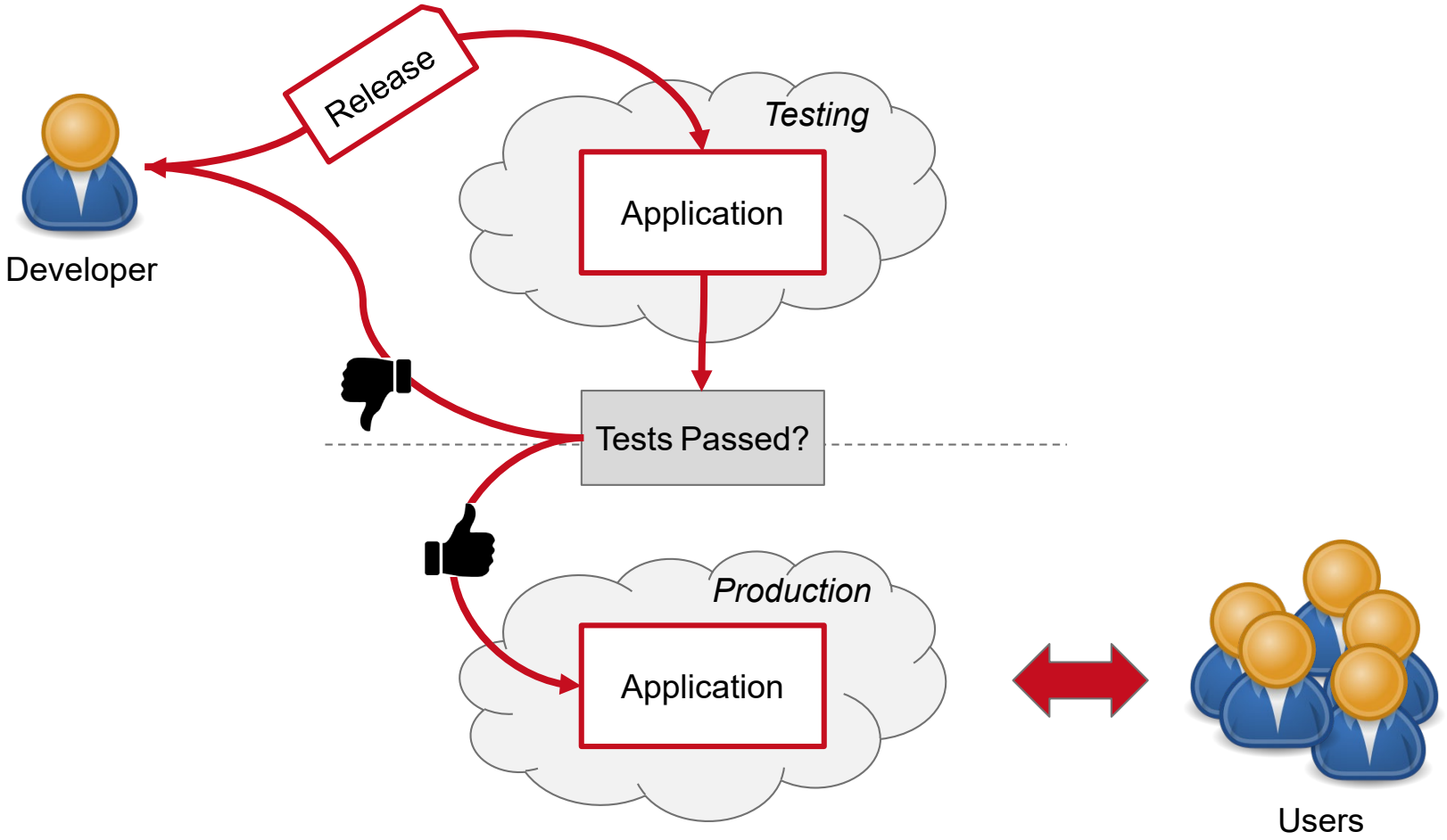
MockFog

Hasenburg et al. MockFog: Emulating Fog Computing Infrastructure in the Cloud. IEEE ICFC 2019.
Hasenburg et al. MockFog 2.0: Automated Execution of Fog Application Experiments in the Cloud. IEEE TCC 2021.

Mobile Cloud Computing

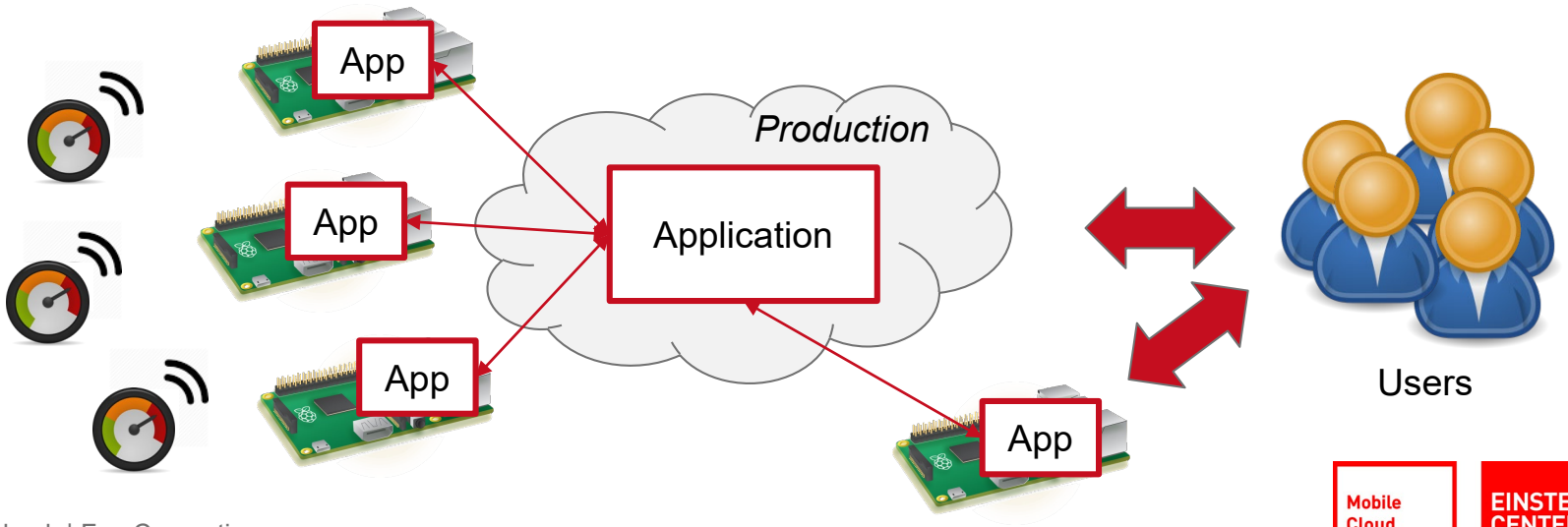
EINSTEIN CENTER Digital Future

Motivation: Rolling out Cloud applications



Motivation: Rolling out Fog applications

What is my testing infrastructure?



How to evaluate your Fog Application?

Technische Universität Berlin

Without a testing infrastructure:

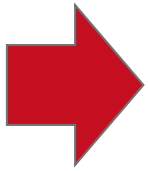
- ➢ Guesses, (small) local testbeds, and simulation

Operate additional edge machines:

- ➢ Expensive, must be at same site as production machines

Idea: Use an emulated fog infrastructure testbed that is set up in the cloud

- Size/Power of VMs: Cloud instance types and Docker resource limits
- Network characteristics: tc, iptables, etc.

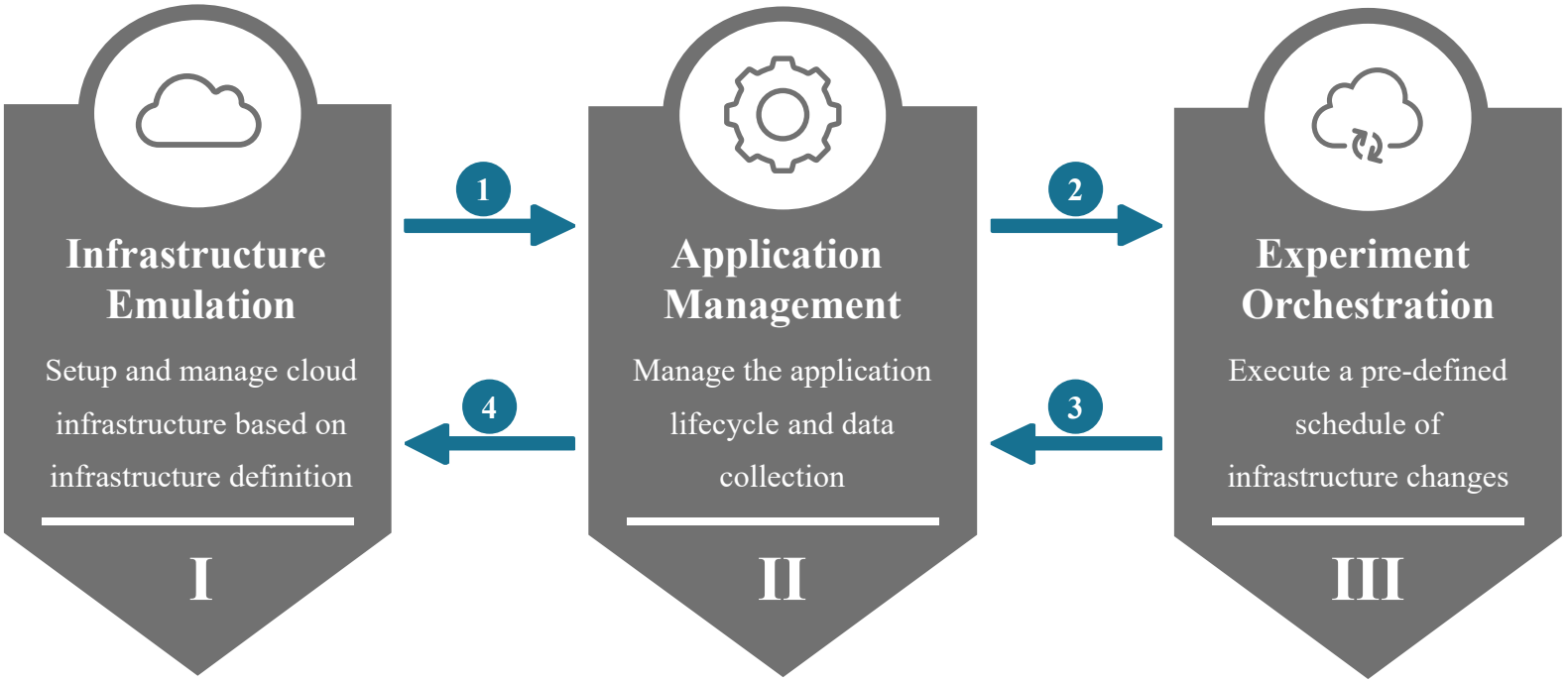


MockFog

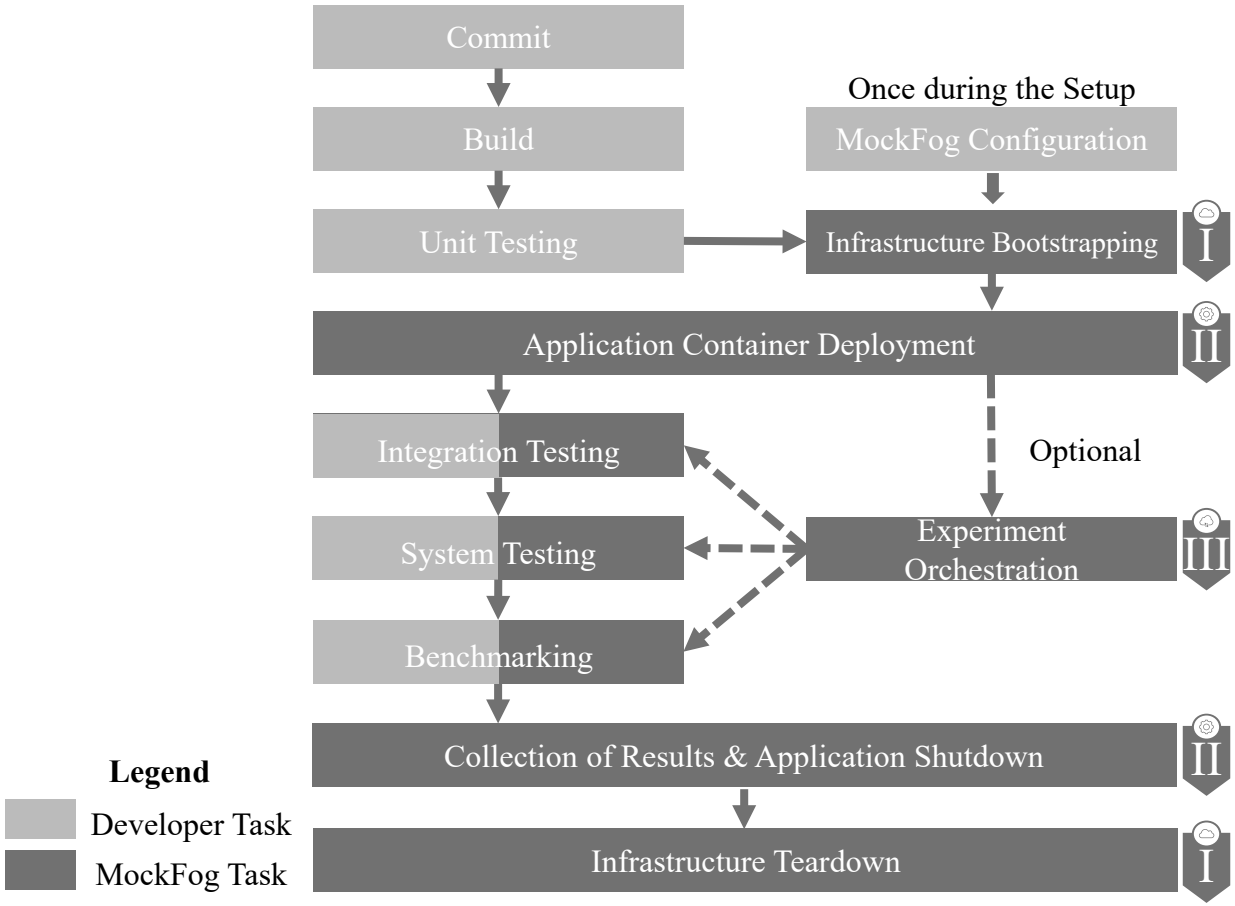
Mobile Cloud Computing

EINSTEIN CENTER Digital Future

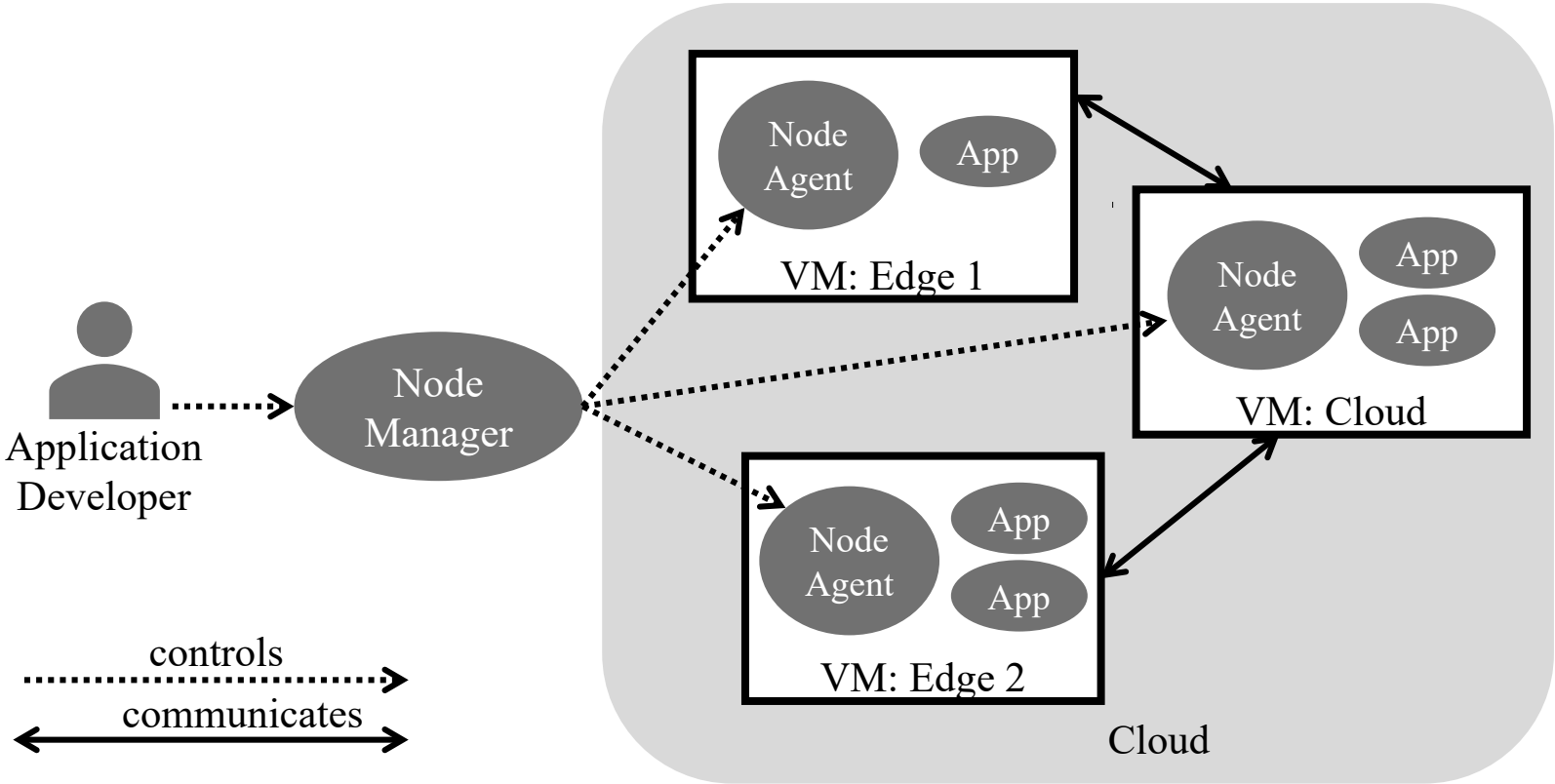
MockFog has three modules



Using MockFog in application engineering

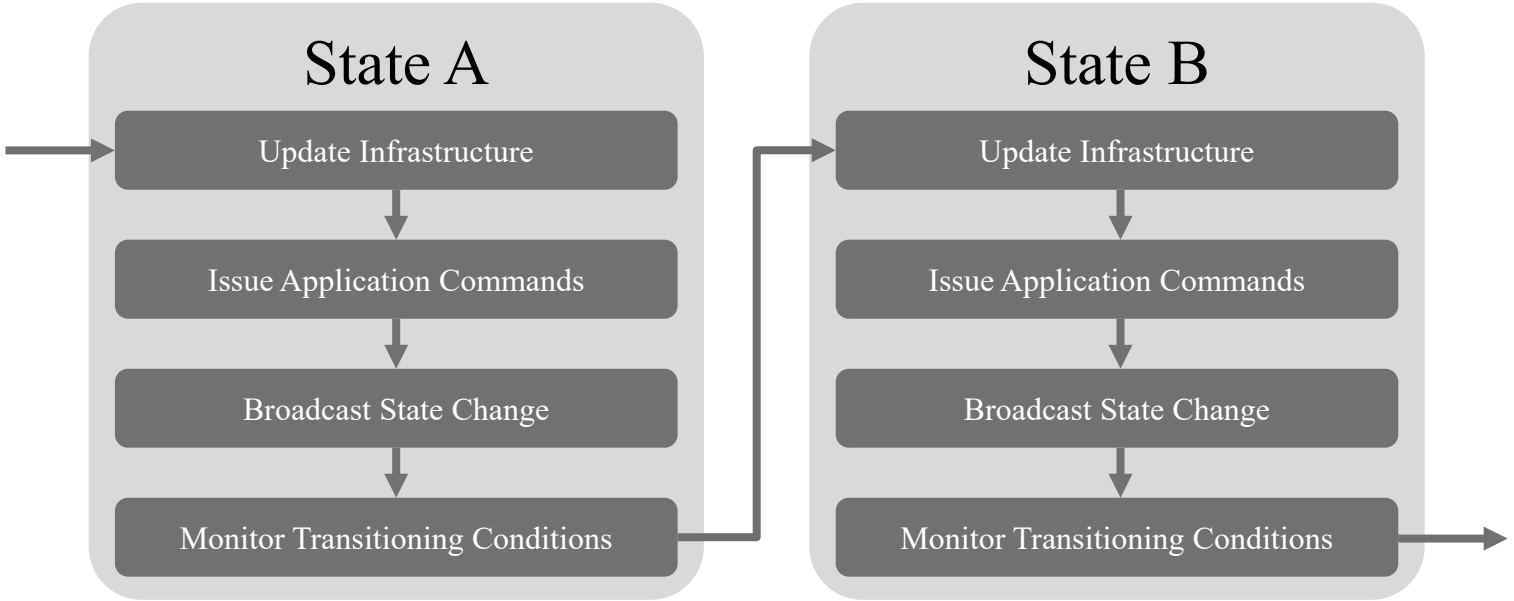


Node Manager and Node Agents

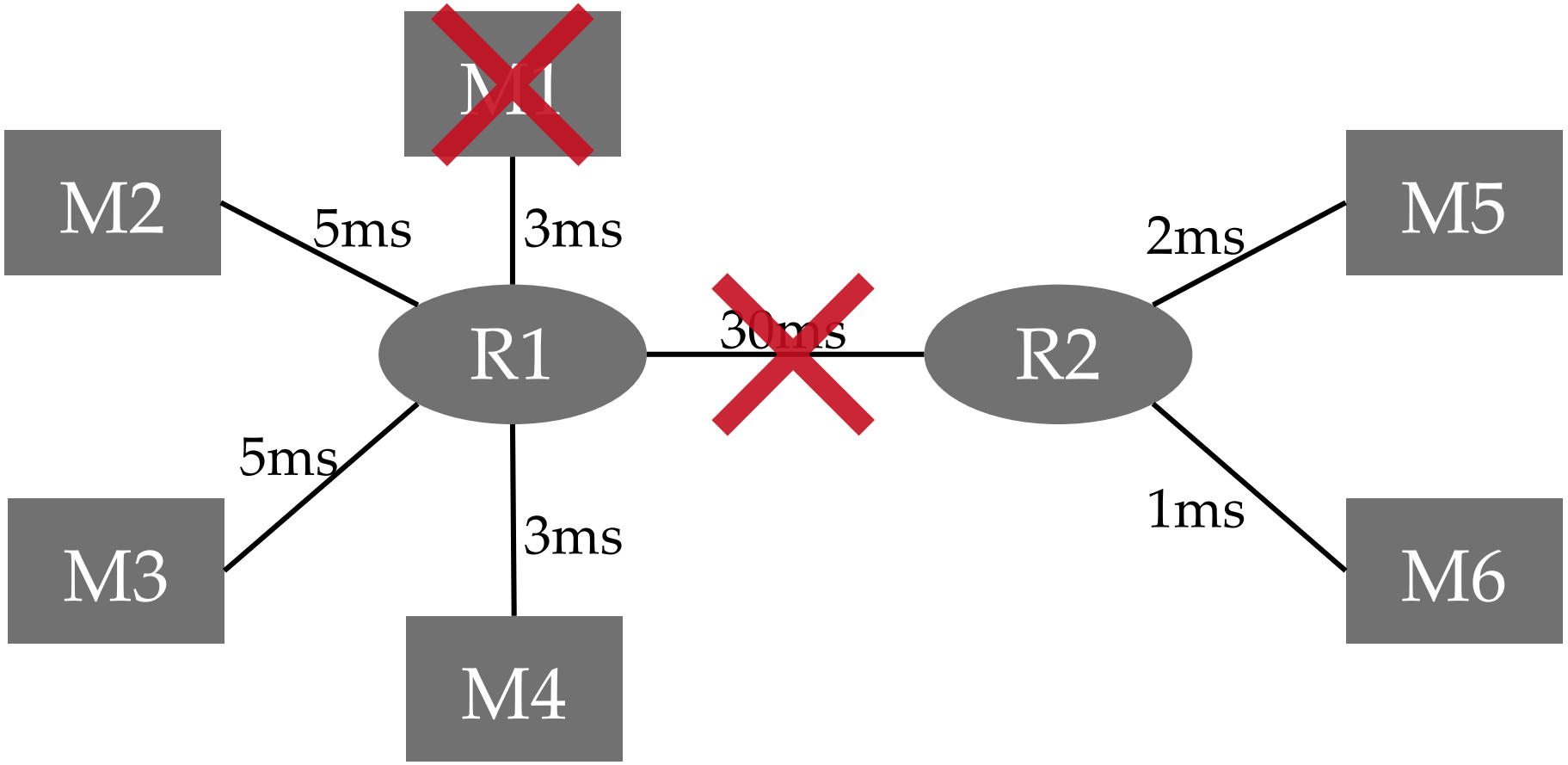


Experiment Orchestration

The orchestration schedule comprises a finite set of states



Failure testing



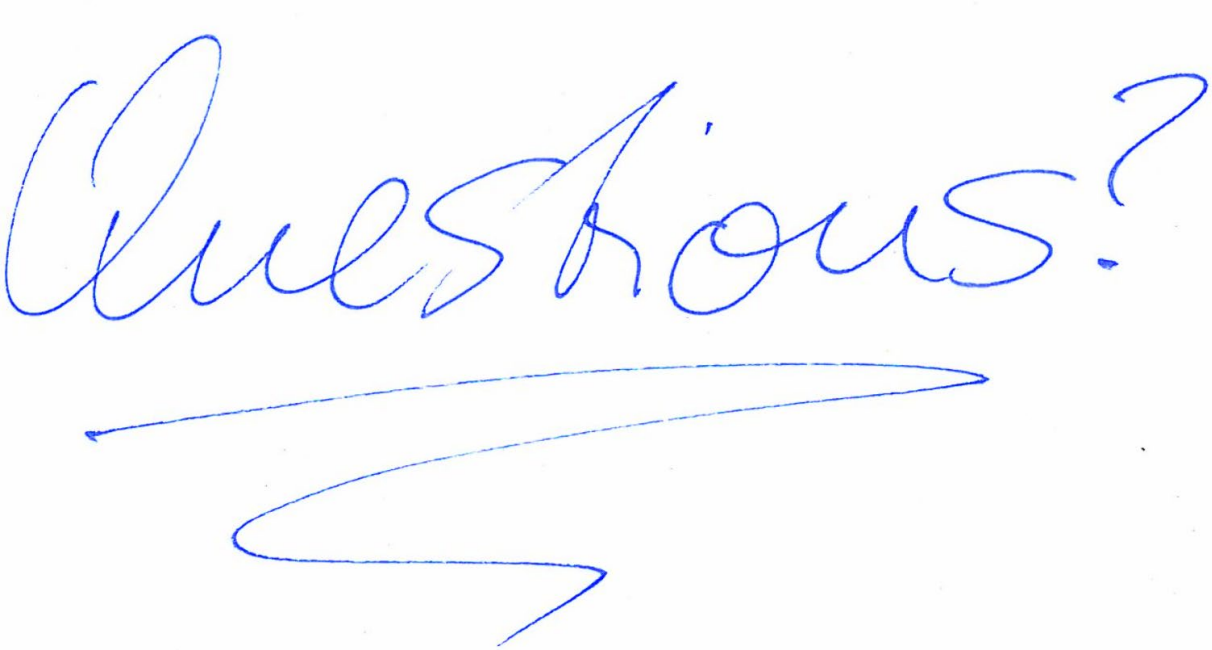
SUMMARY

Summary

Testing != Monitoring != Benchmarking

Live testing approaches can only be used for the cloud part of fog applications
⇒ Mock the rest or
⇒ Use physical testbed

Fog benchmarking is much more complicated
- Multiple workloads
- Geo-distribution
- Deployment of measurement clients for the edge
- More distributed measurements
- …


Questions?