



Edge AI and Beyond

Designing Robust AI Architectures for the 6G Era

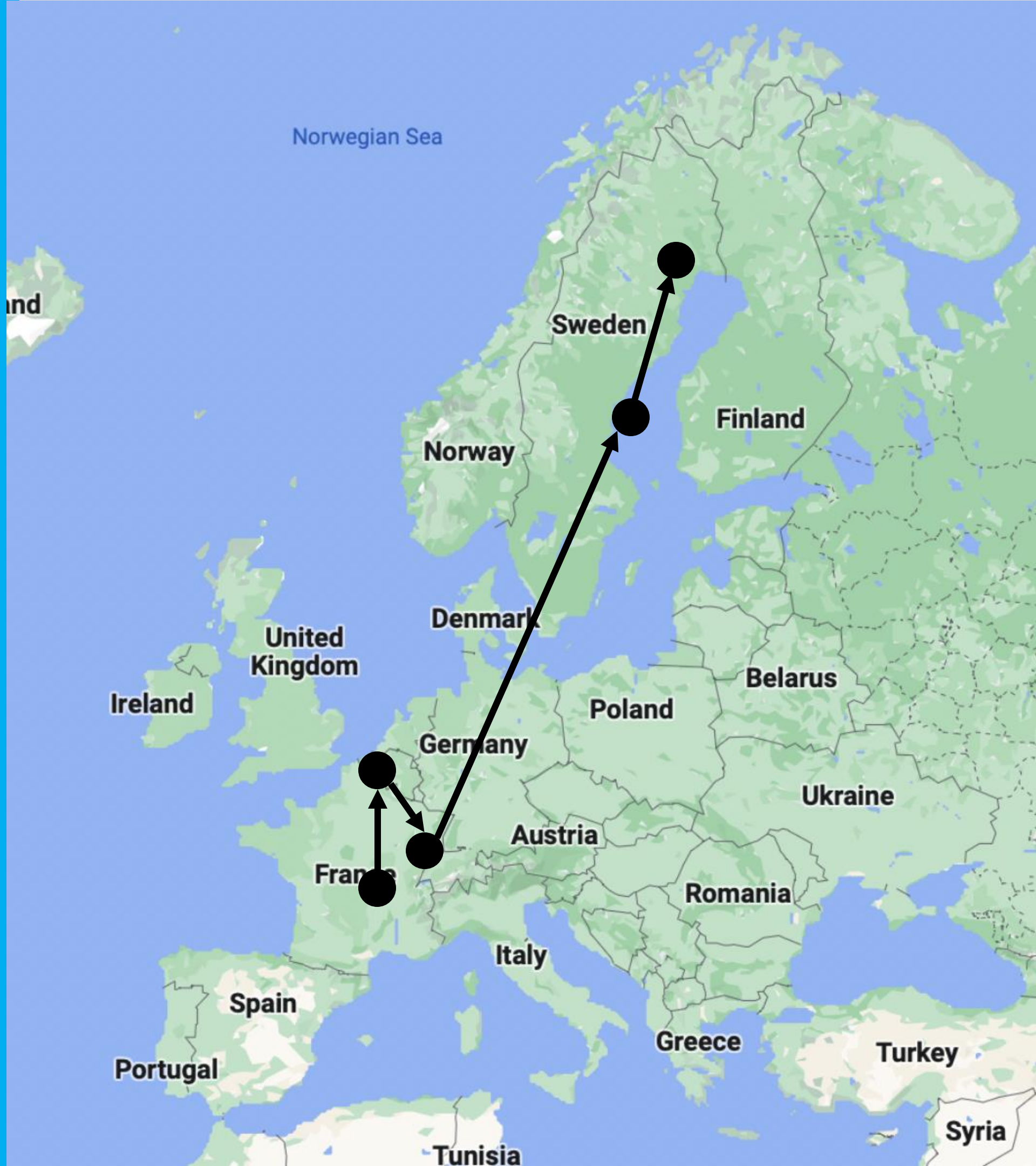
Prof. Davide Taibi

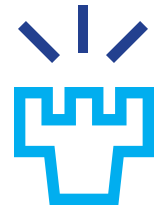
University of Oulu



About me

Full Professor @ University of Oulu
Head of the M3S Cloud Research Group





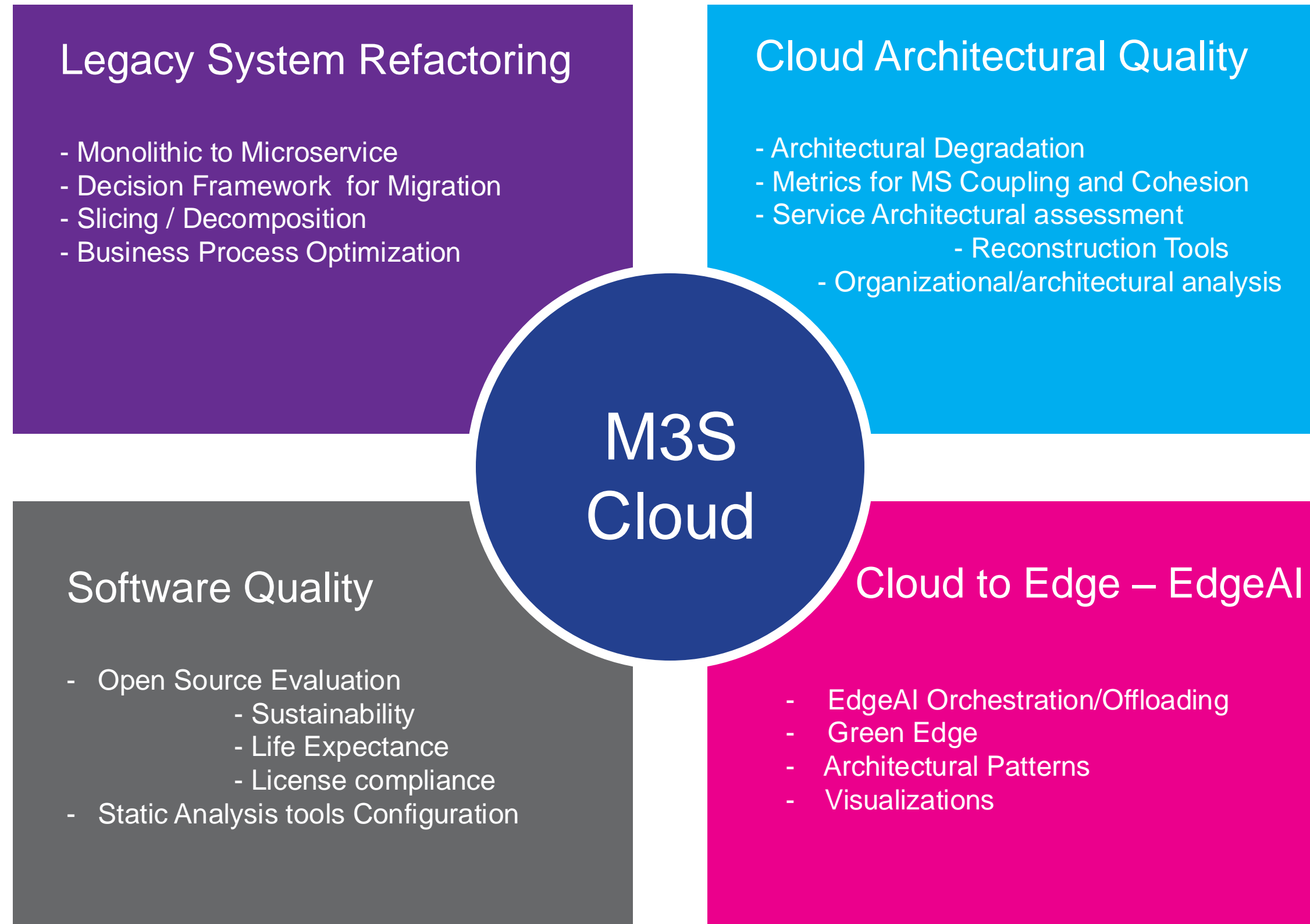
- 3 Post-doctoral
- 6 Ph.D. students
- 3 Lecturers

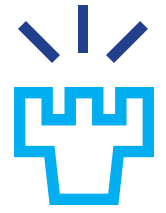
The M3S Cloud Research Group



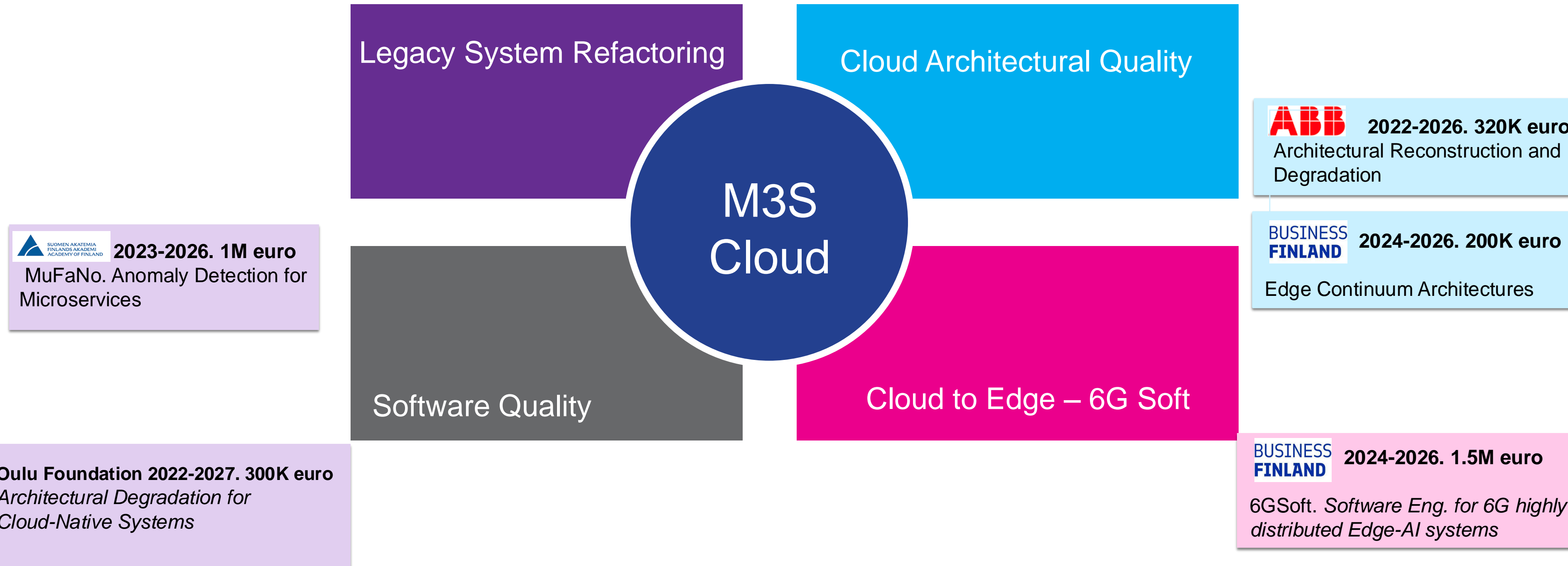


M3S Cloud Research Group





Current Projects



Funded By

NOKIA

 **VENDASTA**



ABB



**Pakistan Science
Foundation**



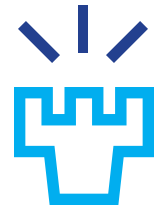
**SUOMEN AKATEMIA
FINLANDS AKADEMI
ACADEMY OF FINLAND**

**BUSINESS
FINLAND**



**Horizon 2020
European Union Funding
for Research & Innovation**





My Research Roadmap on Edge-AI

Edge-AI Architectures

- Best/Bad Practices
- Patterns / Anti-Patterns
- Tools pipelines
- Extremely distributed AI
- Quantum as a Service

Edge AI Orchestration

- Load balancing
- Scheduling
- Resource planning
- Optimization
- Quantum as a Service



Distributed intelligent computing

Hyperlocality, orchestration & trust



The 6GSoft Project (1.5M)

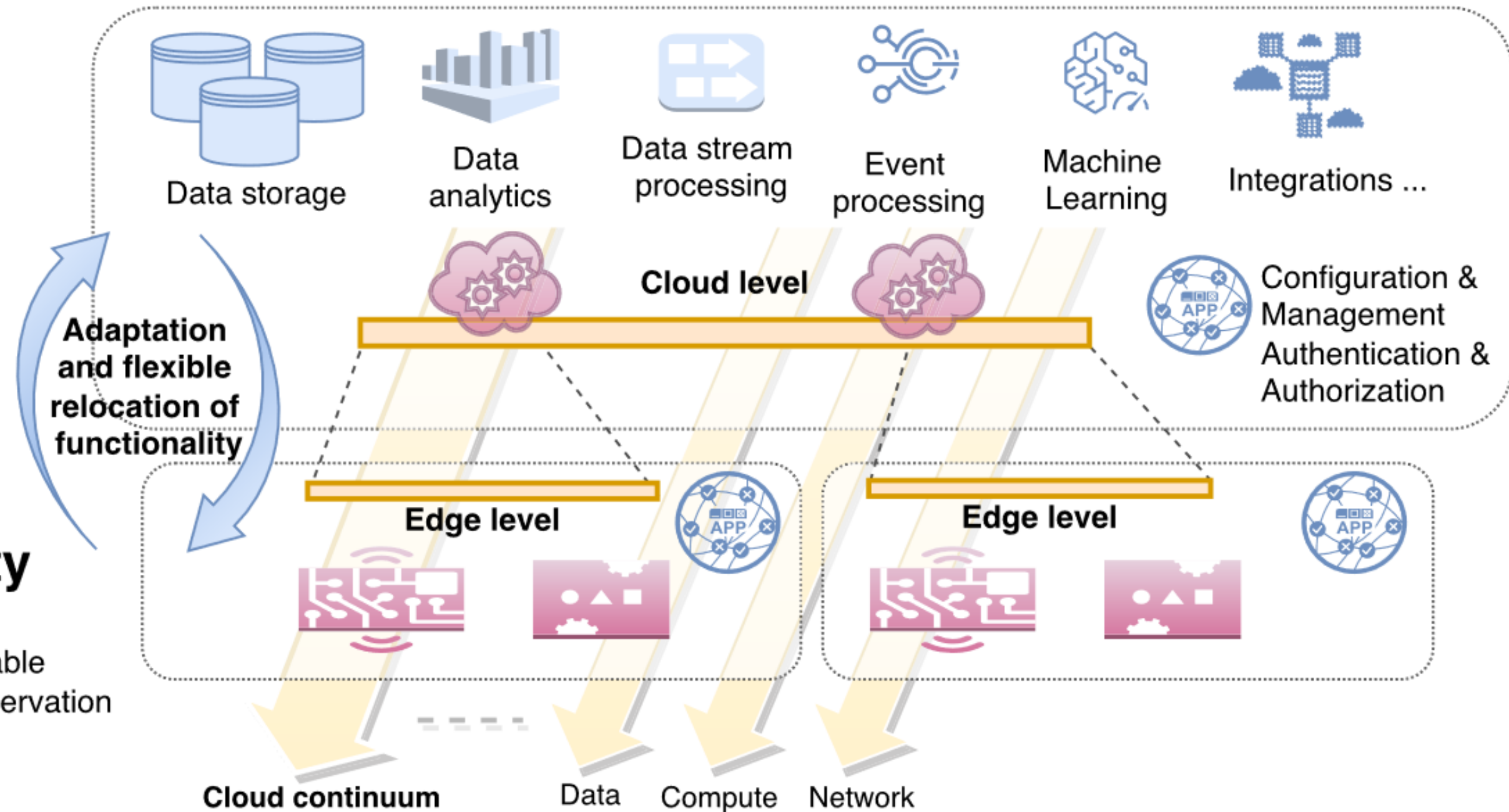
Cloud scalability

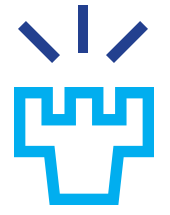
Infinitely scalable resources, wider integration possibilities

Fog flexibility

Edge dependability

Low-latency and deterministic communication capabilities, reliable communications, local data preservation





A vision of the distant future

Everything...

- Sensing
- Connected
- Autonomous
- Multimodal
- Personalized

- **Smart connected devices** are omnipresent
- Assisted automatic common tasks
- Highly personalized devices
- Centralized and coordinated behaviour



- **Research opportunities** in interactions, biometrics, analyzing behaviour, predicting intentions

INTELLIGENT, AI BASED CONNECTED DEVICES

The trend: *Ultra-densification of wireless systems*

” (...communications had) approximately millionfold capacity increase since 1957.

Breaking down these gains shows

- a 25× improvement from wider spectrum,
- a 5× improvement by dividing the spectrum into smaller slices,
- a 5× improvement by designing better modulation schemes, and
- a whopping **1600×** gain through reduced cell sizes and transmit distance

The enormous gains reaped from smaller cell sizes arise from efficient spatial reuse of spectrum or, alternatively, higher area spectral efficiency”.

We should expect **ultradense networks !!**

Near future application 1: *Road safety, overtaking cars*

Seeing through buildings at city street corners, other vehicles and curves of the road



Ultra-dense wireless communications networks needed.
But how to make a scalable application?

Near future application 2: *Augmented environmental information*

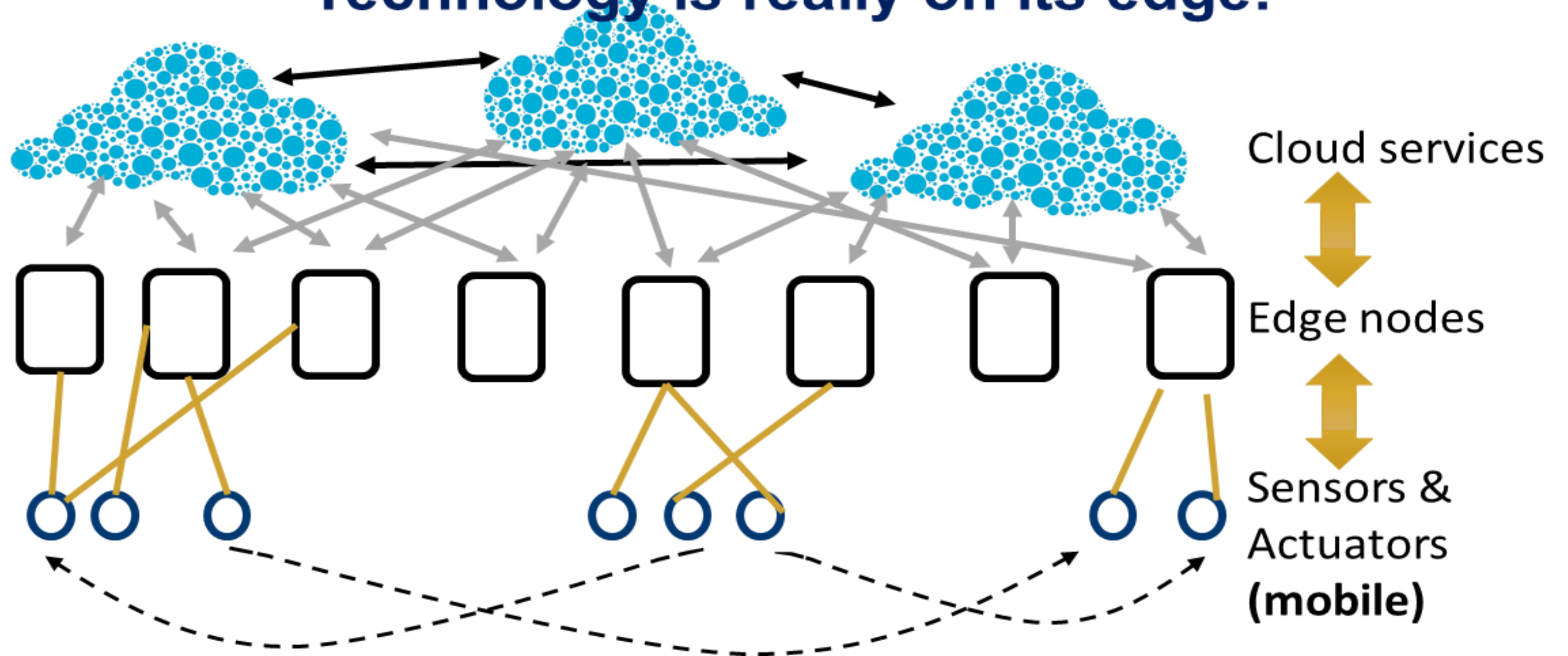
Super-situational awareness for an individual (and about the individual)

- A secure digital twin of an individual follows the person, and connects with relevant sensors in the proximity
- not only just environment information, but also "eyes in the back"
 - tracking of health related information
 - continuous aggregation and personalization of models



Ultra-dense communications and sensor networks that supports the mobility of individuals. **But how?**

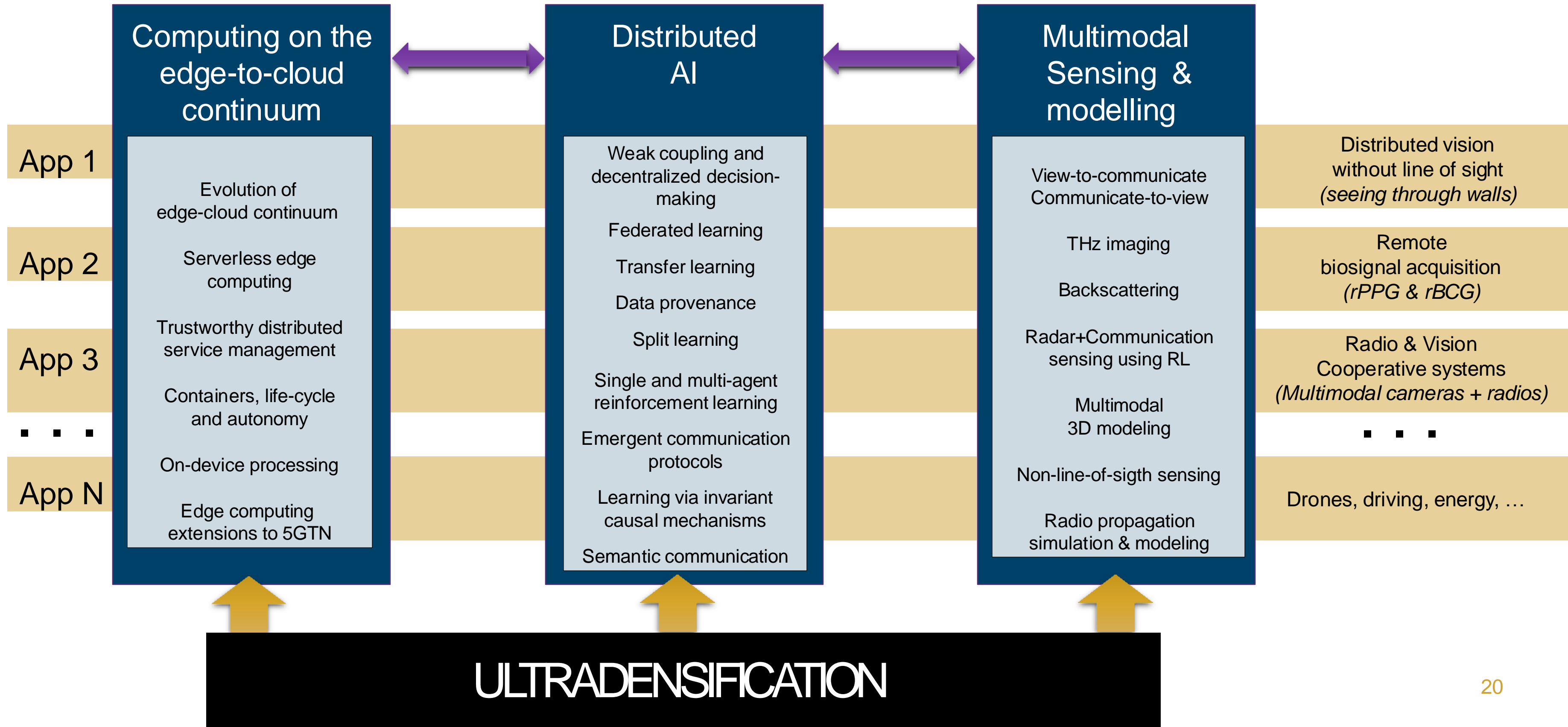
Technology is really on its edge!



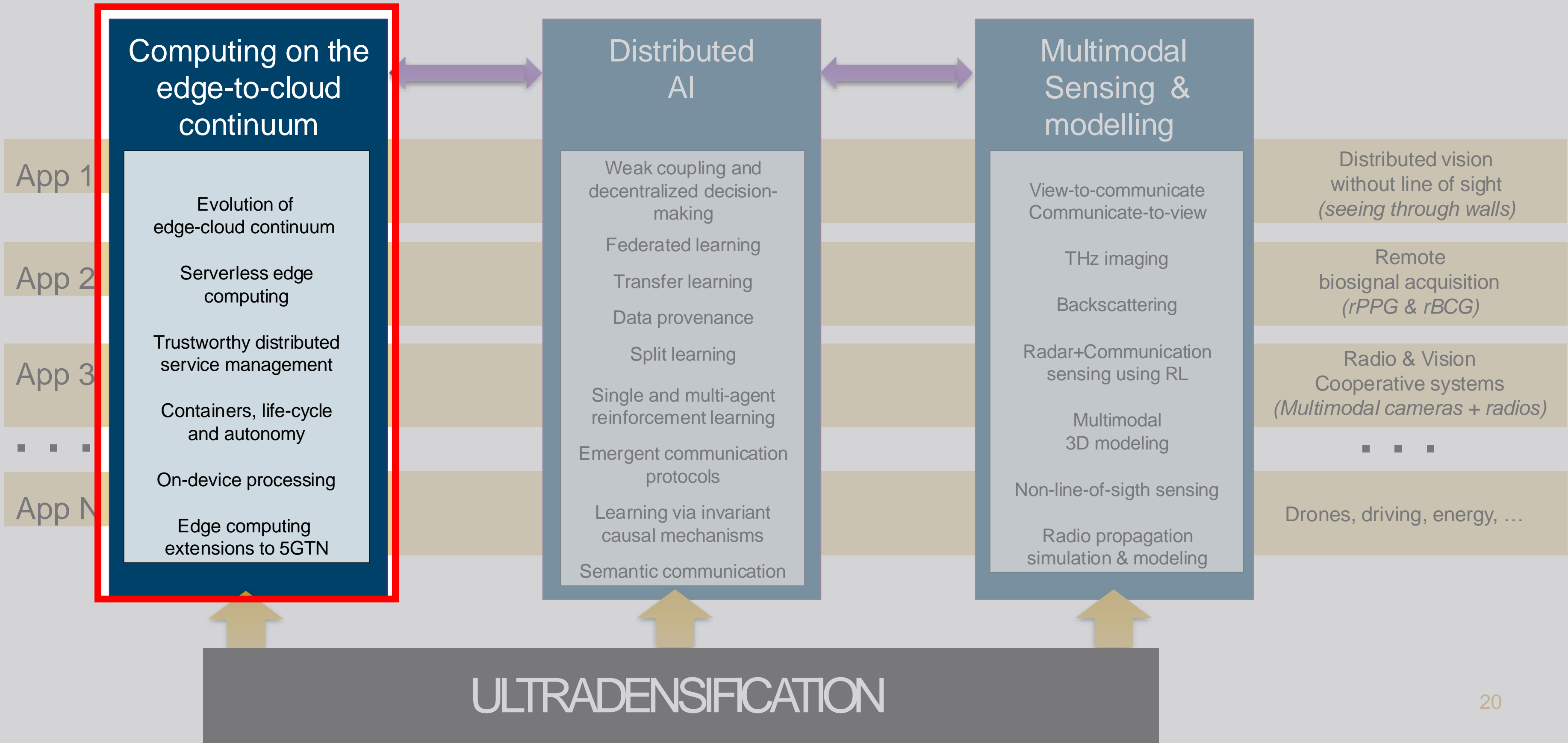
Implementing apparently straightforward application scenarios involves **big logistical challenges** concerning

- distribution of machine learning,
- seamless application mobility,
- information security and privacy.

Distributed Intelligence Overview

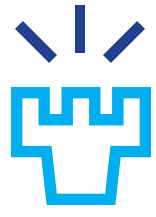


Distributed Intelligence Overview



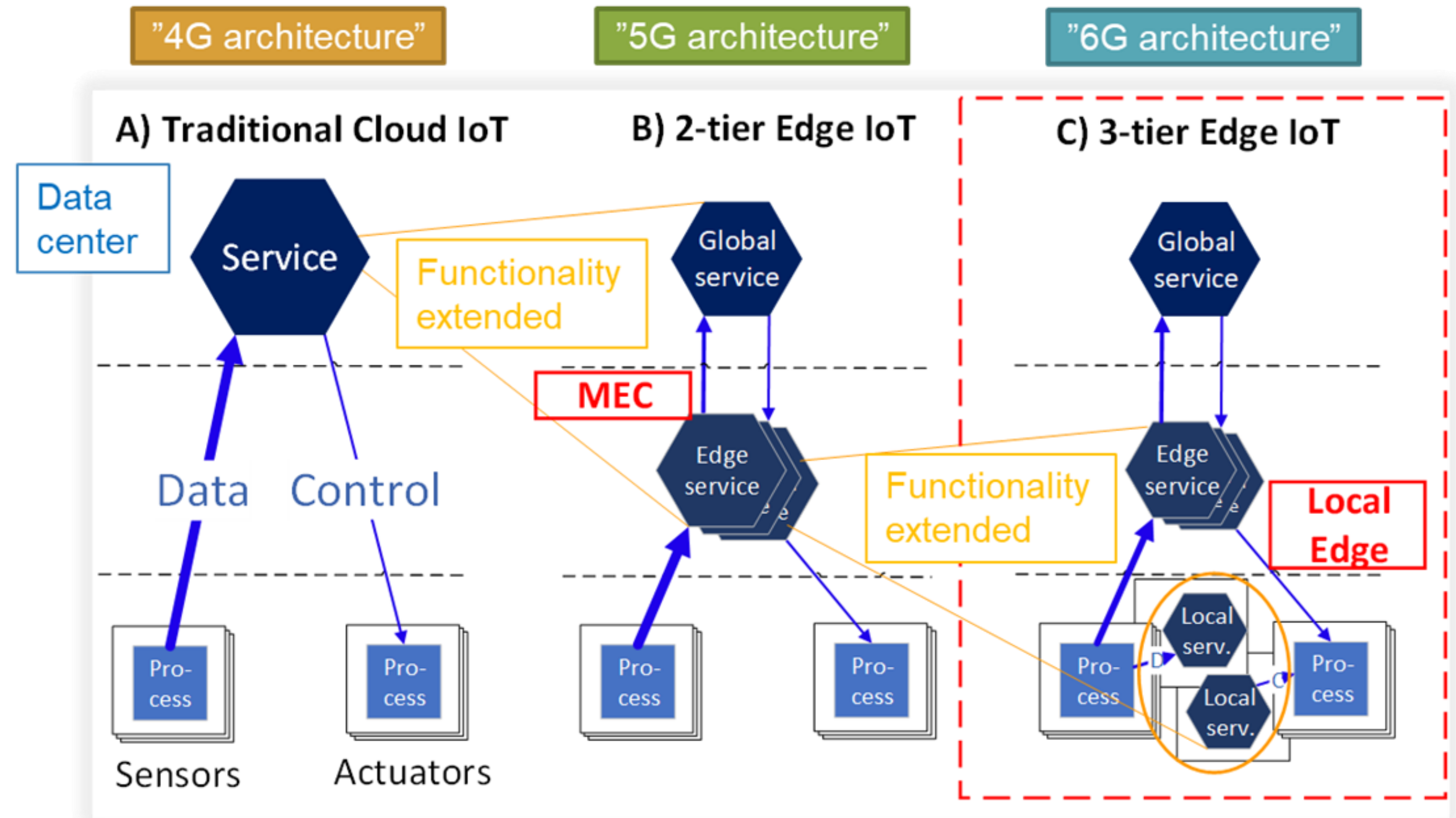
Operating Cloud Native AI based Systems

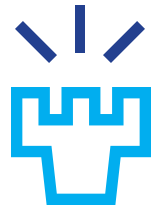




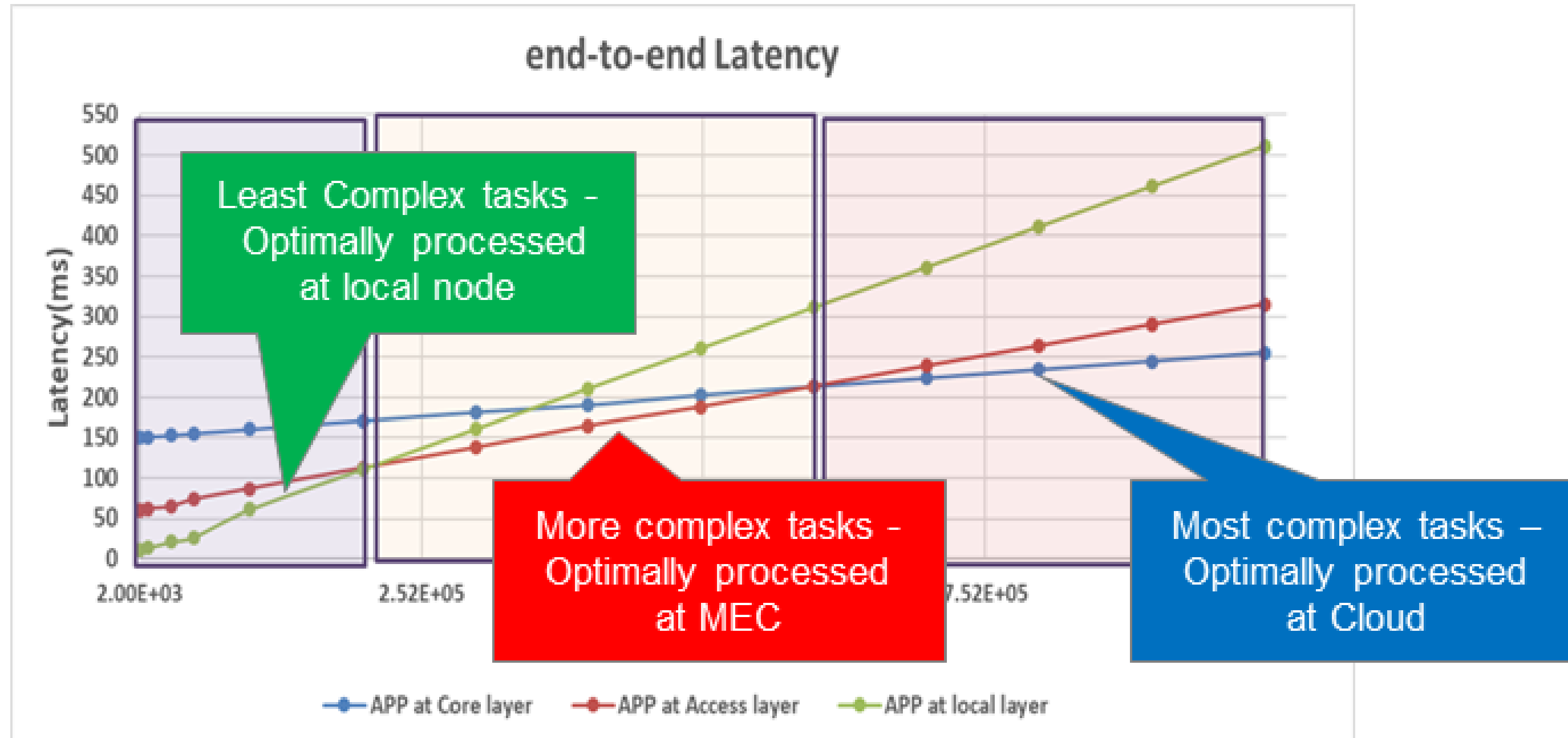
Computing on the edge-to-cloud continuum

- The three-tier edge architecture helps optimizing latency, network usage and power consumption by allowing capacity-aware placement of computational tasks on different tiers.
- Time varying model splitting and compression, accounts for uncertainties in processing & communication, lowers energy consumption and CO2 emission
- Resource consumption and service deployment & initiation times on a feasible level.





Computing on the edge-to-cloud continuum



Computing on the edge-to-cloud continuum

Orchestration:

Local orchestrator deploys lightweight granular microservices in containers

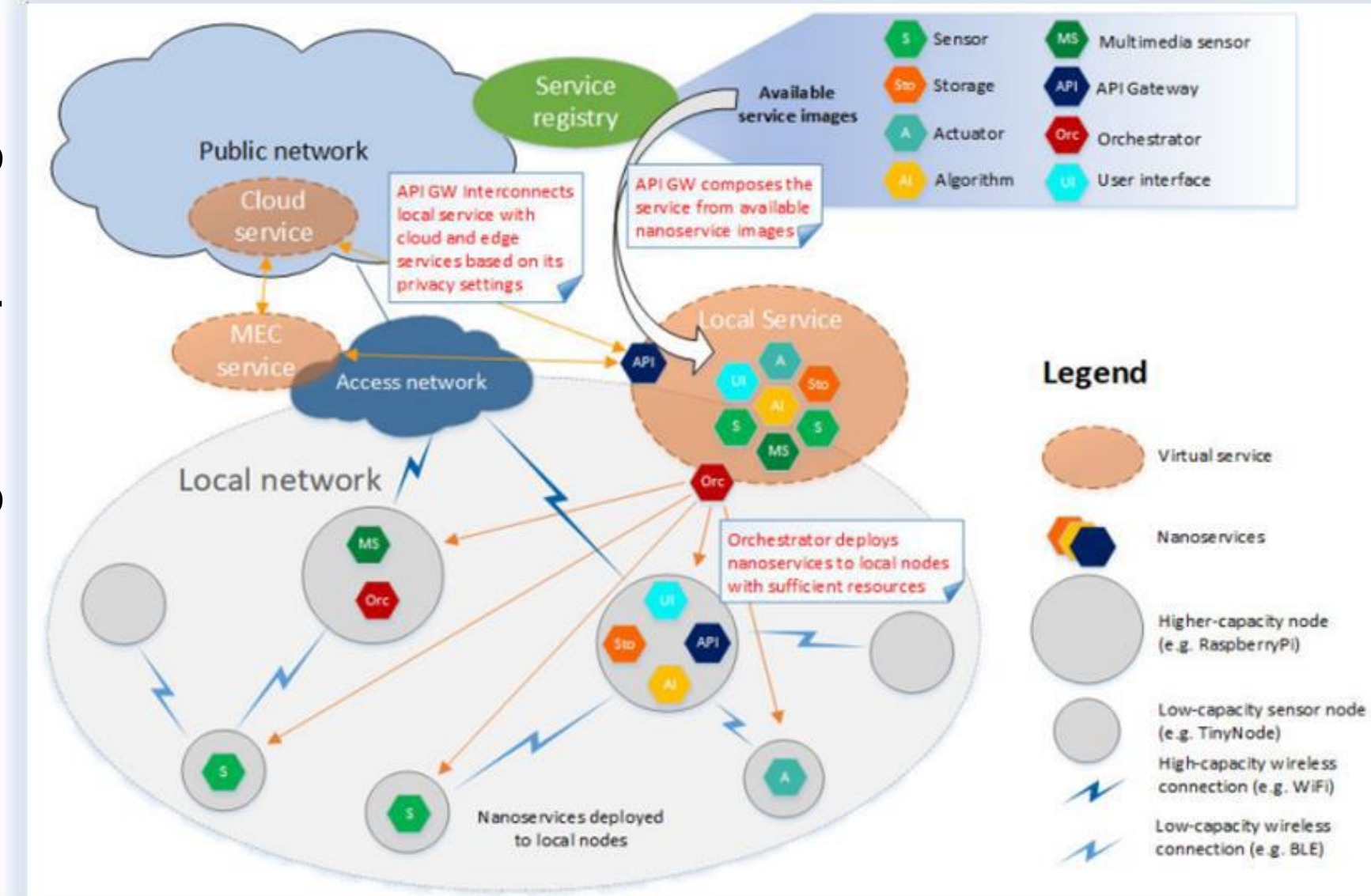
Local API gateway can compose local services of on small virtual microservices (nanoservices).

Orchestrator takes care of deployment, redeployment and undeployment

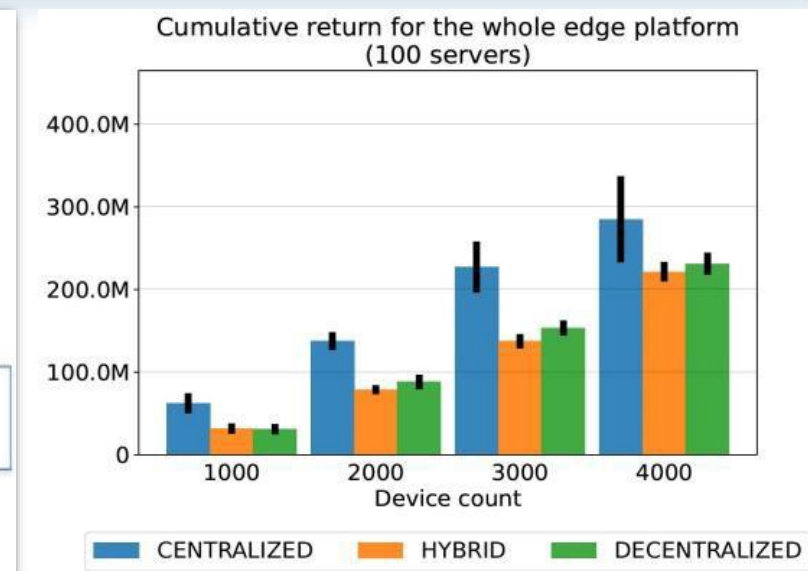
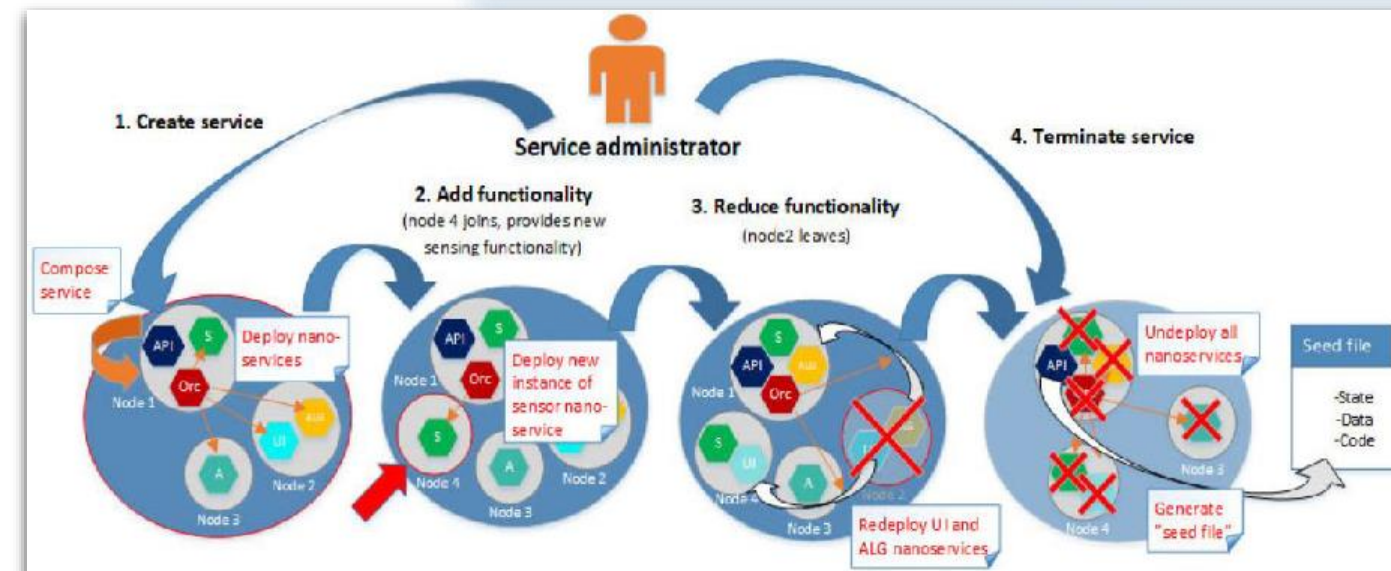
- New devices with new capacity can join the system and existing nodes may leave the system while service is running.

Orchestrator can be simulated !!

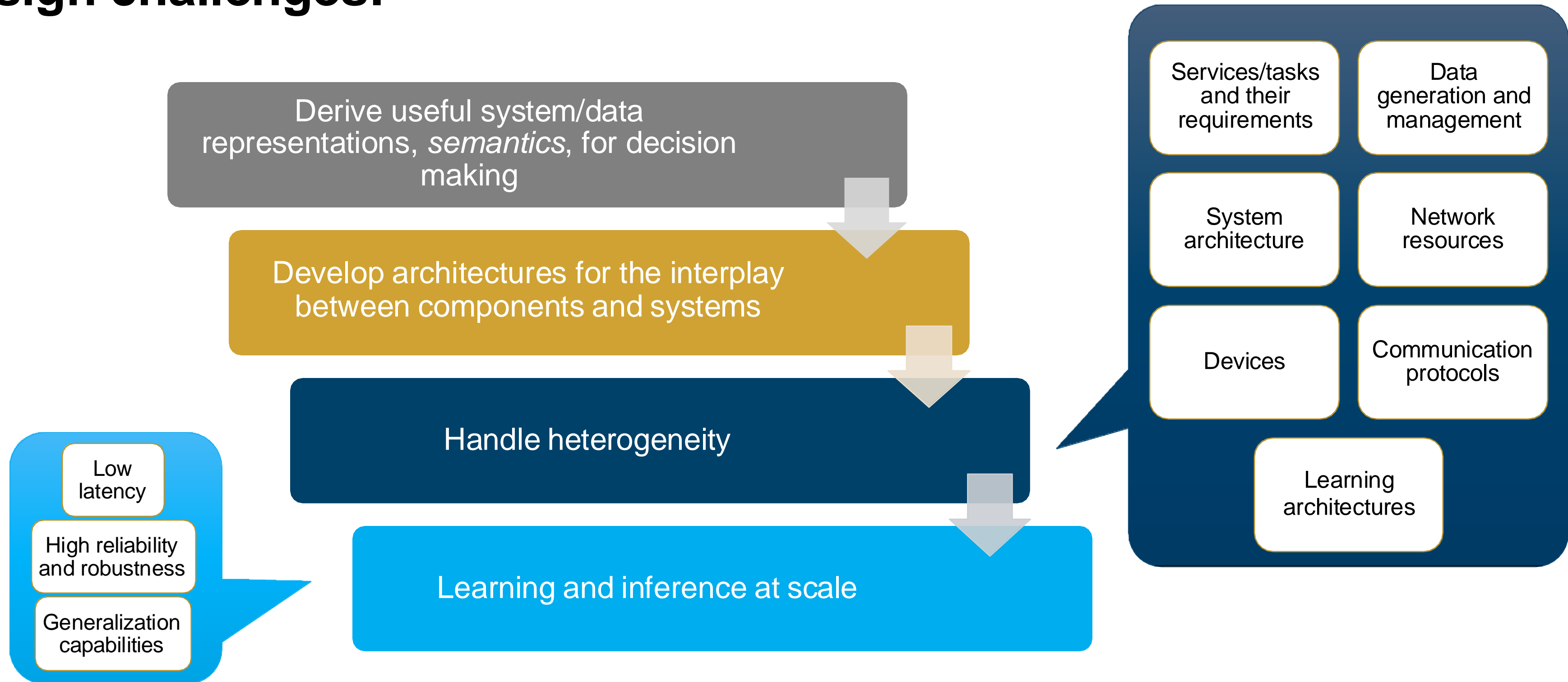
Serverless edge computing

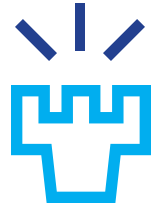


- S. Moreschini, F. Pecorelli, X. Li, S. Naz, D. Hästbacka and D. Taibi, "Cloud Continuum: The Definition," in IEEE Access, vol. 10, pp. 131876-131886, 2022
- A. Droob, D.Morratz, F.Langkilde Jakobsen, J. Carstensen, M.Mathiesen, R.Bohnstedt, M.Albano, S.Moreschini, D.Taibi. Fault Tolerant Horizontal Computation Offloading," 2023 IEEE International Conference on Edge Computing and Communications. 2023
- Cognitive Cloud: The Definition. 19th International Conference on Distributed Computing and Artificial Intelligence, 2023



Design challenges:





6G Software for Extremely Distributed and Heterogeneous Massive Networks of Connected Devices

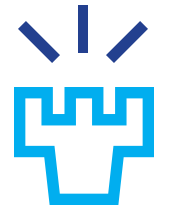
Our Research on Edge Computing

“Investigate sustainable software solutions that are robust, scalable, and energy-efficient.”

1. Implement energy-aware (EA) orchestration and scalability models
2. Software architecture for energy-aware extremely distributed systems

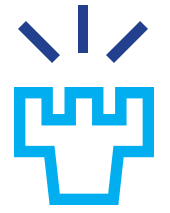
May 2023 - April 2026

**BUSINESS
FINLAND**



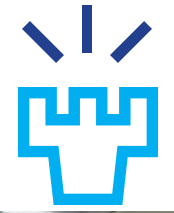
Energy-Aware orchestration and scalability models

- Investigated auction-based orchestration methods
 - Isomorphic implementations
- Investigated multi-layered cloud-native systems architecture reconstruction

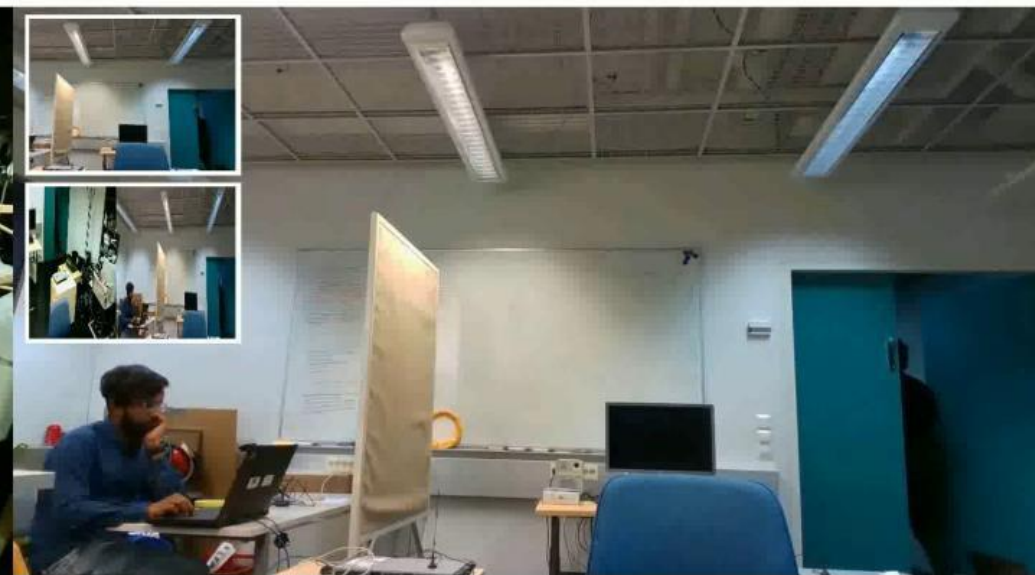
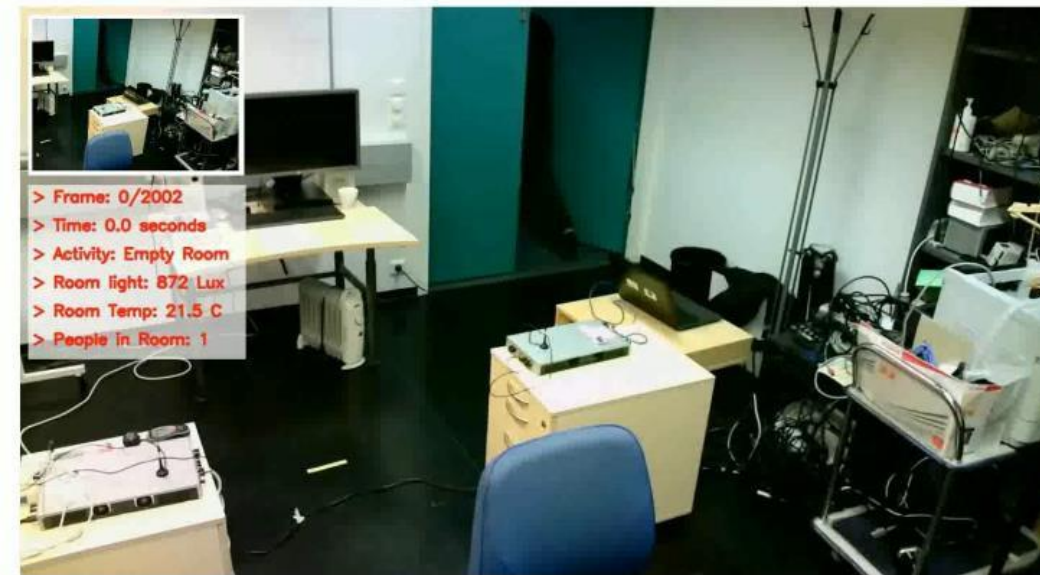
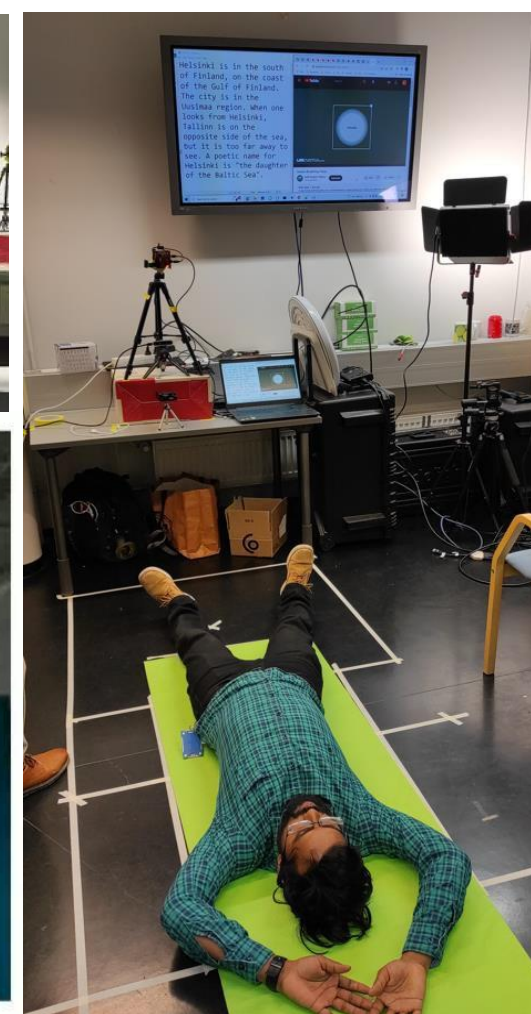
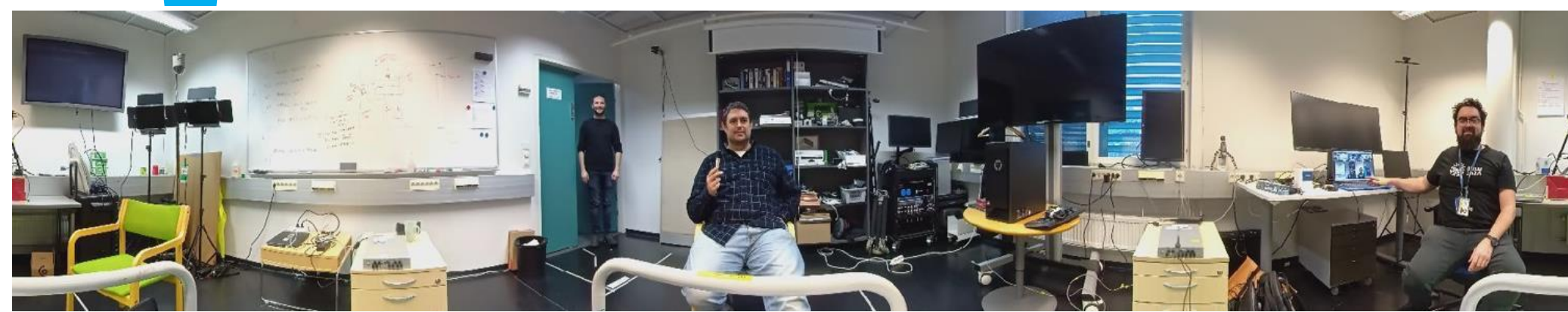


Energy-Aware software architecture

- Prototypes for next generation software applications
 - Decentralised deployment in the continuum
 - SW Architecture for AI-Based models

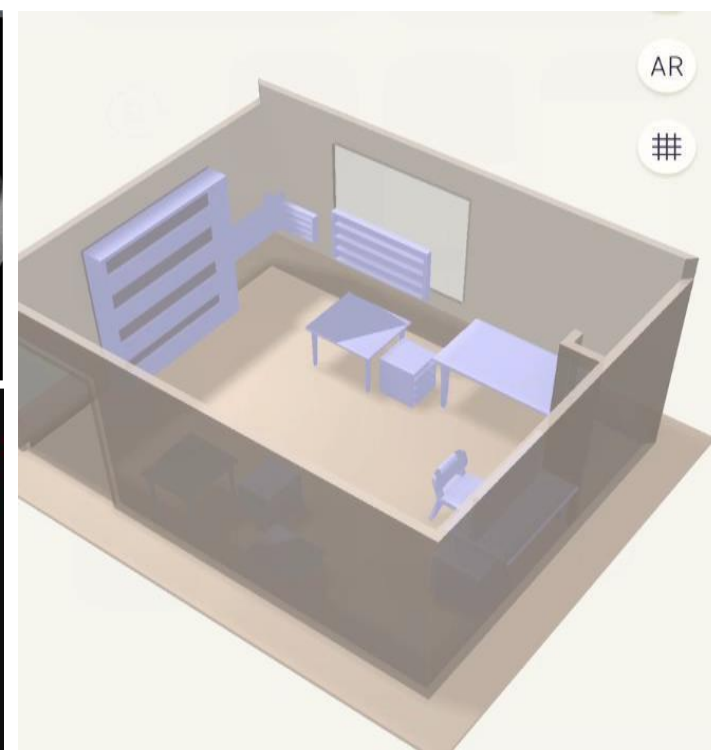
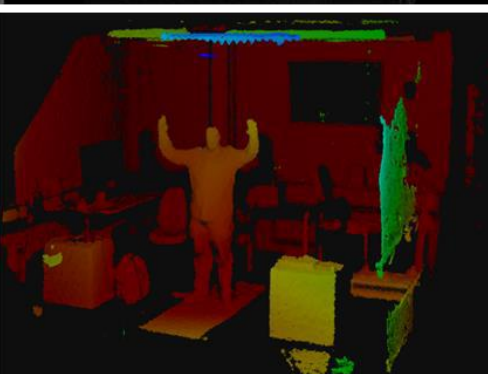
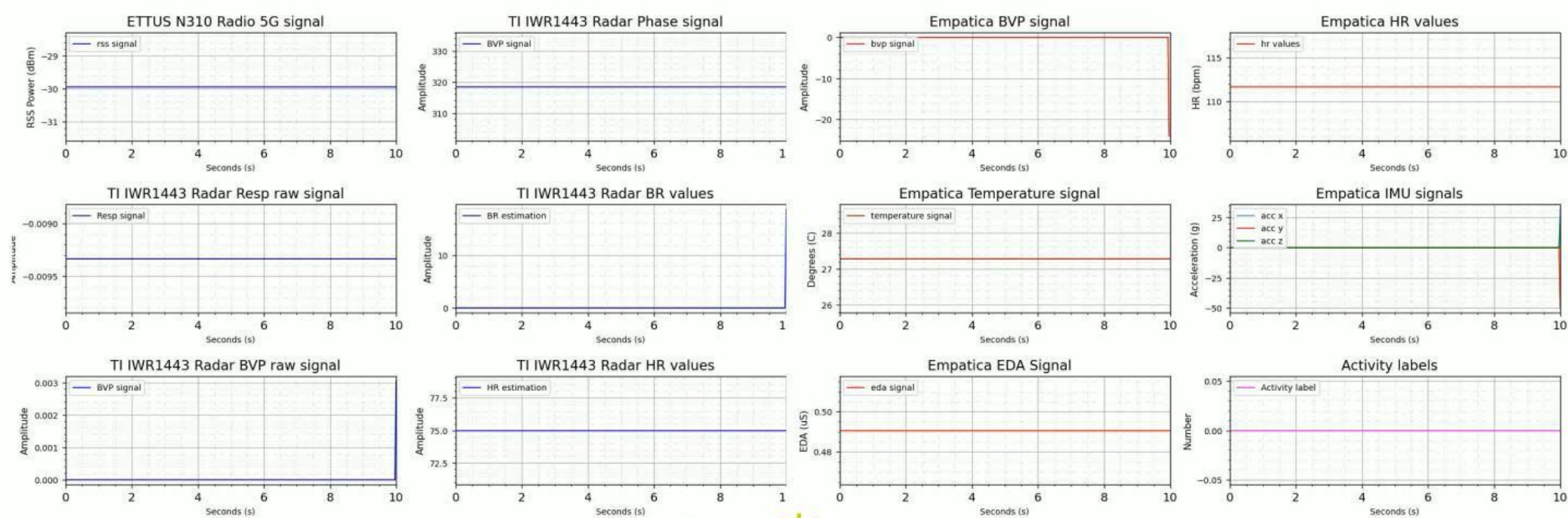


Demo space: Multimodal Sensing Lab



[ROOM SENSORS SIGNALS]

[REFERENCE SIGNALS + LABELS]



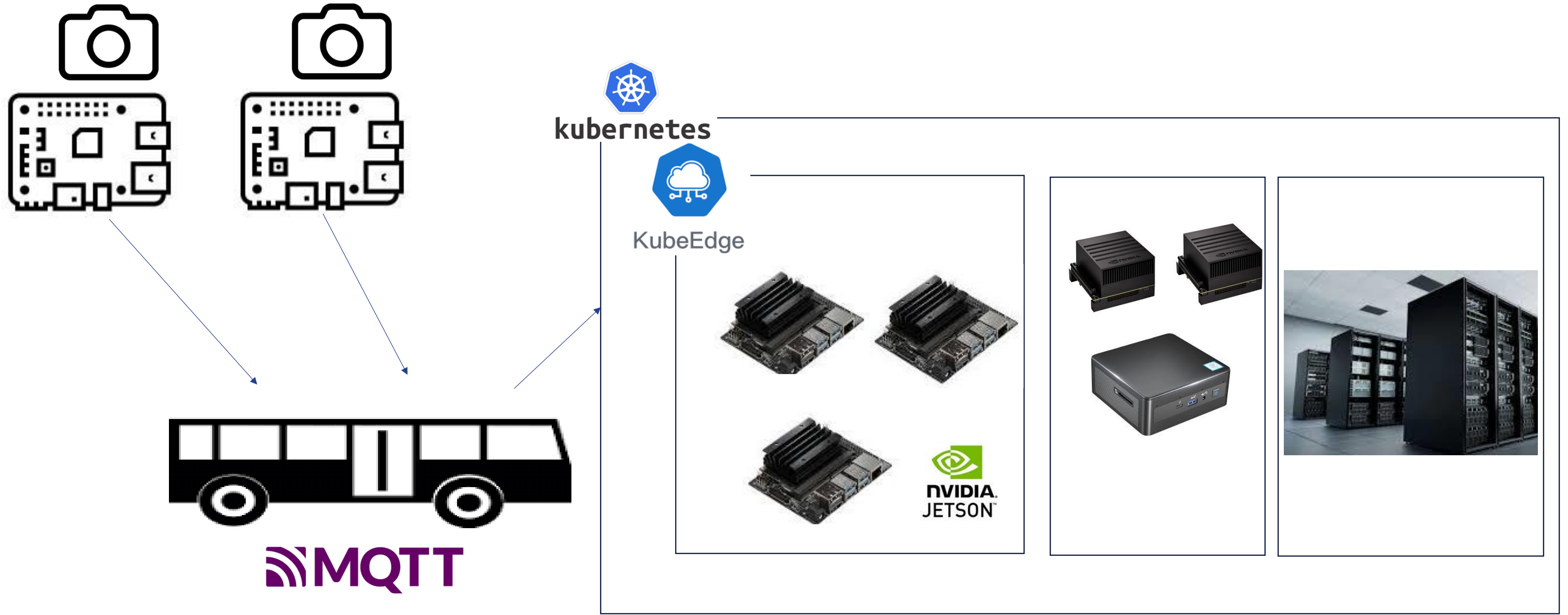
Demo Space: Office Wellbeing Use Case

Use case of Edge-AI on office wellbeing

- Posture monitoring (cameras)
- Torso mobility (Balance Boards)
- Time Standing/seating/balance board
- Heart rate (stress)

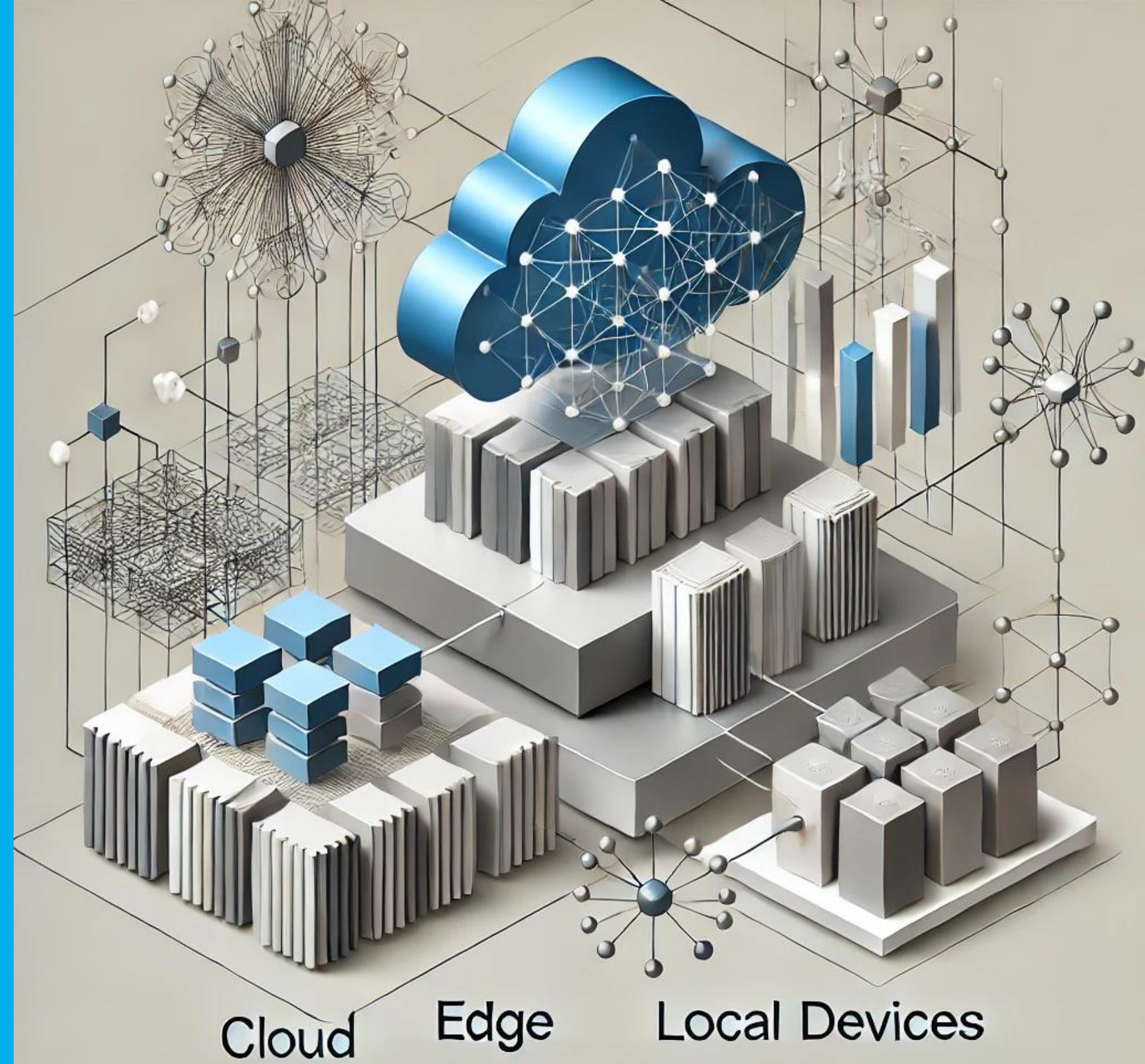


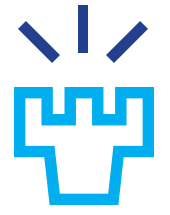
Our Multimodal Lab Architecture





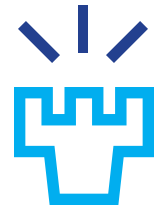
Architecting Edge-AI Systems



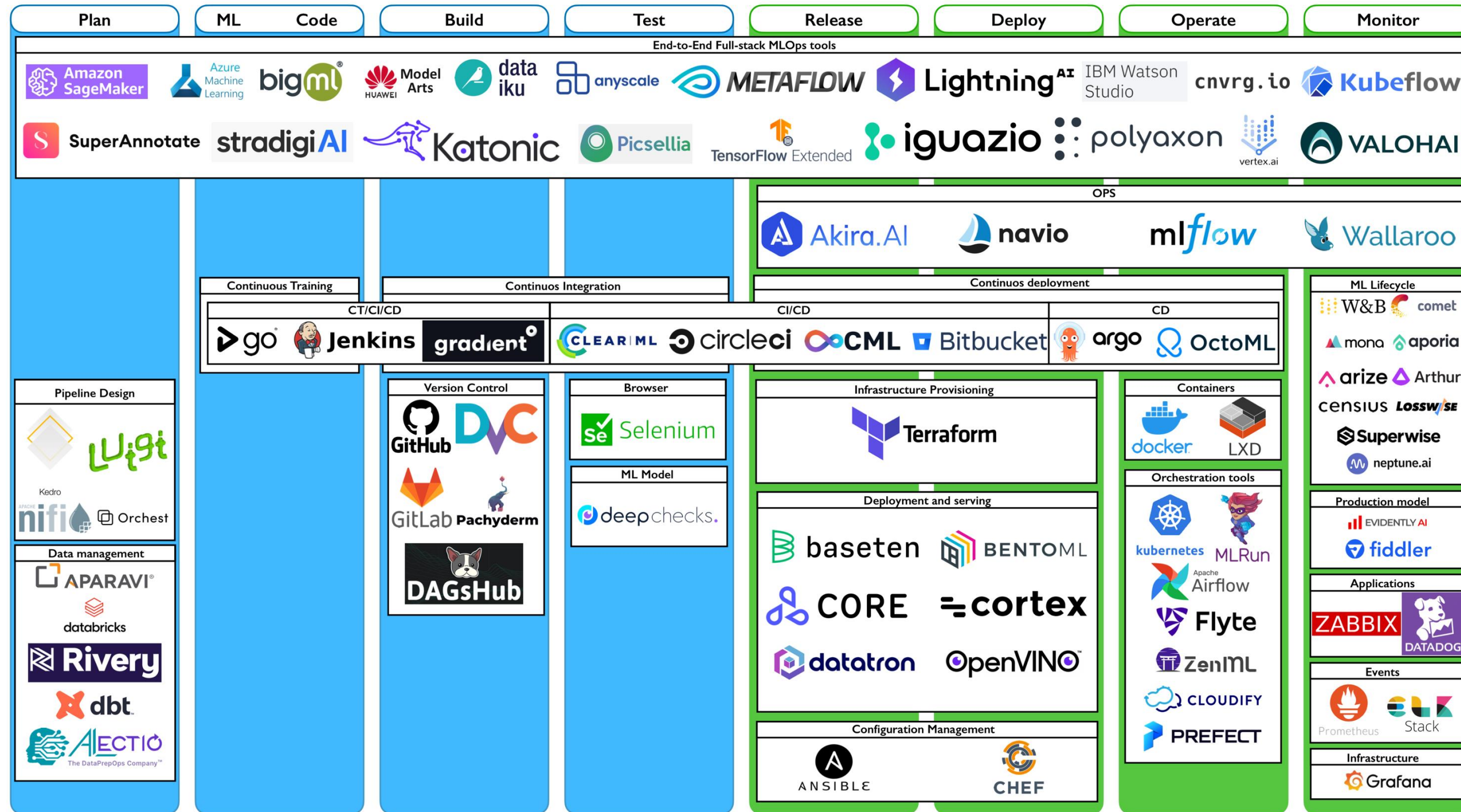


Challenges

- No standardized AI-based system reference architecture
- No standard tools pipeline
- Ad-hoc research studies



Tools Pipelines: AIOps tool Map



Sergio Moreschini, Elham Younesian, David Hästbacka, Michele Albano, Jiří Hošek, Davide Taibi, Edge to cloud tools: A Multivocal Literature Review, **Journal of Systems and Software**. Volume 210, 2024,

Best Practices and Bad Practices

Microservices



Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation

DAVIDE TAIHI, VALENTINA LENARDUZZI, and CLAUS PAHL

DAVIDE TAIHI, Valentina Lenarduzzi, and Claus Pahl, Free University of Bozen-Bolzano

Abstract—Serverless, the new buzzword, has been gaining a lot of attention from the developers and industry. Cloud vendors such as AWS and Microsoft have hyped the architecture almost everywhere, from practitioners' conferences to local events, in

Abstract—Serverless, the new buzzword, has been gaining a lot of attention from the developers and industry. Cloud vendors such as AWS and Microsoft have hyped the architecture almost everywhere, from practitioners' conferences to local events, in

Serverless



Serverless Computing—Where Are We Now, and Where Are We Heading?

DAVIDE TAIHI, Tampere University
JOSEF SPITNER, Zurich University of Applied Sciences
KORNER WERNICH, Thales.com

Abstract—Serverless, the new buzzword, has been gaining a lot of attention from the developers and industry. Cloud vendors such as AWS and Microsoft have hyped the architecture almost everywhere, from practitioners' conferences to local events, in

Abstract—Serverless, the new buzzword, has been gaining a lot of attention from the developers and industry. Cloud vendors such as AWS and Microsoft have hyped the architecture almost everywhere, from practitioners' conferences to local events, in

Micro-Frontends



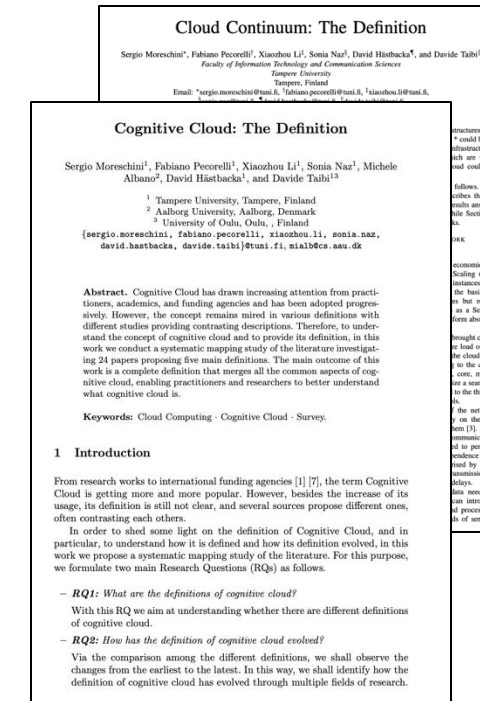
Motivations, benefits, and issues for adopting Micro-Frontends: A Multivoiced Literature Review

SERVI Peltanen¹, Luca Mezzalana², Davide Taihi^{3*}

¹Tampere University, Tampere, Finland
²York University, North York, Canada
³Tampere University, Tampere, Finland

Abstract—Micro-Frontends are becoming increasingly popular, being adopted by several large companies, such as SAP, IBM, Starbucks and many others. Micro-Frontends enable splitting of monolithic frontends into independent and smaller frontends. However, many companies are still hesitant to adopt Micro-Frontends, due to the lack of knowledge concerning their benefits. Additionally, provided state-of-the-art literature review is not comprehensive. This paper aims to address the existing knowledge on Micro-Frontends, by understanding the motivations of companies when adopting such application as well as possible benefits and issues related to this process, as perceived by academic and any literature by means of the Multivoiced Literature Review process, analyzing 171 sources, of which 43 reported motivations, benefits and issues. Results: The results show that existing architectural options to build web applications are evaluated if the application and development team sizes, and if multiple teams need to develop the same frontend application. In such cases, companies adopt Micro-Frontends to increase team independence and to reduce the overall complexity of the frontend. The application of an Micro-Frontend, combined to several benefits, can Micro-Frontends enabled to provide the same benefits as microservices in the back end side, including the development team size, a fully cross-functional development team that can work in parallel with each other. However, Micro-Frontends also showed some issues, such as the increased product size of the application, increased code duplication and complex business logic, and increasing complexity. Conclusions: Micro-Frontends allow companies to scale development according to business needs in the same way as microservices do with the back end side. In addition, Micro-Frontends have a lot of potential and require careful planning if an advantage is obtained by using Micro-Frontends, rather than to develop a centrally managed front end. By helping practitioners to understand how to use Micro-Frontends as well as understand in which context they are the most beneficial.

Cognitive and Continuum Cloud



Cloud Continuum: The Definition

Sergio Moroschitz¹, Fabiano Pecorelli², Xiaohou Li¹, Sonia Naz¹, Davide Taihi³, Alberto David Härtel⁴, and Davide Taihi^{1,2}

¹Tampere University, Tampere, Finland
²Aalborg University, Aalborg, Denmark
³University of Oulu, Oulu, Finland
⁴University of Applied Sciences, Oulu, Finland

[sergio.moroschitz@utu.fi, fabiano.pecorelli@aalb.aau.dk, xiaohou.li@utu.fi, sonia.naz@utu.fi, david.haertel@utu.fi, david.taihi@utu.fi, niall@cs.aau.dk]

Cognitive Cloud: The Definition

Abstract. Cognitive Cloud has drawn increasing attention from practitioners, academia, and funding agencies and has been adopted progressively. However, the concept remains mixed in various definitions with different studies providing contrasting descriptions. Therefore, to understand the concept of cognitive cloud and to provide its definition, in this work we conduct a systematic mapping study of the literature investigating 24 papers proposing five main definitions. The main outcome of this work is a complete definition that merges all the common aspects of cognitive cloud, enabling practitioners and researchers to better understand what cognitive cloud is.

Keywords: Cloud Computing · Cognitive Cloud · Survey.

1 Introduction

From research works to international funding agencies [1] [7], the term Cognitive Cloud is getting more and more popular. However, besides the increase of its usage, its definition is still not clear, and several sources propose different ones, often contrasting each other.

In order to shed some light on the definition of Cognitive Cloud, and in particular, to understand how it is defined and how its definition evolved, in this work we propose a systematic mapping study of the literature. For this purpose, we formulate two main Research Questions (RQs) as follows.

- **RQ1: What are the definitions of cognitive cloud?** With this RQ we aim at understanding whether there are different definitions of cognitive cloud.
- **RQ2: How has the definition of cognitive cloud evolved?** Via the comparison among the different definitions, we shall observe the changes from the earliest to the latest. In this way, we shall identify how the definition of cognitive cloud has evolved through multiple fields of research.

Patterns and Anti-Patterns

FOCUS: MICROSERVICES

On the Definition of Microservice Bad Smells

Davide Taibi and Valentina Lenarduzzi, Tampere University of Technology

// To identify microservice-specific bad smells, researchers collected evidence of bad practices by interviewing developers experienced with microservice-based systems. They then classified the bad practices into 11 microservice bad smells frequently considered harmful by practitioners. //



MICROSERVICES ARE CURRENTLY enjoying increasing popularity and diffusion in industrial environments, being adopted by several big players such as Amazon, LinkedIn, Netflix, and SoundCloud. Microservices are relatively small and autonomous services that work together, are modeled around a business capability, and have a single and clearly defined purpose.^{1,2} Microservices enable independent deployment, allowing small teams to work on separated and focused services by using the most suitable technologies for their

job that can be deployed and scaled independently.^{1,2} Microservices are a newly developed architectural style. Several patterns and platforms such as nginx (www.nginx.org) and Kubernetes (kubernetes.io) exist on the market. During the migration process, practitioners often face common problems, which are due mainly to their lack of knowledge regarding bad practices and patterns.^{3,4} However, some practitioners have started to discuss bad practices in microservices. In his book *Microservices Anti-Patterns and Pitfalls*,

with possible these smells, we surveyed experienced de of two years tices they fo ment of m and on how identified a c specific bad open and s dure to deriv the practice The goal practitioners tices altoget more efficien migrating m based system As with smells, whi monly consid design,^{1,6} w specific bad service sm indicators of desired patte practices—th software qu understanda bility, reusab of the system

Background

Several gen detection ta been defined Moreover, specific arch been defined of our know work and, in particular, no empirical studies have proposed bad practices, antipatterns, or smells specifically concerning microservices. However, some practitioners have started to discuss bad practices in microservices. In his book *Microservices Anti-Patterns and Pitfalls*,

Microservices Anti-patterns: A Taxonomy

Davide Taibi, Valentina Lenarduzzi, and Claus Pahl

Abstract Several companies are rearchitecting their monolithic information systems with microservices. However, many companies migrate to microservices without experience, mainly learning how to migrate from books or from practitioners' blogs. Because of the novelty of the topic, practitioners and consultancies are learning by doing how to migrate, thus facing several issues but also several benefits. In this chapter, we introduce a catalog and a taxonomy of the most common microservices anti-patterns in order to identify common problems. Our anti-pattern catalog is based on the experience summarized by different practitioners we interviewed in the last 3 years. We identified a taxonomy of 20 anti-patterns, including organizational (team oriented and technology/tool oriented) anti-patterns and technical (internal and communication) anti-patterns. The results can be useful to practitioners to avoid experiencing the same difficult situations in the systems they develop. Moreover, researchers can benefit from this catalog and further validate the harmfulness of the anti-patterns identified.

1 Introduction

Microservices are increasing in popularity, being adopted by several companies, including SMEs, but also big players such as Amazon, LinkedIn, Netflix, and Spotify.

Microservices are small and autonomous services deployed independently, with a single and clearly defined purpose [11, 14]. Microservices propose to vertically decompose the applications into a subset of business-driven independent services.

D. Taibi (✉) · V. Lenarduzzi
Tampere University, Tampere, Finland
e-mail: davide.taibi@tuni.fi; valentina.lenarduzzi@tuni.fi
C. Pahl
Free University of Bozen-Bolzano, Bolzano, Italy
e-mail: claus.pahl@unibz.it



Patterns for Serverless Functions (Function-as-a-Service): A Multivocal Literature Review

Davide Taibi¹, Nabil El Ioini², Claus Pahl³ and Jan Raphael Schmid Niederkofler²
¹Tampere University, Tampere, Finland

Serverless: From Bad Practices to Good Solutions

Davide Taibi
Tampere University
Tampere, Finland
davide.taibi@tuni.fi

Ben Kehoe
iRobot
New York, USA
bkehoe@irobot.com

Danilo Poccia
Amazon Web Services
London, Great Britain
danielop@amazon.co.uk

Abstract—Serverless computing is increasing its popularity in the industry. However, practitioners still have issues when using it. In this work, we identify the main bad practices experienced by practitioners during the development of serverless-based applications. We interviewed 91 experienced practitioners and analyzed the solutions they adopted to solve the issues generated by the bad practice. Moreover, we propose the most appropriate solutions based on our professional experience. The results can be helpful to other practitioners to avoid facing the same issues, or to understand how to overcome them and to researchers that can better validate them and propose alternative solutions.

Keywords—component, formatting, style, styling, insert (key words)

I. INTRODUCTION (HEADING 1)

Serverless computing, and in particular Function-as-a-Service (FaaS), is one of the most recent technologies that enables building cloud-based software based on components and infrastructure entirely managed by cloud providers [8].

One of the main reasons for the increased diffusion is the support and availability of serverless computing platforms, such as AWS Lambda, Azure Functions, and Google Cloud Functions. Serverless enable developers to focus only on the business logic, leaving all the overhead of monitoring, provisioning, scaling and managing the infrastructure to the cloud service providers, and adopting the pay-as-you-go model, allowing companies to pay only for the amount of computational time they actually use [4].

During the development of serverless-based applications, practitioners often face common problems, which are due mainly to not applying best practices and patterns and anti-patterns [4]. While patterns to create serverless-based applications have been already introduced, anti-patterns are still not clear. Moreover, based on the experience we collected working in collaboration with several companies, we believe that developers might still have some wrong assumptions of serverless bad practices.

In order to help practitioners to understand the most common serverless bad practices, in this work we design and conduct a survey based on face-to-face interviews, to elicit the practices that developers considered as bad. Then, our co-authors Danilo Poccia (AWS Lambda chief architect and evangelist), and Ben

Kehoe (CTO at iRobot) validated the bad practices proposed solutions to overcome them together with the AWS lambda core development team at Amazon. Danilo Poccia Ben Kehoe and the AWS lambda core team are among the most experienced practitioners on serverless-based development, as they were involved in the original creation of Amazon AWS Lambda and they have been following more than 1K customers in the development of serverless-based applications.

Our goal is to help practitioners avoid these bad practices altogether or to help practitioners to deal with these practices when developing serverless-based applications.

The remainder of this paper is structured as follows. Section 2 presents related works. Section 3 described the methods we applied to collect the bad practices and review them. Section 4 presents the results while Section 5 lists and discusses the bad practices. Section 6 discusses the results and presents implications while finally Section 7 draws conclusions and future works.

II. RELATED WORKS

Different serverless patterns have been proposed by practitioners in the last years [7][9]. Practitioners also started to propose some possible issues that should be considered when developing serverless-based applications, while others proposed some anti-patterns in technical talks or technical forums.

As for anti-patterns, only non-peer reviewed works were published. Among them, Joe Emison [3] initially proposed in 2018 four anti-patterns in a practitioner talk: thick middle tier, functions calling functions, multiple "spofs" (single point of failure), and custom code. In the same year, William Anderson [2] proposed four different anti-patterns in Forbes blog post: not careful usage of asynchronous calls, shared code between functions and tight coupling between functions

Rohit Akiwatar [1], in 2019, also wrote a blog post mentioning the shared code between functions as a possible anti-pattern and proposed seven new ones: distributed monolith, complex processing, big data ETL Pipeline, long processing tasks, real-time Communication with IoT, the high granularity of functions, excessive usage of communication protocols.

erhead for provisioning, ting Serverless, by m- patterns for composing tions to solve the same al) In this work, we aim n and reporting possible ying peer-reviewed and), together with benefits assified as orchestration, onclusion] Practitioners ed different solutions to i, while others for some

der of this work, we verless Functions". serverless functions is ng different functions application or part of it the lack of patterns et al., 2019)[51], and tting out this problem ing patterns that are

practitioners and re- e of the art on server- propose a review and a ed both by practition-

literature Review pro- pture both the state of n the field, including reviewed papers) and dustrial whitepapers. We selected 24 stud- end of January 2019. y of architectural pat- ommon problems, to-

Technology Assessment Framework

Information and Software Technology 137 (2021) 106600



Contents lists available at ScienceDirect
Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof



From monolithic systems to Microservices: An assessment framework

Florian Auer^{a,*}, Valentina Lenarduzzi^b, Michael Felderer^{a,c}, Davide Taibi^d

^a University of Innsbruck, Austria

^b LUT University, Finland

^c Blekinge Institute of Technology, Sweden

^d Tampere University, Finland

ARTICLE INFO

Keywords:

Microservices
Cloud migration
Software measurement

ABSTRACT

Context: Re-architecting monolithic systems with Microservices-based architecture is a common trend. Various companies are migrating to Microservices for different reasons. However, making such an important decision like re-architecting an entire system must be based on real facts and not only on gut feelings.

Objective: The goal of this work is to propose an evidence-based decision support framework for companies that need to migrate to Microservices, based on the analysis of a set of characteristics and metrics they should collect before re-architecting their monolithic system.

Method: We conducted a survey done in the form of interviews with professionals to derive the assessment framework based on Grounded Theory.

Results: We identified a set consisting of information and metrics that companies can use to decide whether to migrate to Microservices or not. The proposed assessment framework, based on the aforementioned metrics, could be useful for companies if they need to migrate to Microservices and do not want to run the risk of failing to consider some important information.

1. Introduction

Microservices are becoming more and more popular. Big players such as Amazon,¹ Netflix,² Spotify,³ as well as small and medium-sized enterprises are developing Microservices-based systems [1].

Microservices are autonomous services deployed independently, with a single and clearly defined purpose [2]. Microservices propose vertically decomposing applications into a subset of business-driven independent services. Each service can be developed, deployed, and tested independently by different development teams and using different technology stacks. Microservices have a variety of different advantages. They can be developed in different programming languages, can scale independently from other services, and can be deployed on the hardware that best suits their needs. Moreover, because of their size, they are easier to maintain and more fault-tolerant since the failure of one service will not disrupt the whole system, which could happen in a monolithic system. However, the migration to Microservices is not an easy task [1,3]. Companies commonly start the migration without any experience with Microservices, only rarely hiring a consultant to support them during the migration [1,3].

Various companies are adopting Microservices since they believe that it will facilitate their software maintenance. In addition, companies hope to improve the delegation of responsibilities among teams. Furthermore, there are still some companies that refactor their applications with a Microservices-based architecture just to follow the current trend [1,3].

The economic impact of such a change is not negligible, and taking such an important decision to re-architect an existing system should always be based on solid information, so as to ensure that the migration will allow achieving the expected benefits.

In this work, we propose an evidence-based decision support framework to allow companies, and especially software architects, to make their decision on migrating monolithic systems to Microservices based on the evaluation of a set of objective measures regarding their systems. The framework supports companies in discussing and analyzing potential benefits and drawbacks of the migration and re-architecting process.

* Corresponding author.

E-mail addresses: florian.auer@uibk.ac.at (F. Auer), valentina.lenarduzzi@lut.fi (V. Lenarduzzi), michael.felderer@uibk.ac.at (M. Felderer), davide.taibi@tuni.fi (D. Taibi).

¹ <https://gigaom.com/2011/10/12/419-the-biggest-thing-amazon-got-right-the-platform/>

² <http://nginx.com/blog/microservices-at-netflix-architectural-best-practices/>

³ www.infoq.com/presentations/linkedin-microservices-urn

<https://doi.org/10.1016/j.infsof.2021.106600>

Received 31 January 2020; Received in revised form 1 March 2021; Accepted 14 April 2021

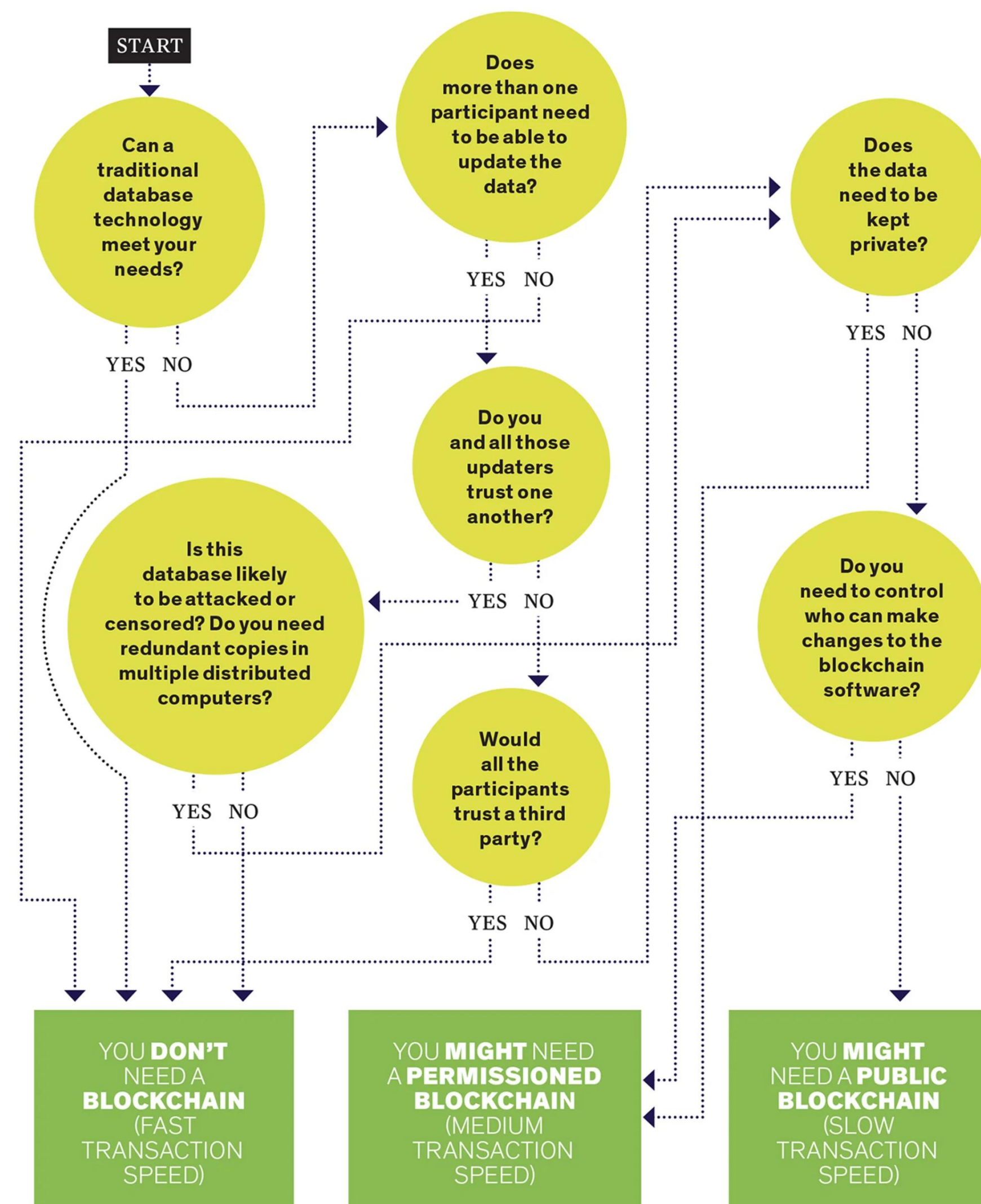
Available online 30 April 2021

0950-5849/© 2021 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

Technology Assessment

My favorite example (Blockchain)

M.E. Pech DO YOU NEED A BLOCKCHAIN?
IEEE Spectrum. September 2017.
<https://spectrum.ieee.org/do-you-need-a-blockchain>

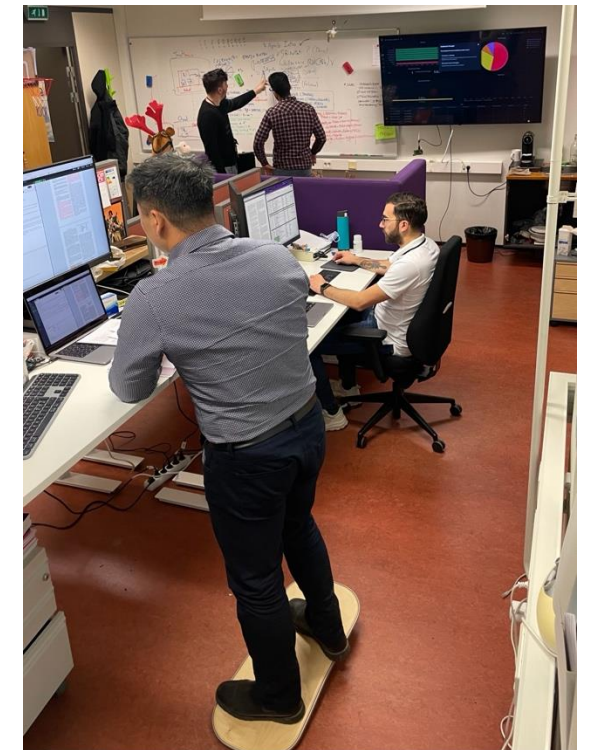


Future Goals

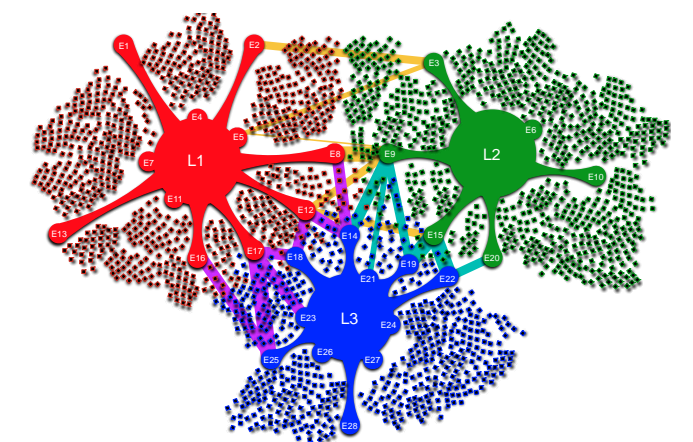
- Anomaly detection
- Energy analysis

- Orchestration optimization
 - Energy
 - Performance
 - QoS
 - ...
- Quantum computing into the cloud continuum

- Visualizations



BUSINESS
FINLAND

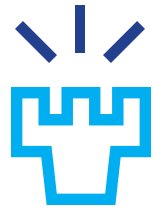


Edge Nodes Upgrade Optimization

NOKIA

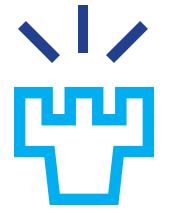


Conclusions



Morale of the story!

- AI-Based Systems are not only AI
- AI Devs and Ops need to be synchronized
- Orchestration is fundamental
- AI Deployment is expensive
 - AI-Ops is not enough!
- AI Operational costs might be unexpectedly high



Thanks!

Are you interested in our research?
Do you wish to work with us?

Get in touch!

davide.taibi@oulu.fi
<https://m3s-cloud.github.io>