AIoTwin

Twinning action for spreading excellence in Artificial Intelligence of Things

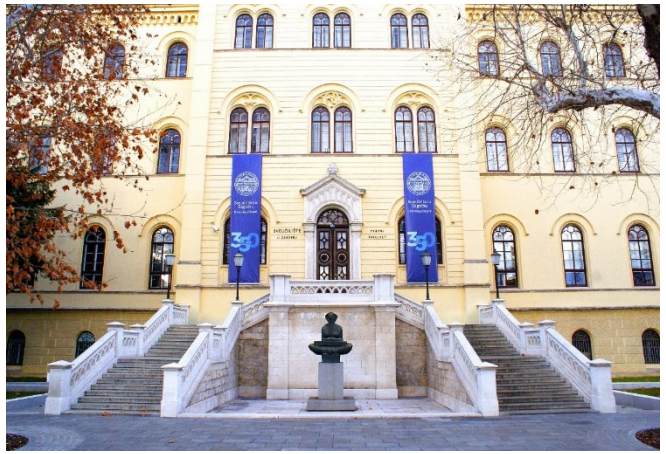
Artificial Intelligence of Things (AIoT): Bringing Intelligence into the Physical World

Ivana Podnar Žarko

University of Zagreb, Faculty of Electrical Engineering and Computing (UNIZG-FER)

Seminar – Digital Futures, August 29, 2024

Funded by the European Union



The University **of Zagreb**

is the biggest and the oldest Croatian public university

Faculty of Electrical Engineering and Computing

3500 **BACHELOR & MASTER STUDENTS**

500 **POSTGRADUATE STUDENTS**

730+ **EMPLOYEES**

300 **ACADEMIC STAFF**

AIoTwin

IoTLab @ UNIZG-FER

Internet of Things Laboratory
http://www.iot.fer.hr/

Established within Department of Telecommunications in 2015

Faculty members (3):
- Prof. Ivana Podnar Žarko, PhD (Head)
- Prof. Gordan Ježić, PhD
- Prof. Mario Kušek, PhD

PhD students (7):
- Ivan Čilić, MSc
- Mislav Has, MSc
- Ivan Kralj, MSc
- Dora Kreković, MSc
- Katarina Mandarić, MSc
- Ana Petra Jukić, MSc
- Katarina Vuknić, MSc

IoTLab: Selected research projects

- **AIoTwin: Twinning action for spreading excellence in Artificial Intelligence of Things,** funded the Horizon Europe program (01/2023 – 12/2025)
- **symbIoTe**: Symbiosis of smart objects across IoT environments, H2020 R&I project (2016 – 2018)
- **OpenIoT:** Open Source blueprint for large scale self-organizing cloud environments for IoT applications, FP7 project (2013 – 2015)
- **DIGIPHY**: XR Communication and Interaction Through a Dynamically Updated Digital Twin of a Smart Space, funded from Recovery and Resilience Facility (RRF), coordinated by UNIZG-FER, partners: Ericsson Nikola Tesla d.d. and Diversitas IT Sustavi d.o.o. (07/2024 – 06/2026)
- **IoT-field**: An Ecosystem of Networked Devices and Services for IoT Solutions Applied in Agriculture, funded from the European Structural and Investment Funds, (03/2020 – 11/2023)
- **DATACROSS:** research project of the Centre of Research Excellence for Data Science and Advanced Cooperative Systems (2017 – 2022)
- **IoT4us**: Human-centric smart services in interoperable and decentralised IoT environments (IoT4us), research project No. 1986 funded by the Croatian Science Foundation, (01/2020 - 04/2024)
- **Communication Challenges in Machine-to-Machine Communication**, UNIZG-FER & Ericsson Nikola Tesla, (2011– now)

About the AIoTwin project

Striving for research excellence for artificial intelligence in IoT

- “The EU-funded AIoTwin project offers a solution by introducing a cooperative effort between leading European researchers, universities and institutions. Through this effort, they plan to increase the research excellence of UNIZG-FER and other European universities in the field of IoT, while also educating new researchers.”

  from: https://cordis.europa.eu/project/id/101079214

- January 2023 – December 2025
- Grant agreement ID: 101079214
- Programme: HORIZON.4.1 - Widening participation and spreading excellence
- Topic: HORIZON-WIDERA-2021-ACCESS-03-01
- Type of action: HORIZON Coordination and Support Actions

AloTwin consortium

Coordinator

SVEUCILISTE U ZAGREBU FAKULTET ELEKTROTEHNIKE I RACUNARSTVA

Address

Unska 3
10000 Zagreb
Croatia

Net EU contribution
€ 544 500,00

Other funding
€ 0,00

Participants (3)

RISE RESEARCH INSTITUTES OF SWEDEN AB
Sweden

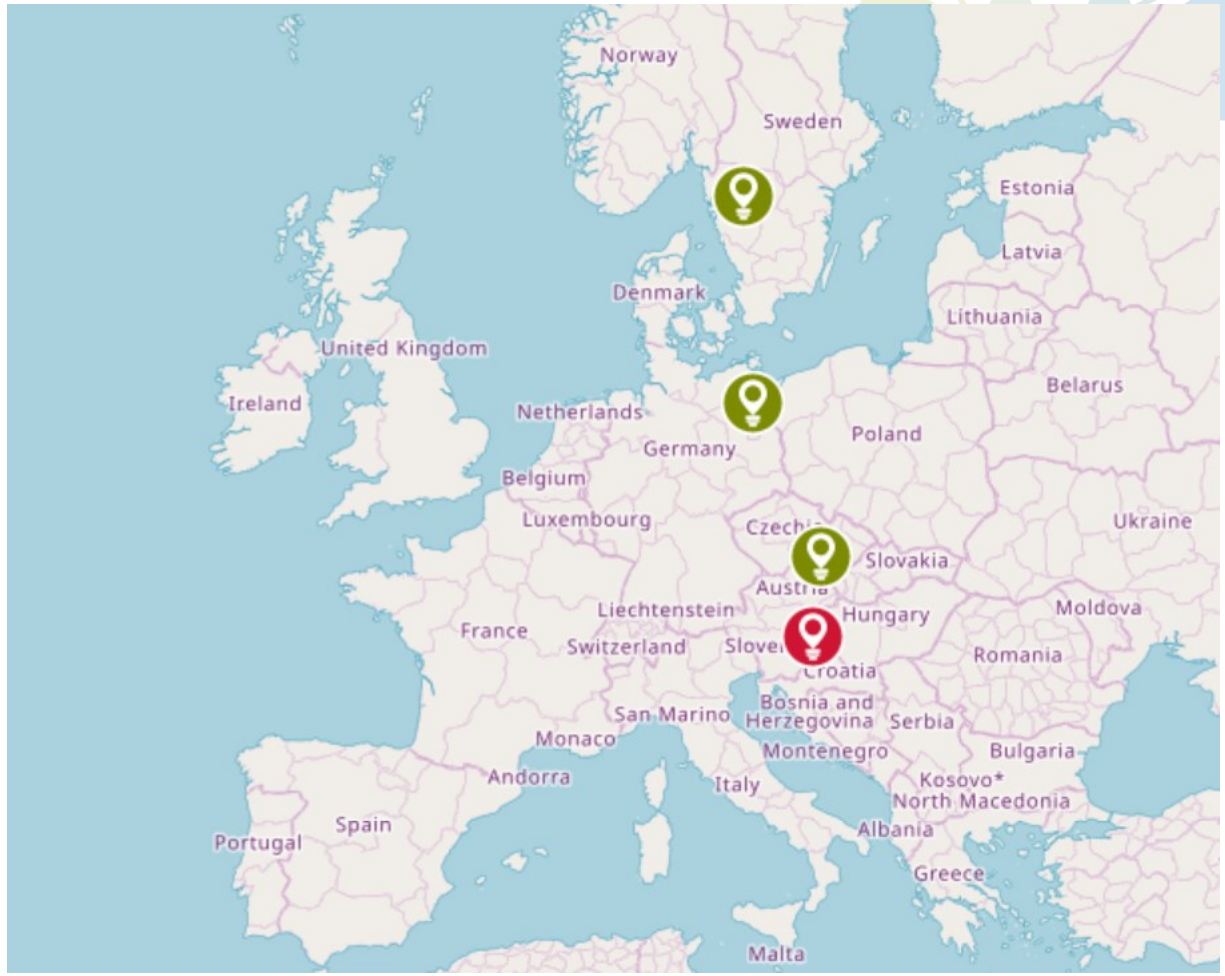
Net EU contribution
€ 335 062,50

TECHNISCHE UNIVERSITAET WIEN
Austria

Net EU contribution
€ 289 312,50

TECHNISCHE UNIVERSITAT BERLIN
Germany

Net EU contribution
€ 330 750,00



Strategic goal of the AIoTwin project

- to significantly strengthen **the scientific excellence and innovation capacity of UNIZG-FER** in the area of the Internet of Things through knowledge transfer and strategic networking activities between UNIZG-FER researchers and world-class scientists from leading EU institutions.
- to create a stimulating research environment at UNIZG-FER and increase scientific quality to attract new national and international talents
- to create opportunities for future career developments of young researchers
- involves a **research component** – development of AIoTwin orchestration middleware

Talk outline

- Introduction: AIoT
- Orchestration at the edge
- Orchestration Middleware for AIoT
- Adaptive orchestration of federated learning workflows
- QoS-aware data routing in the IoT-edge-cloud continuum
- Future work

1st AIoTwin Summer School, Šibenik, Sept 2023

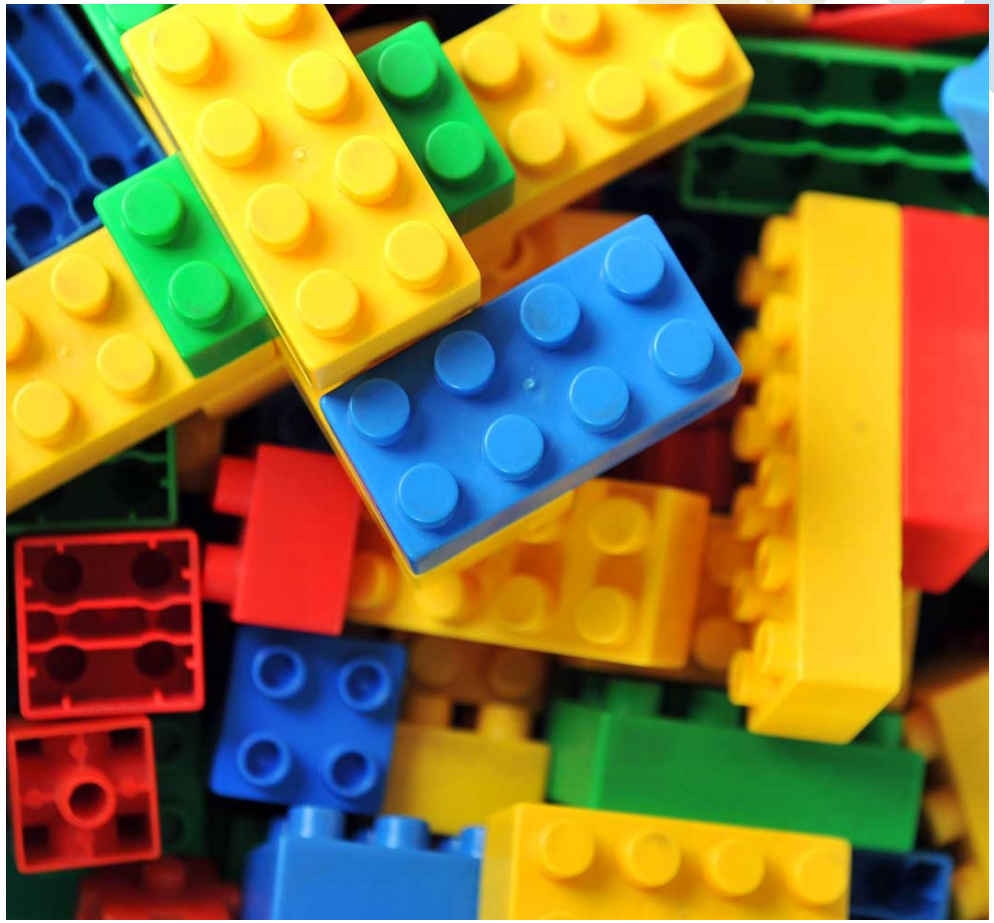
AIoTwin

Introduction

Artificial Intelligence of Things, AIoT

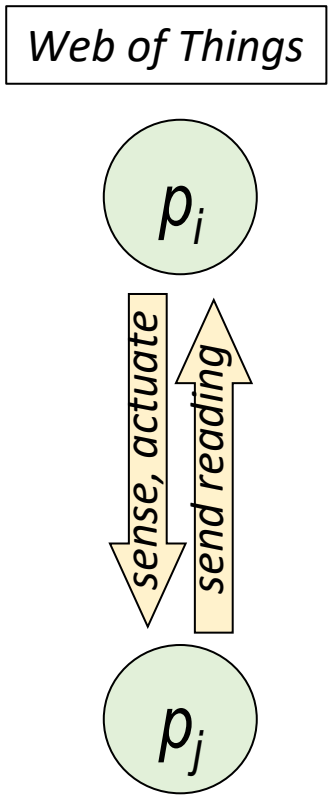
IoT: the state of the art

- No. of connected IoT devices worldwide in 2023: 18 billion, prediction for 2030 – 30 billion (source: Statista)
- Cloud-based IoT platforms are still predominant
- A set of mature IoT protocols
- The lack of interoperable ecosystems across platforms and domains
- Existing infrastructure is underutilized
- Users are faced with a multitude of applications within a single domain at various locations

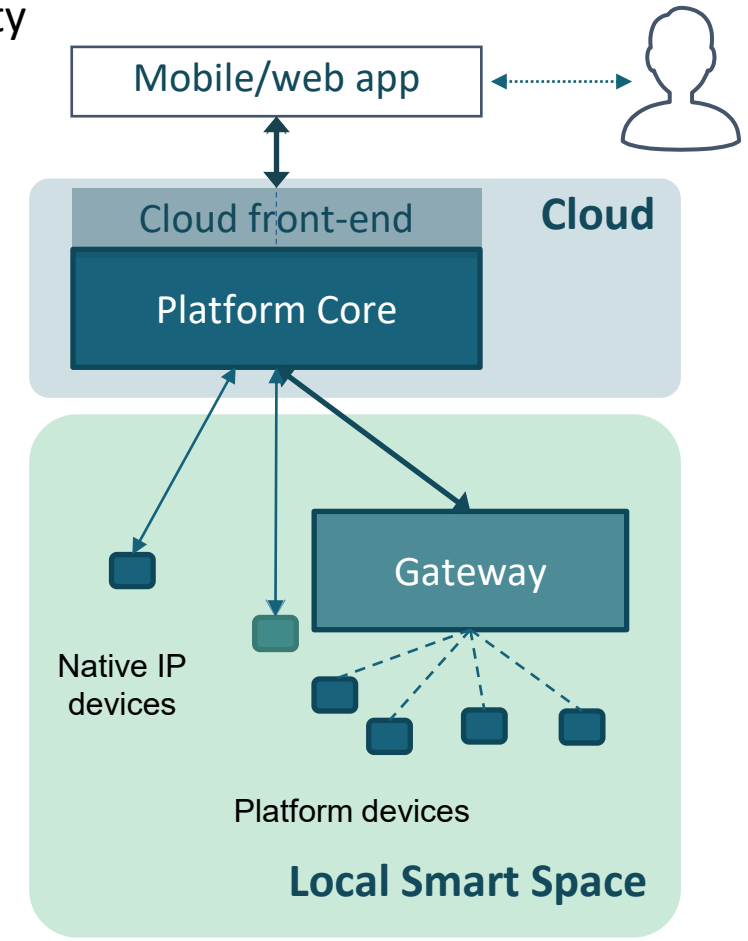


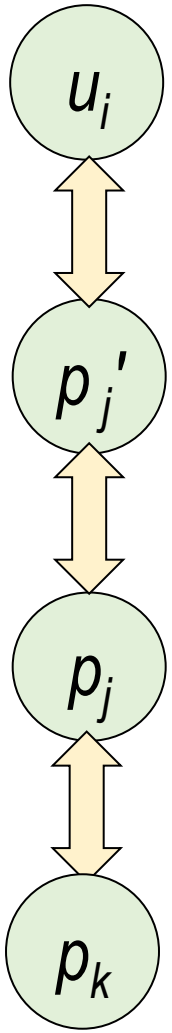
IoT vision vs. reality

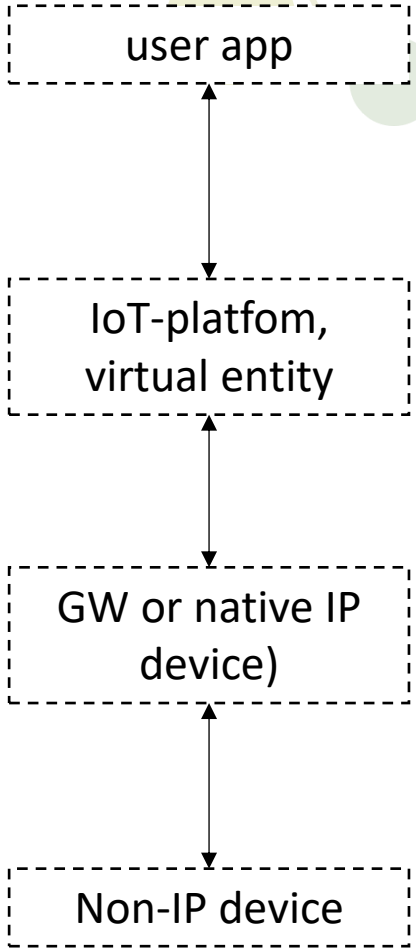
vision



reality



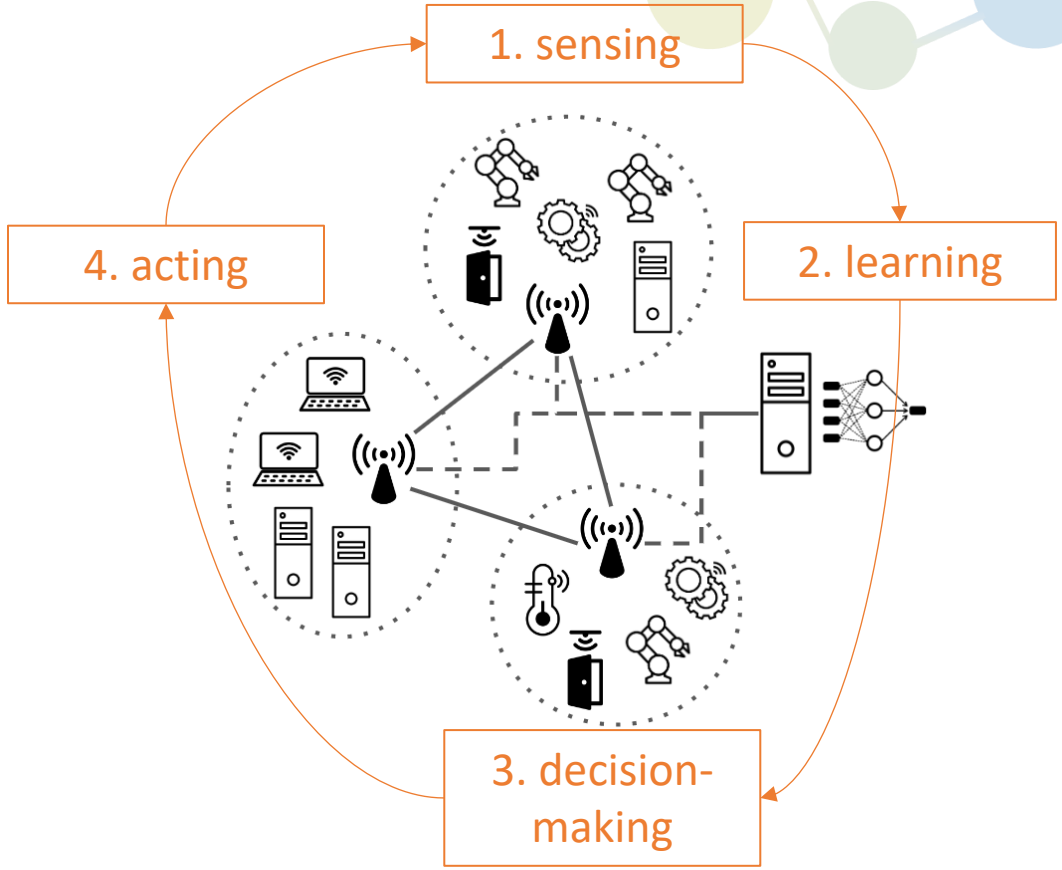




AIoTwin

Artificial Intelligence of Things, AIoT

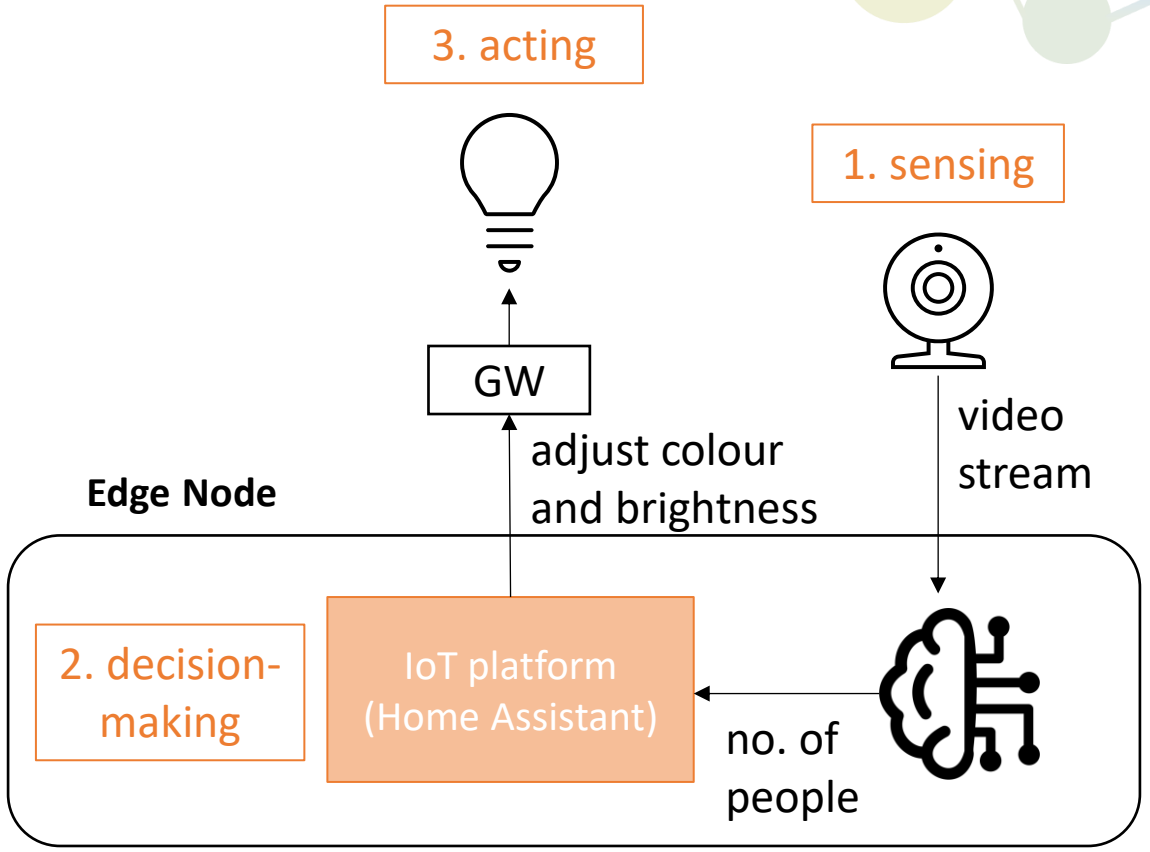
- IoT evolution
- Brings AI into smart physical spaces
- Boundaries between physical and digital world disappear
- Smart physical spaces generate large amounts of streaming data (**sensing**) for **learning**, creation of new AI models
- AI models are increasingly used in smart physical spaces, **inference** facilitates **decision-making**
- **Actuation** enables machines to **act**



AIoT: a simple example application

Occupancy Detection and Smart Lighting

- smart home environment that automatically adjusts lighting (e.g. color and brightness) based on the number of people present
- Hardware: an edge device, USB webcam and smart LED bulb
- Edge device hosts
  1) a pretrained ML model which analyses a video stream for people counting; the number of people detected is sent to 2)
  2) an IoT platform for integration and control of smart devices: a smart LED bulb for color and brightness control



AIoT challenges

- distributed and heterogeneous environments with limited resources in terms of available processing power and energy
  - requires efficient orchestration of services in the computing continuum, algorithms adapted to the **distributed IoT-edge-cloud environment**
- real-time data processing
  - ML algorithms need to be adapted to **online learning**
  - data streams from IoT devices are often incomplete and prone to errors
- strict privacy and security requirements
  - protection of sensitive user data
  - ensuring device integrity and security of the physical environment

AIoTwin

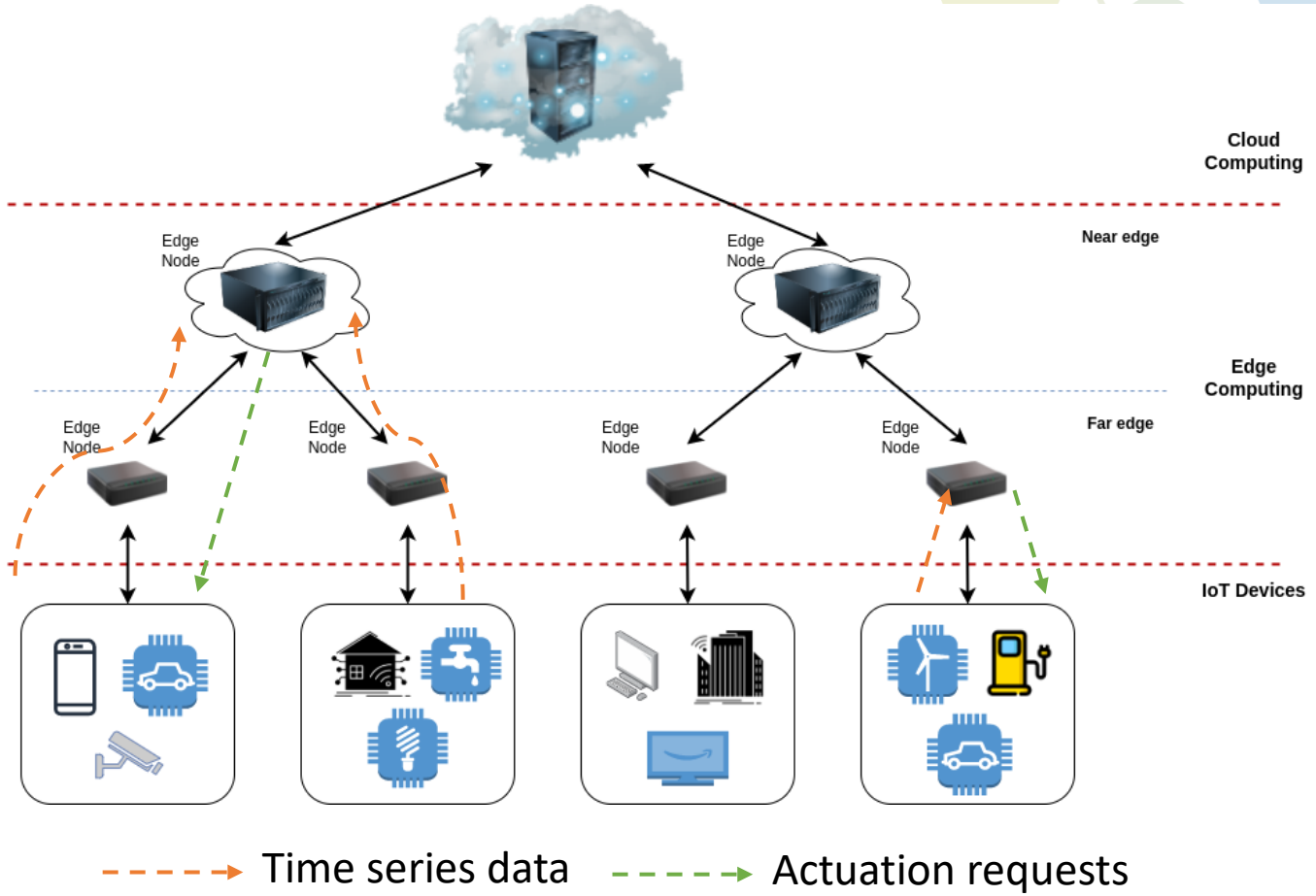
Orchestration at the edge

IoT-edge-cloud continuum

Orchestration tools

Edge computing

- ***IoT-edge-cloud continuum***
- Brings cloud services closer to IoT devices
  - reallocation of services from the cloud to computing resources in the vicinity of the IoT devices and at the network edge
- Characteristics
  - hierarchical multi-tier infrastructure, heterogeneous and dynamic environment
  - near-edge vs. far-edge devices



IoT-edge-cloud continuum

Benefits

- reduced processing latency and overall network traffic
- improved reactivity
- increased scalability
- resilience to failures (both node and link)
- improved privacy and security (data remains within the same administrative domain)
- reduced operating costs

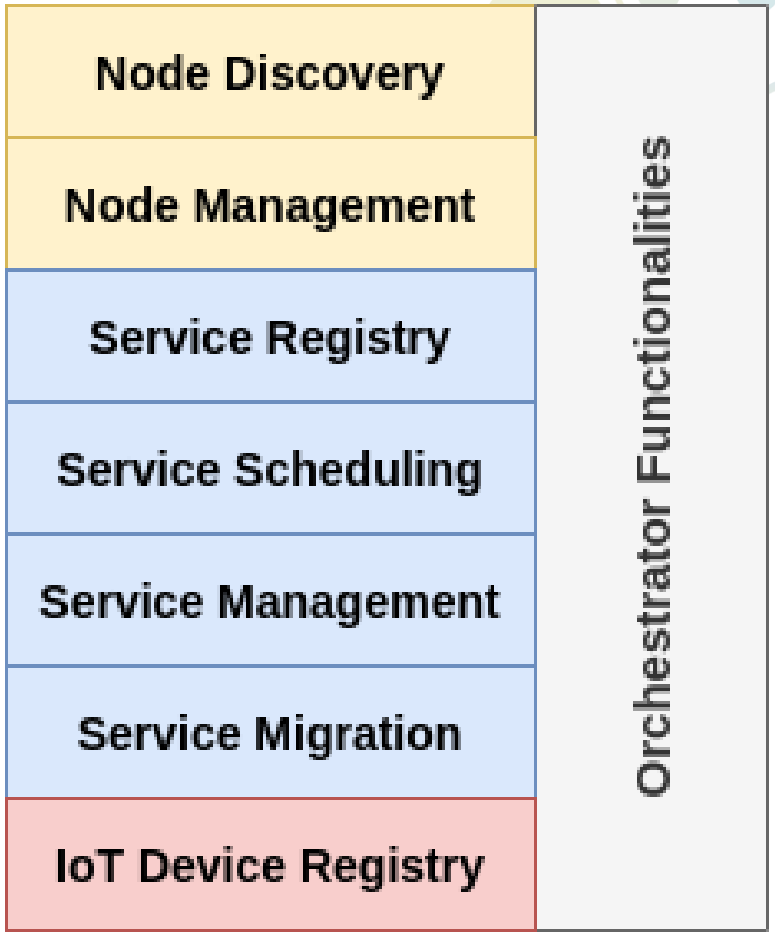
Drawbacks

- heterogeneity increases system complexity
- unstable nodes with limited resources
- dynamic system with frequent network changes
- increased capital costs for the edge infrastructure

Edge orchestration

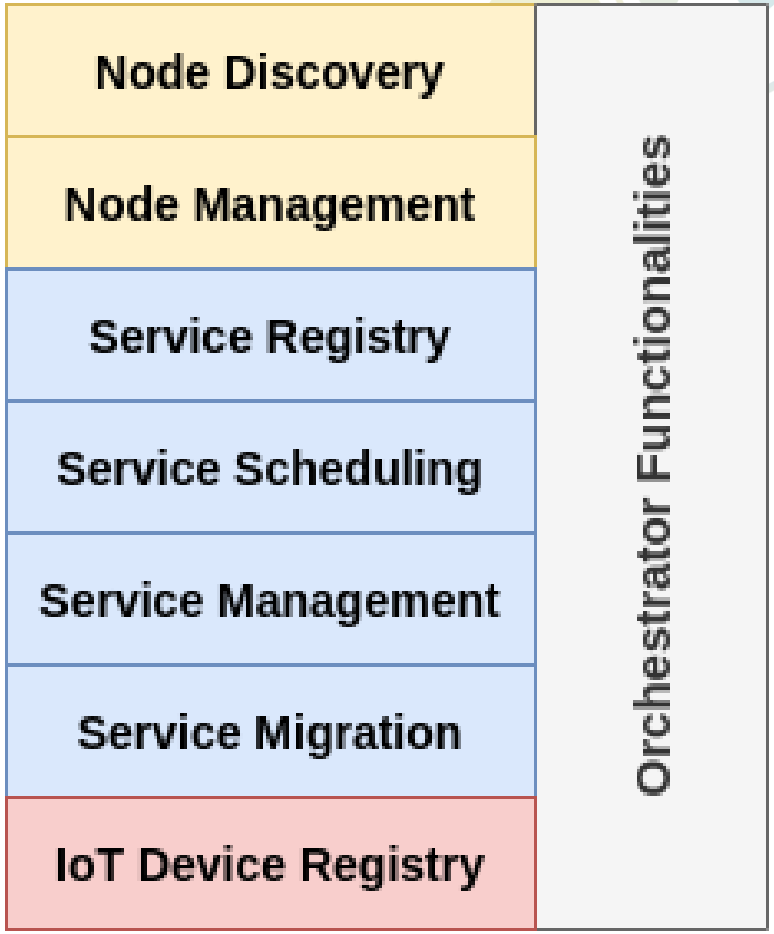
- Services running on edge nodes have to be orchestrated to ensure their high availability
  - technologies: microservices, containers, container orchestration tools
- Service orchestration is needed to
  - **schedule**
  - **deploy**
  - **manage**

  } services in a distributed edge computing environment
- Main goal:
  - continuously ensure the required QoS level to IoT devices and application-level services exposed to end users



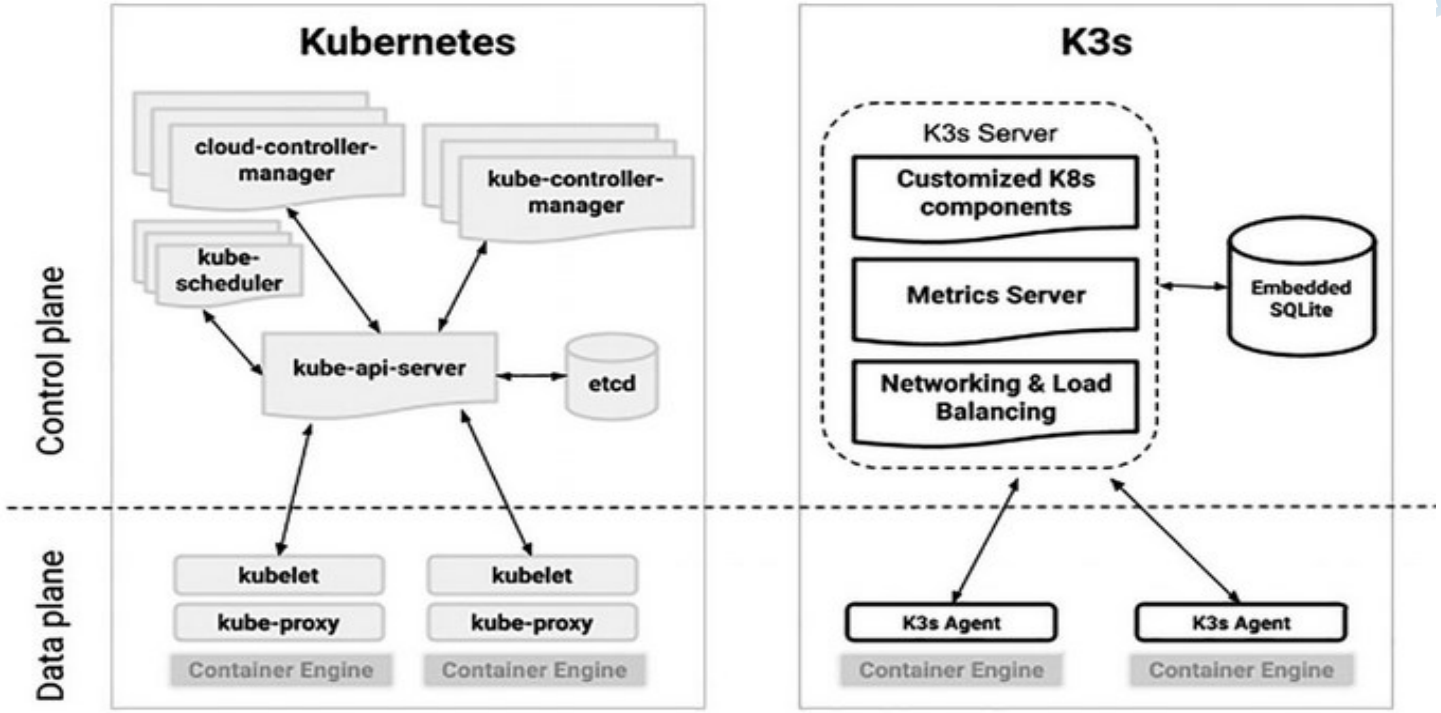
Edge orchestration architecture

- Essential building blocks
  - IoT Device (limited resources) – data source and/or destination
  - Edge Node (runs containerized edge services) – “heterogeneous infrastructure”
  - Edge Service (autonomous, stateless, and portable) – deploy, start, stop, replicate, migrate
  - **Orchestrator** – centralized component



Orchestration tools

- Kubernetes: cloud orchestration
- K3s: lightweight Kubernetes
- Centralized arhitecture: orchestrator and worker nodes



Čilić, Ivan; Krivić, Petar; Podnar Žarko, Ivana; Kušek, Mario. (2023) **Performance Evaluation of Container Orchestration Tools in Edge Computing Environments**. *Sensors* **2023**, *23*, 4008. https://doi.org/10.3390/s2308400

Orchestration tools: comparison

Parameter	**Kubernetes**	**K3s**	**KubeEdge**	**ioFog**
Documentation Quality	**High**	Medium	Medium	Low
Deployment Complexity	High	**Low**	Medium	Medium
Private Networks	No	**Yes**	Yes	Yes
Heterogeneity	Yes	Yes	Yes	Yes
Resource-Constrained	Yes	Yes	Yes	Yes
Memory Footprint (MB)	~50	~50	~40	~240

Time (s)	**Kubernetes**	**K3s**	**KubeEdge**	**ioFog**
Total Startup Time	1.799	2.798	2.858	34.537
Total Migration Time	1.838	2.782	2.781	23.244
Startup Time Overhead	0.504	1.502	1.5622	33.326
Migration Time Overhead	0.542	1.486	1.485	22.032

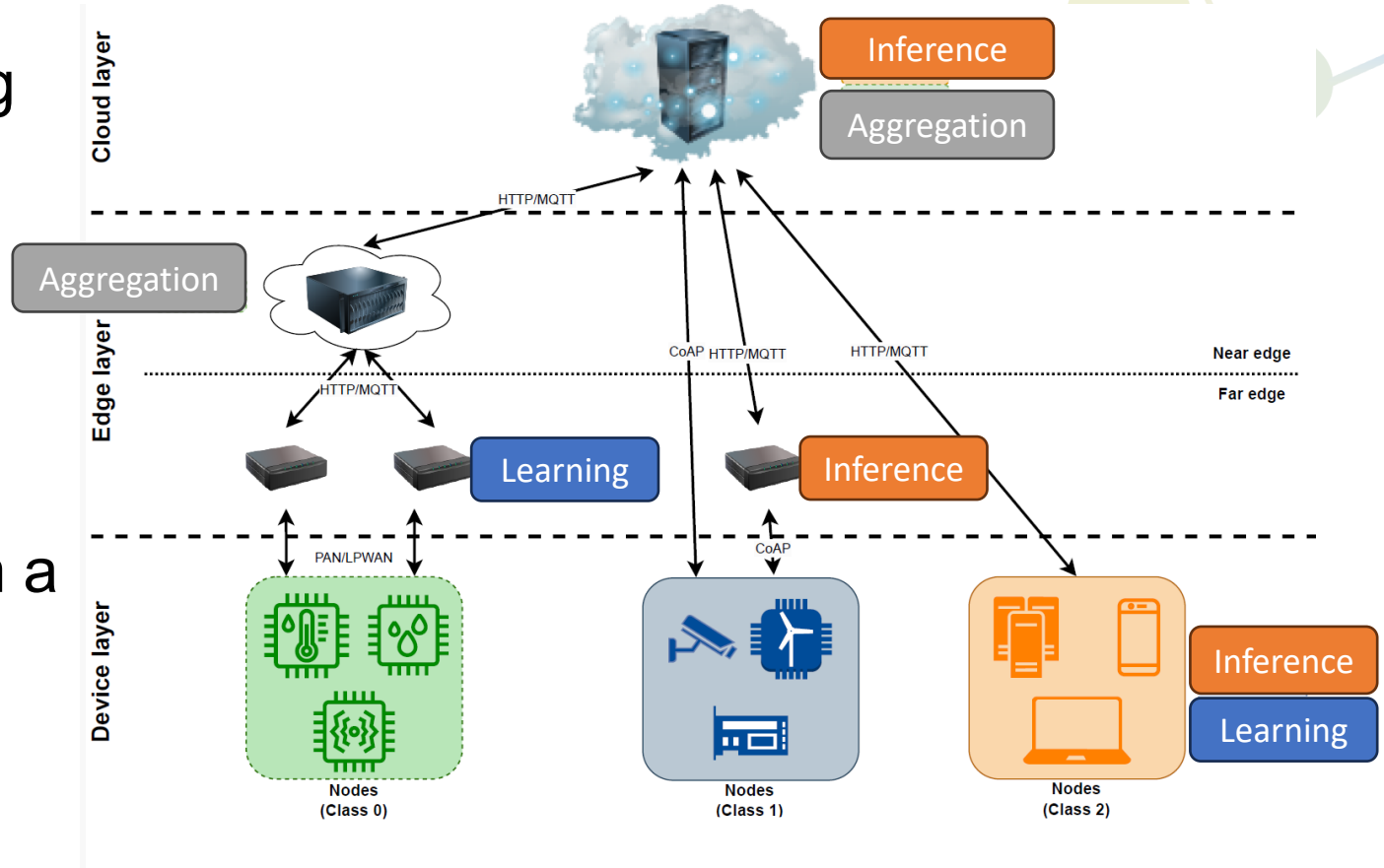
Orchestration Middleware for AIoT

Requirements and Architecture

AIoTwin Deliverable 1.1 - Report on Use Cases, Requirements, and Architecture (1). 31 Dec 2023 **PDF**

What is so special about orchestration middleware for AIoT?

- ML workflows/pipelines: learning vs. inference
- Placement of ML models
- Federated learning and aggregation
- Which node should be used for inference for a data stream from a particular IoT device?



Specified requirements

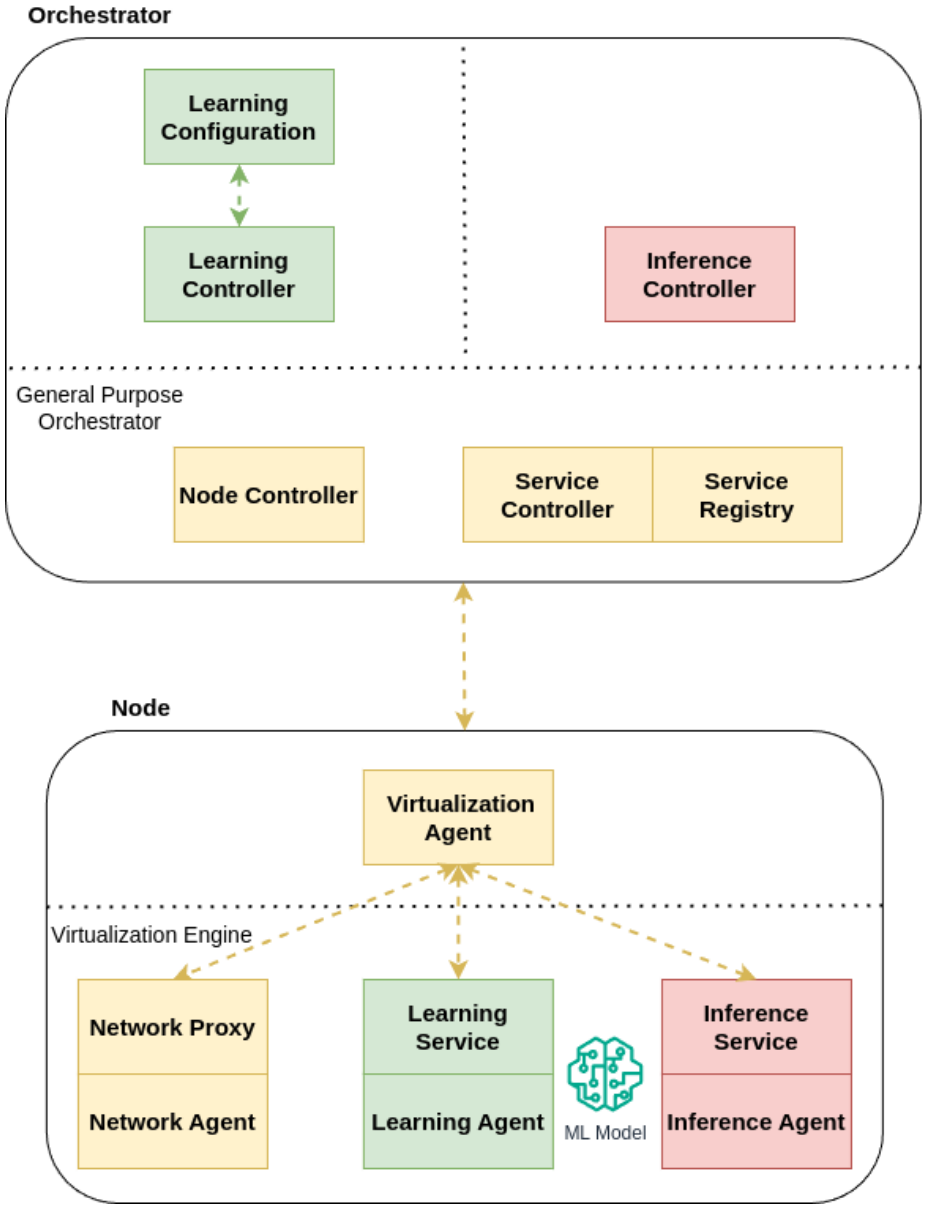
No	Description
1	efficiently manage and monitor resources on each node
2	collect the distribution of data available on node for ML
3	collect the information on the underlying network connecting nodes in the continuum
4	deploy and manage services across the continuum
5	run a configuration model to output configuration of an ML pipeline
6	deploy ML components based on a learning configuration
7	monitor learning performance

No	Description
8	reconfigure the learning pipeline if a better learning performance can be achieved
9	deploy and manage inference components.
10	monitor inference accuracy
11	monitor inference service performance
12	maintain inference performance and dynamically adapt to changes in the system
13	maintain a desired QoS level for clients using the inference services

FL-related Inference-related

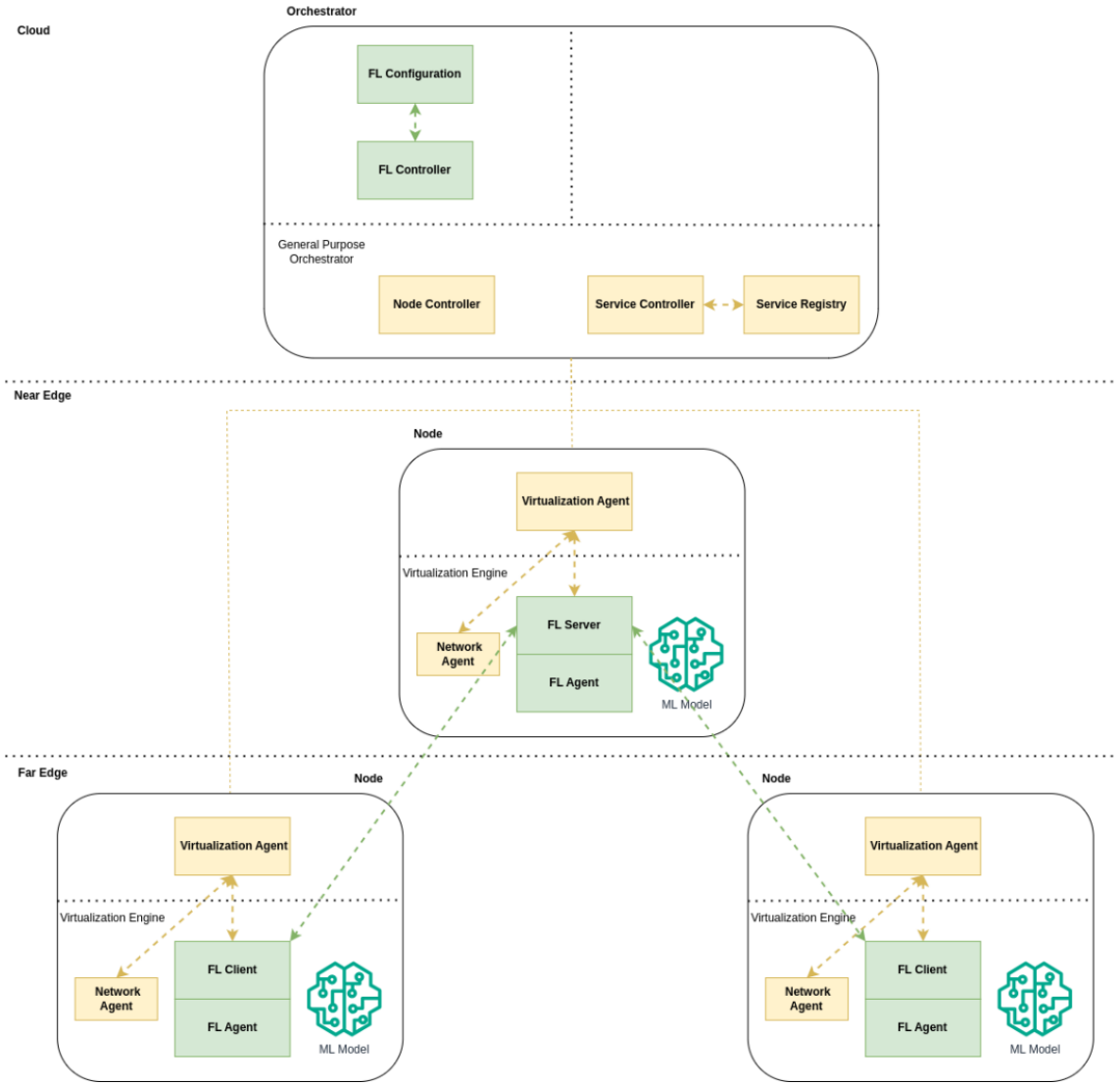
General Architecture for Orchestration of ML Pipelines

- Orchestrator
  - central entity, deployed in the cloud for high availability
  - general purpose orchestration, learning- and inference-specific components
- Node
  - runs ML pipeline services in a Docker container or WebAssembly



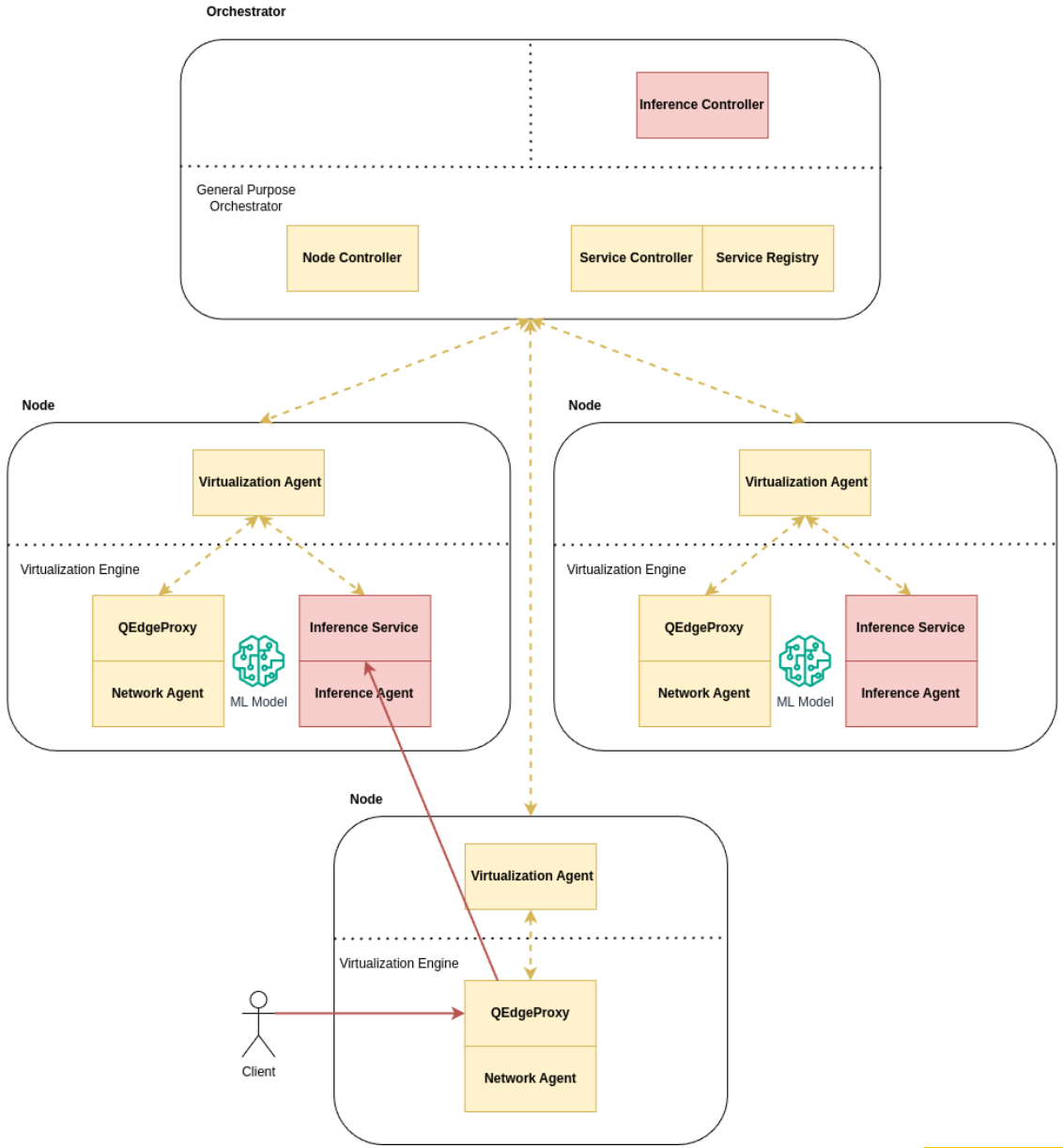
Adaptive orchestration of **FL pipelines**: the architecture

- FL Clients and Aggregators
- Nodes participating in training may have different (i) hardware specifications, (ii) network characteristics, or (iii) data distributions.
- An **adaptive orchestration mechanism** is needed to deploy the entities of the FL pipeline, monitor the execution of the pipeline, and perform reconfiguration when needed.



QoS-aware load balancing for **inference**: the architecture

- **QEdgeProxy**, a distributed QoS-aware load balancer
- QEdgeProxy serves as a „QoS agent” for IoT clients within the computing continuum, and acts as an external routing component, i.e., an intermediary between IoT clients and IoT services across the computing continuum.
- Adapts to changes in the continuum to meet QoS requirements



AIoTwin

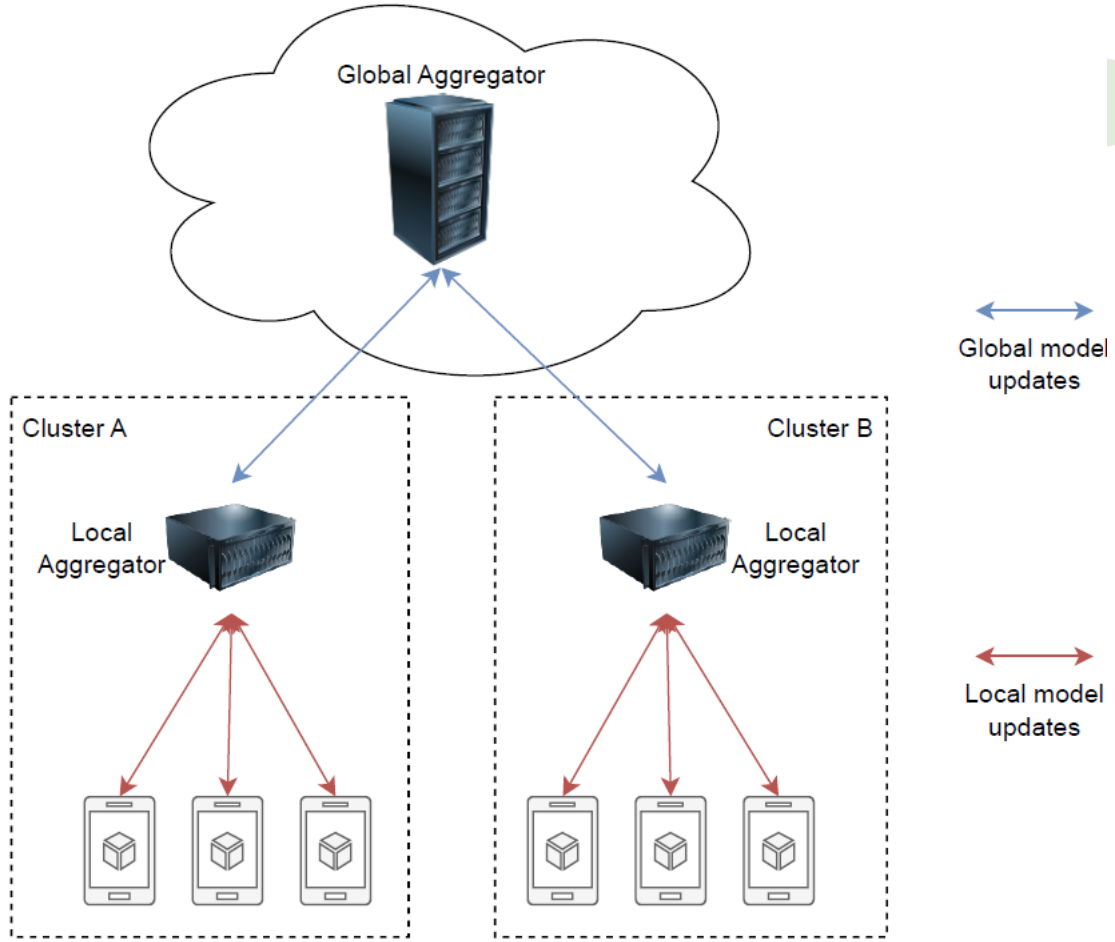
Orchestration Middleware

Adaptive orchestration of federated learning workflows

Ivan Čilić, Anna Lackinger, Alireza Furutanpey, Ilir Murturi, Pantelis Frangoudis, Ivana Podnar Žarko, Schahram Dustdar. **Adaptive Orchestration of Federated Learning Workflows**. *In preparation for journal submission*. Sept 2024.

Hierarchical Federated Learning

- FL challenges
  - Hardware heterogeneity -> stragglers
  - Unstable and bandwidth-limited network
  - Unbalanced data distribution (non-IID)
  - Privacy requirement
- Hierarchical FL to reduce **communication costs**



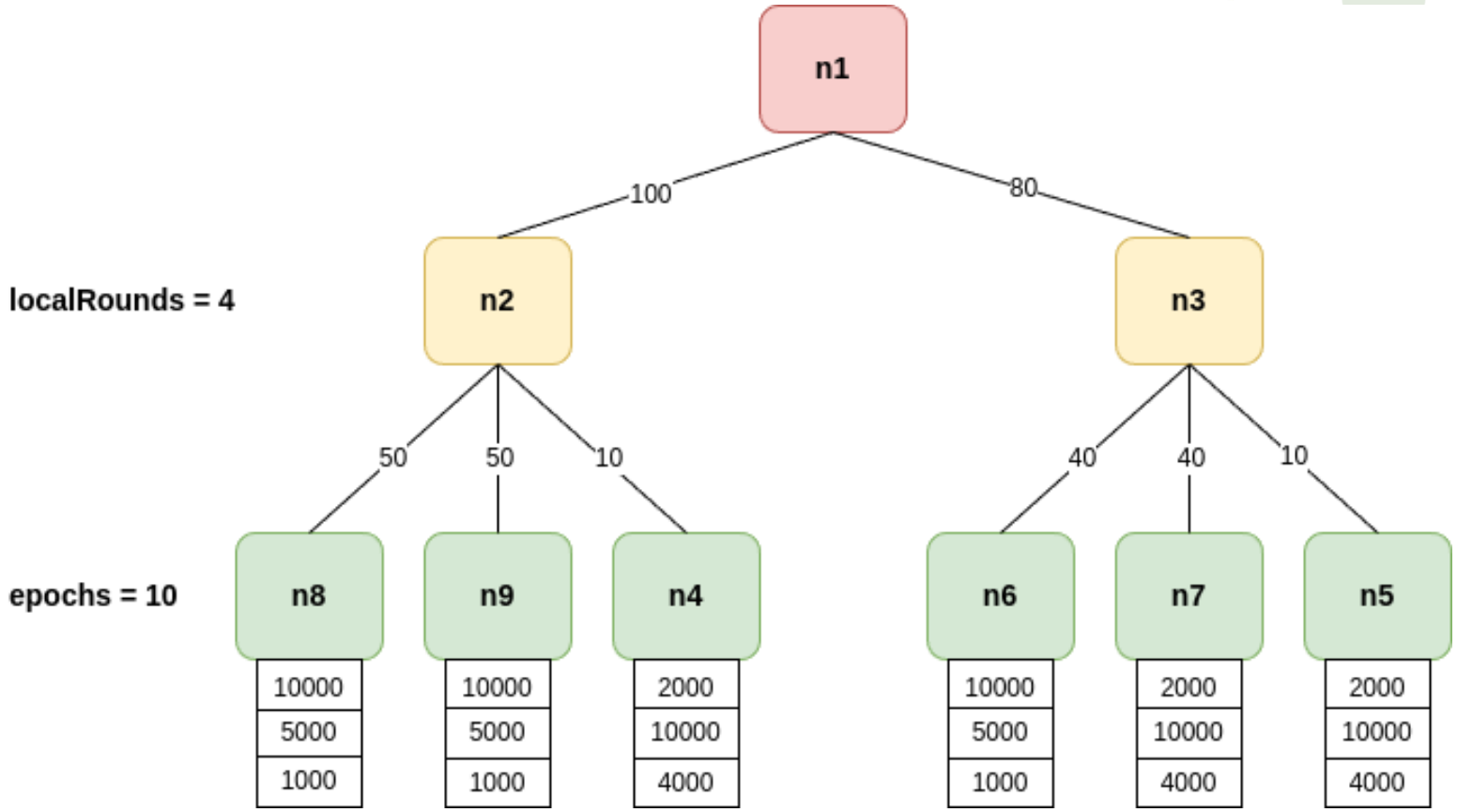
Hierarchical FL configuration: an example

How should we organize clients into clusters?

- Data distribution
- Communication costs

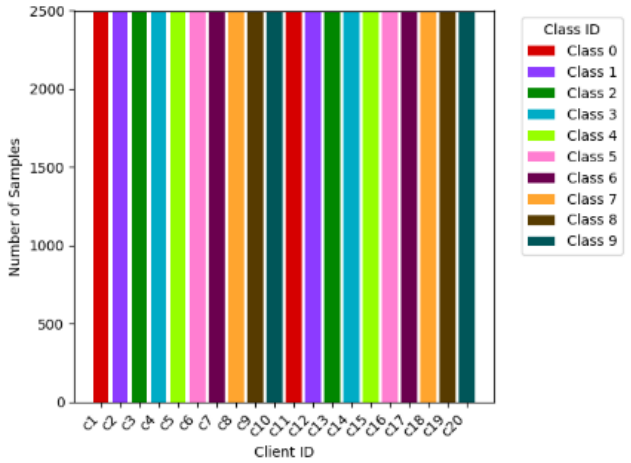
FL configuration?

- aggregation algorithm
- aggregation frequency

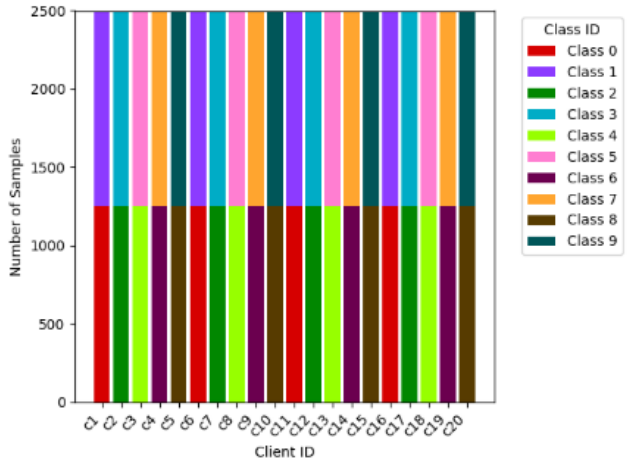


Hierarchical FL experiments

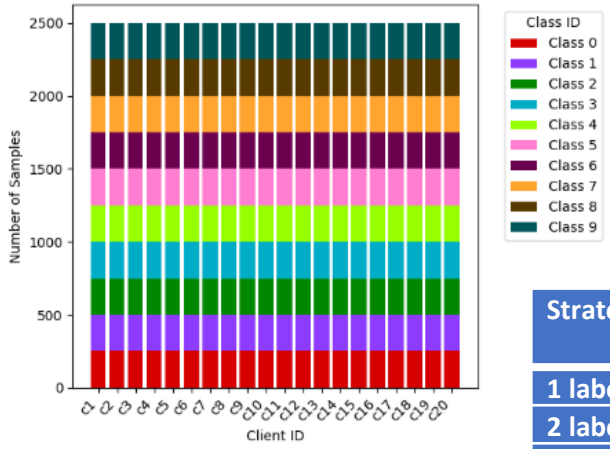
- What influences the training performance?
  - different number of clients and data samples per client, different number of class labels per client, position of clients within clusters, the number of epochs and local aggregation rounds
- CNN for classification of photos, CIFAR-10
- An example setup: possible data distribution strategies when 20 clients share the same number of data samples



Strategy with 1 label per client



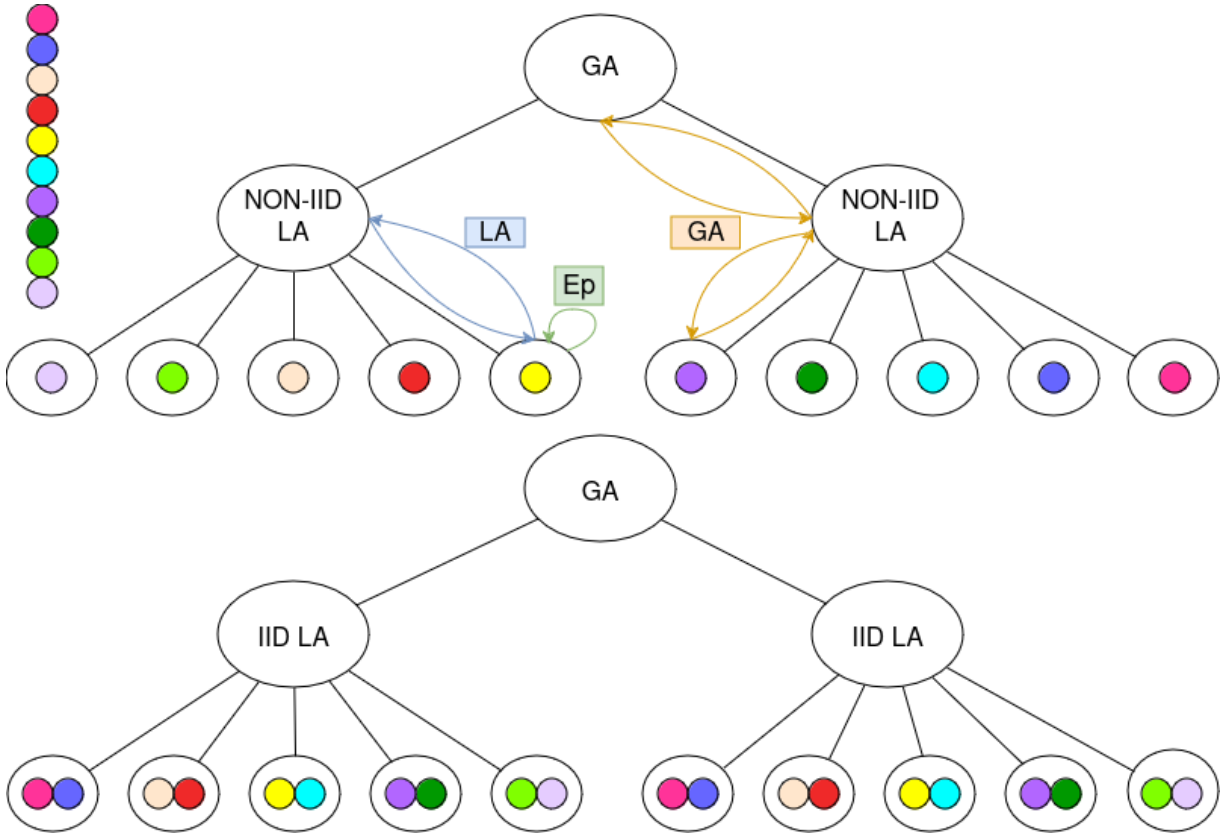
2 labels per client



10 labels per client

Strategy	Accuracy after 500 epochs
1 label/client	22.71
2 labels/client	47.92
10 labels/client	**69.39**

Hierarchical FL experiments



IID cluster	Hierarchi-cal	Ep	LA	Accuracy after 500 epochs
No	No	10	-	22.73
No	Yes	10	1	22.59
No	**Yes**	**5**	**2**	**24.33**
No	Yes	2	5	22.91
Yes	No	10	-	22.71
Yes	Yes	5	2	22.73
Yes	Yes	2	5	21.37

Adaptive Orchestration of Federated Learning Workflows

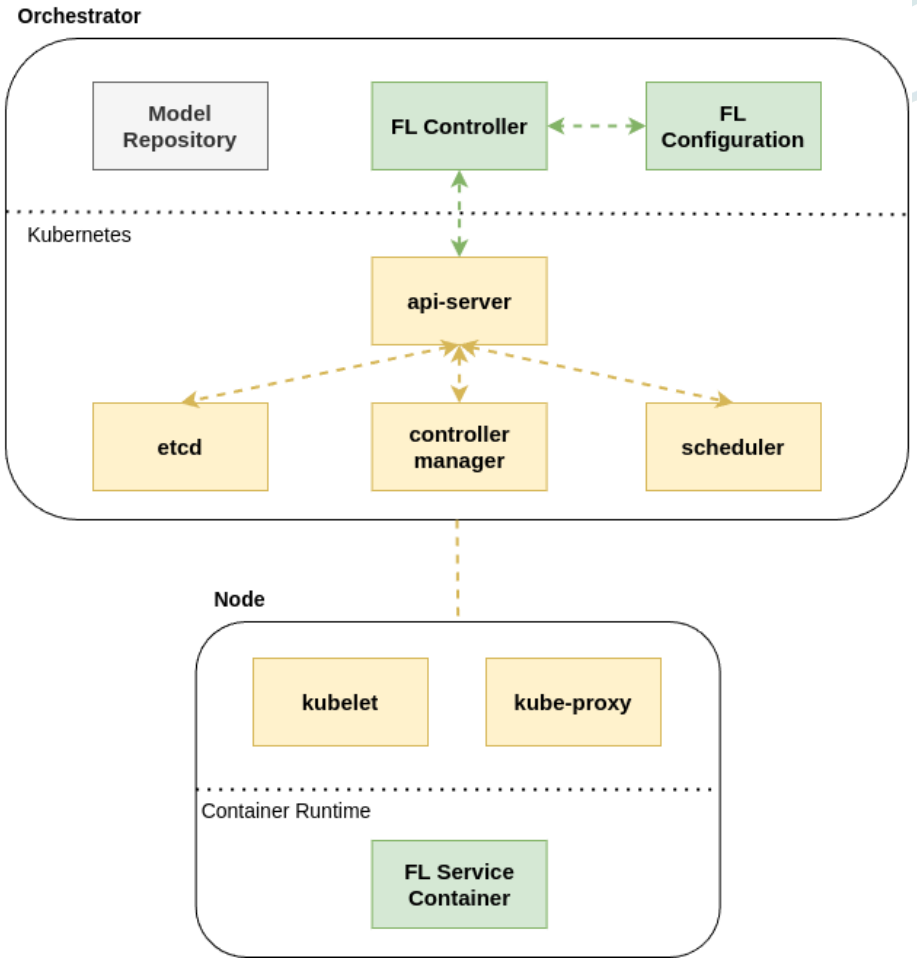
- The continuum is dynamic -> adaptive orchestration
- challenges of hierarchical FL:
  - configuration of an FL pipeline depends on the model to be trained, the underlying infrastructure, and the data distribution
  - dynamicity of the edge environment influences the configuration of an FL pipeline (nodes come and go)

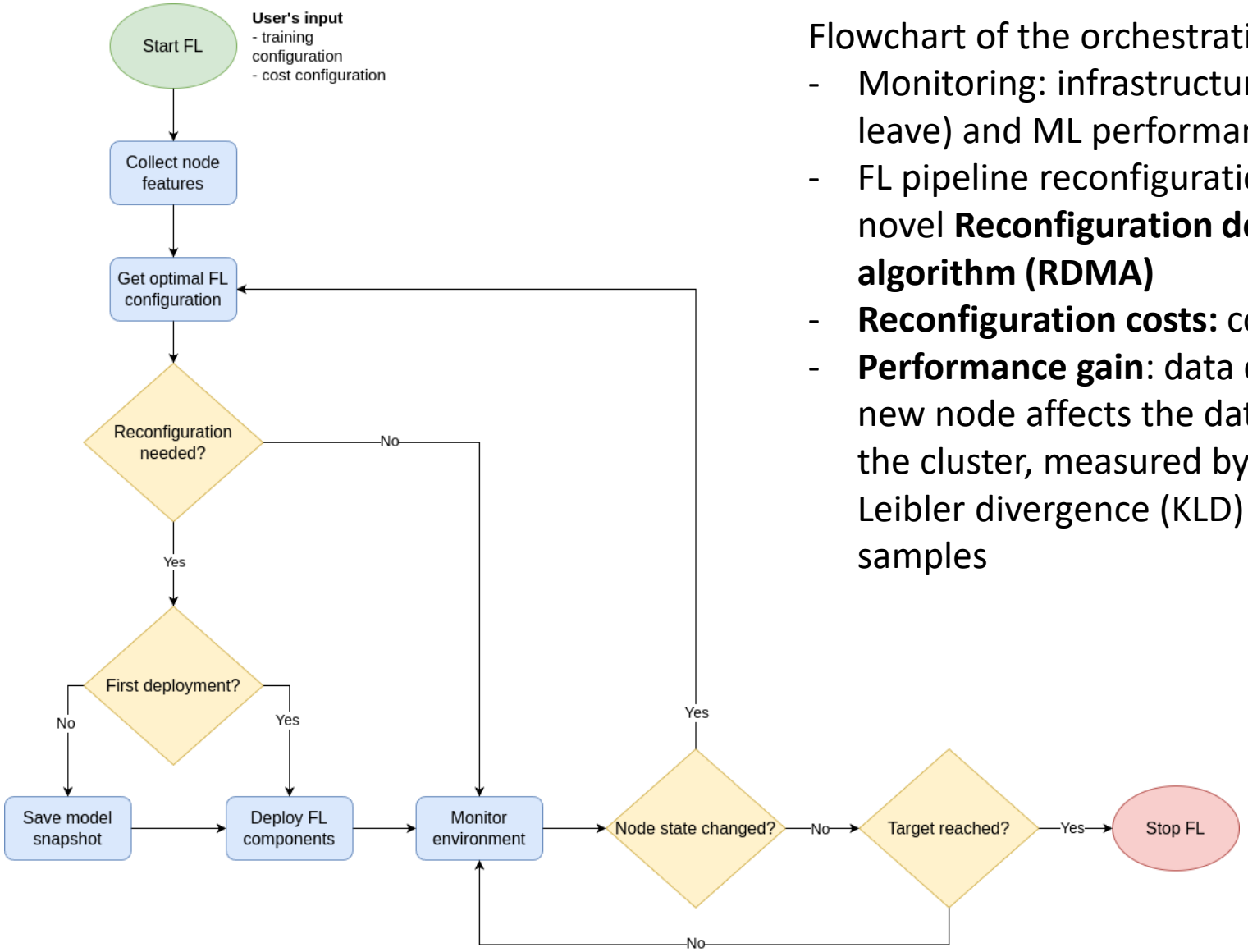
ORCHESTRATOR FUNCTIONALITIES COMPARED TO GENERAL-PURPOSE ORCHESTRATOR (GPO).

Functionality	GPO
Node Discovery	Yes
Collecting Node Resource Information	Yes
Collecting Node Data Distribution	No
Collecting Communication Cost Matrix	No
Running FL Configuration Model	No
Service Registry	Yes
Service Deployment	Yes
Node Monitoring	Yes
FL Performance Monitoring	No

Implementation: Adaptive FL Orchestration on Top of Kubernetes

- FL Orchestrator
  - implemented in Golang
  - built on top of Kubernetes
  - connects to Kubernetes API to deploy services and obtain node information
- FL Service
  - Client, local aggregator or global aggregator
  - Implemented in Python
  - Extends Flower framework for FL
- Evaluation
  - K3s cluster





Flowchart of the orchestration workflow

- Monitoring: infrastructure (node join and leave) and ML performance (accuracy, loss)
- FL pipeline reconfiguration: we propose a novel **Reconfiguration decision-making algorithm (RDMA)**
- **Reconfiguration costs:** communication
- **Performance gain**: data distribution of the new node affects the data distribution of the cluster, measured by the Kullback-Leibler divergence (KLD) and the number of samples

AIoTwin

Orchestration Middleware

QoS-aware data routing in the IoT-edge-cloud continuum

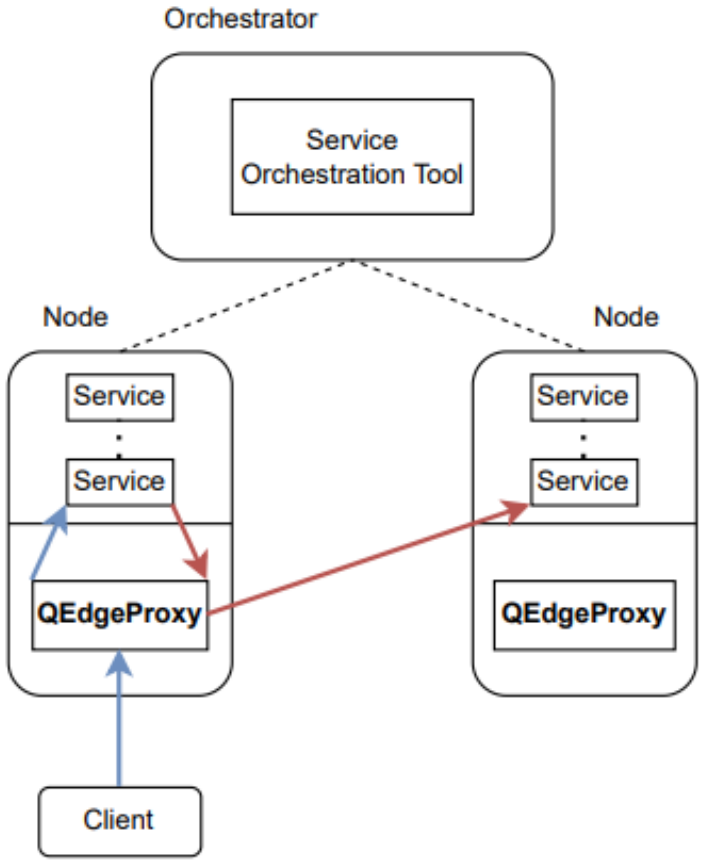
I. Čilić, V. Jukanović, I.Podnar Žarko, P. Frangoudis, S. Dustdar. QEdgeProxy: QoS-Aware Load Balancing for IoT Services in the Computing Continuum, IEEE International Conference on Edge Computing & Communications (EDGE 2024), 2024.
https://arxiv.org/abs/2405.10788

Motivation

- Each edge service is associated with a set of QoS requirements (SLOs), such as:
  - latency, throughput, availability, security
- Significant body of works on QoS-aware scheduling and placement
- Research question: *How can we continuously deliver IoT data from client devices to service instances deployed across the continuum while meeting the QoS requirements for data processing?*
- Related work
  - identify instances that optimize QoS or perform trade-offs between QoS and load balancing

QEdgeProxy

- QoS-aware load balancer designed as a QoS agent for IoT clients
  - Identifies service instances across the continuum that can fulfill QoS requirements for a given service (**QoS pool – maintained continuously**)
  - Forwards a request to a particular service instance while performing load balancing

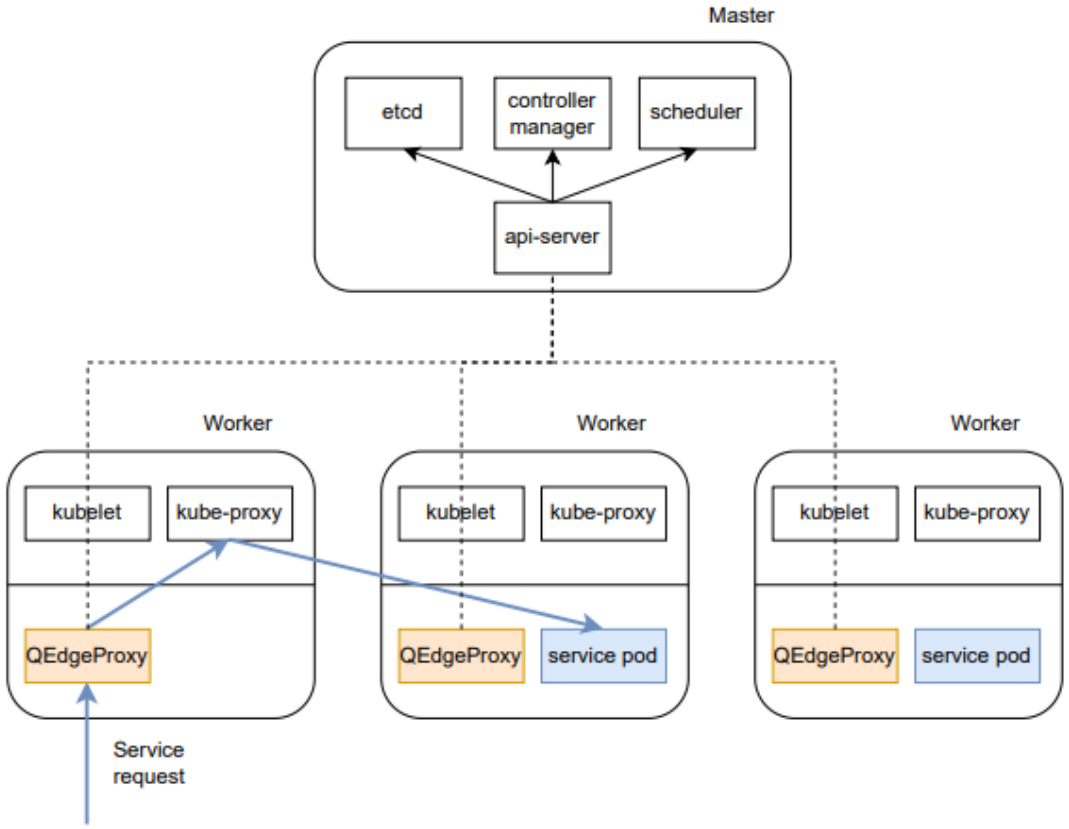


QEdgeProxy

- QoS pool
  - Multiple service replicas -> instances
  - Each QEdgeProxy predicts QoS of a specific instance based on monitoring and approximations
    - Based on previous instance and node behavior, or network characteristics
  - QoS pool is a subset of service instances that are likely to meet the QoS requirements
- Events that trigger QoS pool updates
  - Initial QoS pool creation
  - Instance state updates
  - QoS measurement updates
  - Environment monitoring and reactive updates

Implementation: QEdgeProxy within Kubernetes

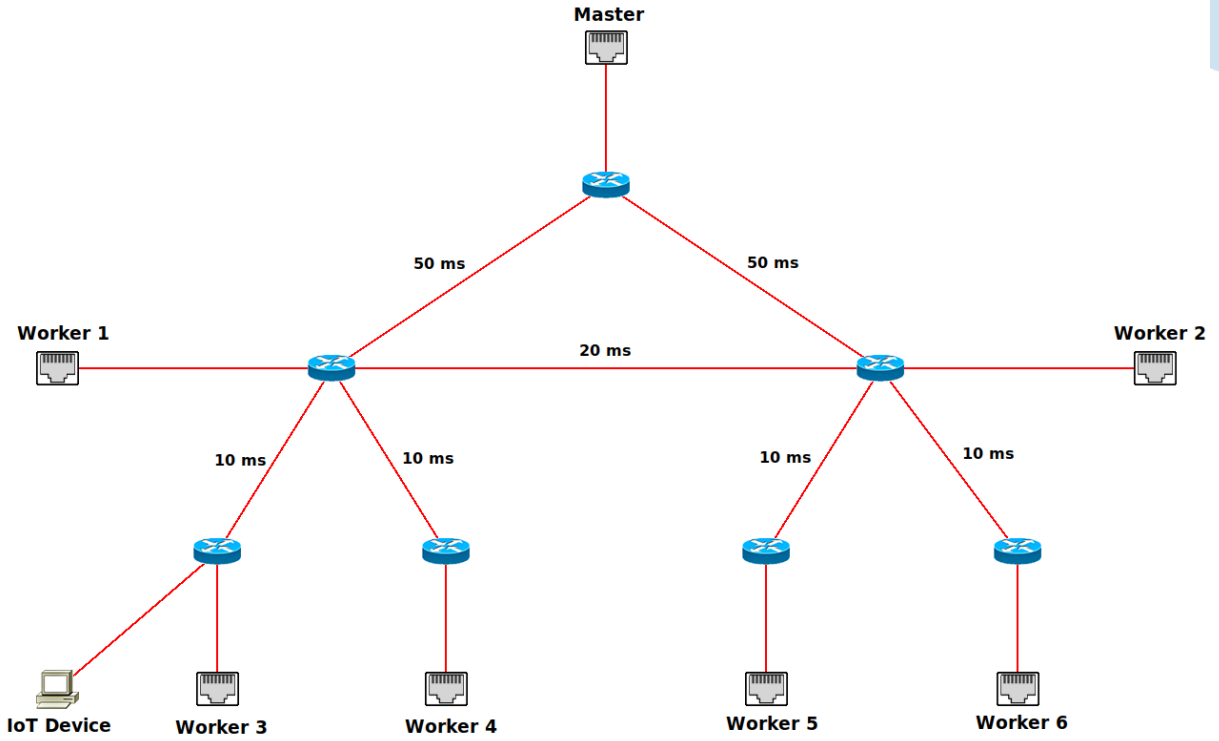
- Runs as a Kubernetes DaemonSet
- HTTP server written in Golang
- Clients connect to the services defined in the request header
- Routing directly to service pod through "kube-proxy“
- Subscribes to events in the Kubernetes API



Software repository
https://github.com/AIoTwin/qedgeproxy

Evaluation

- K3s cluster of 7 nodes
- Network emulation with Imunes
- Three configurations
  - Kubernetes NodePort Service
  - Proximity-based routing (α=0.8 and α=1.0) [1]
  - QEdgeProxy



[1] A. J. Fahs and G. Pierre, “Proximity-aware traffic routing in distributed fog computing platforms,” in 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2019

Evaluation scenarios

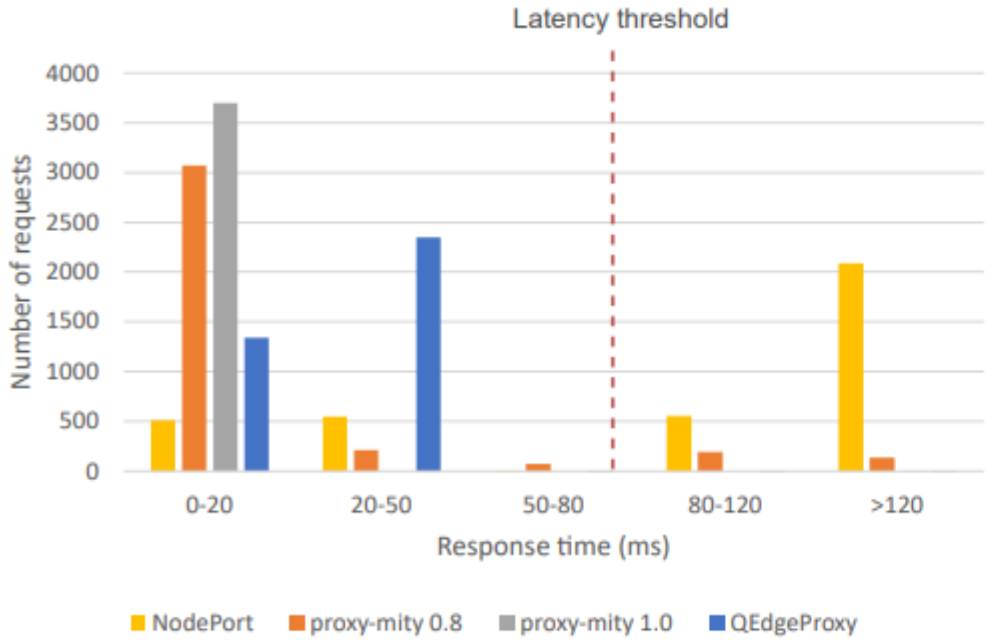
- Instance s_i running on node Worker i
- QoS requirement:
  - Latency < 80 ms
- Static scenario
  - 7 instances, one per node
- Dynamic scenario
  - Instance failures
  - Network changes
  - Increased processing latency

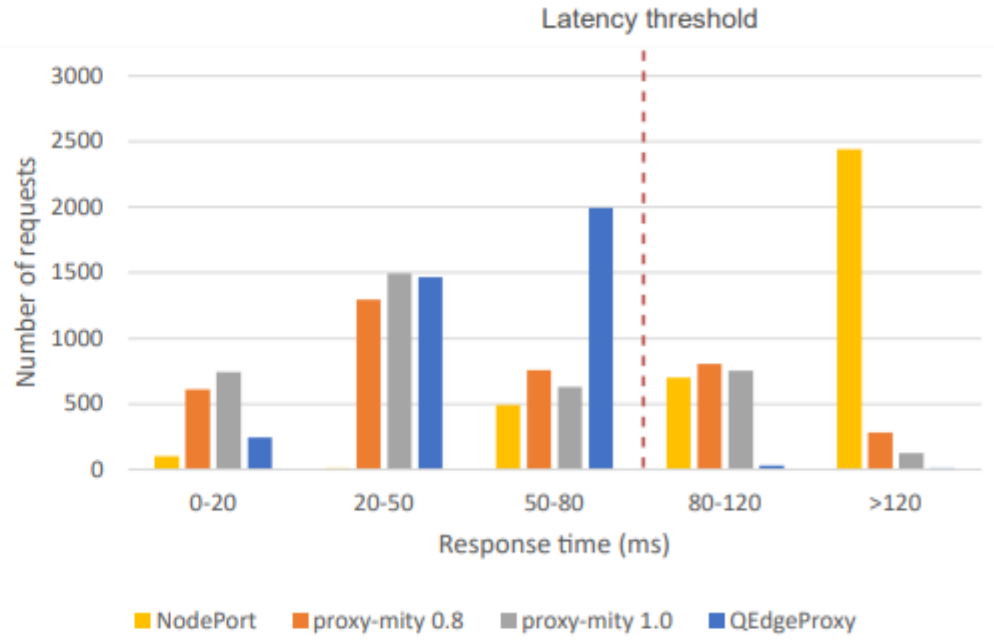
TABLE I: System changes in the dynamic scenario.

Timestamp	s_1	s_2	s_3	s_4	s_5	s_6	s_7
T0	✓	✓	x	✓	✓	✓	✓
T1	✓	✓	✓	✓	✓	✓	✓
T2	✓	✓	✓↑	✓	✓	✓	✓
T3	✓	✓	x	✓	✓	✓	✓
T4	✓↑	✓	x	✓	✓	✓	✓

↑: increased processing time; ↑: increased network latency

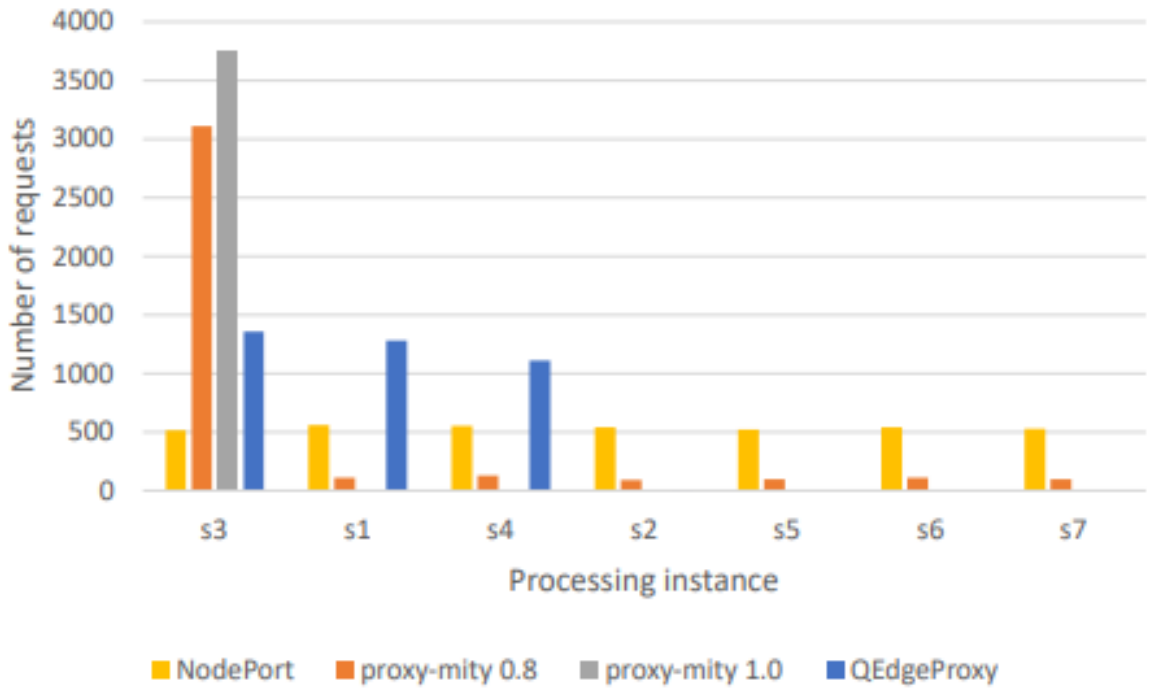
Results – response time





Configuration	Average response time		Successful request rate (QoS)	
	Static	*Dynamic*	*Static*	*Dynamic*
NodePort	120,11 ms	120,11 ms	28,70%	28,70%
proxy-mity 0.8	20,29 ms	61,05 ms	91,01%	70,78%
proxy-mity 1.0	**6,02 ms**	51.40 ms	**99,97%**	76,27%
QEdgeProxy	26,94 ms	**44,04 ms**	99,86%	**98,86%**

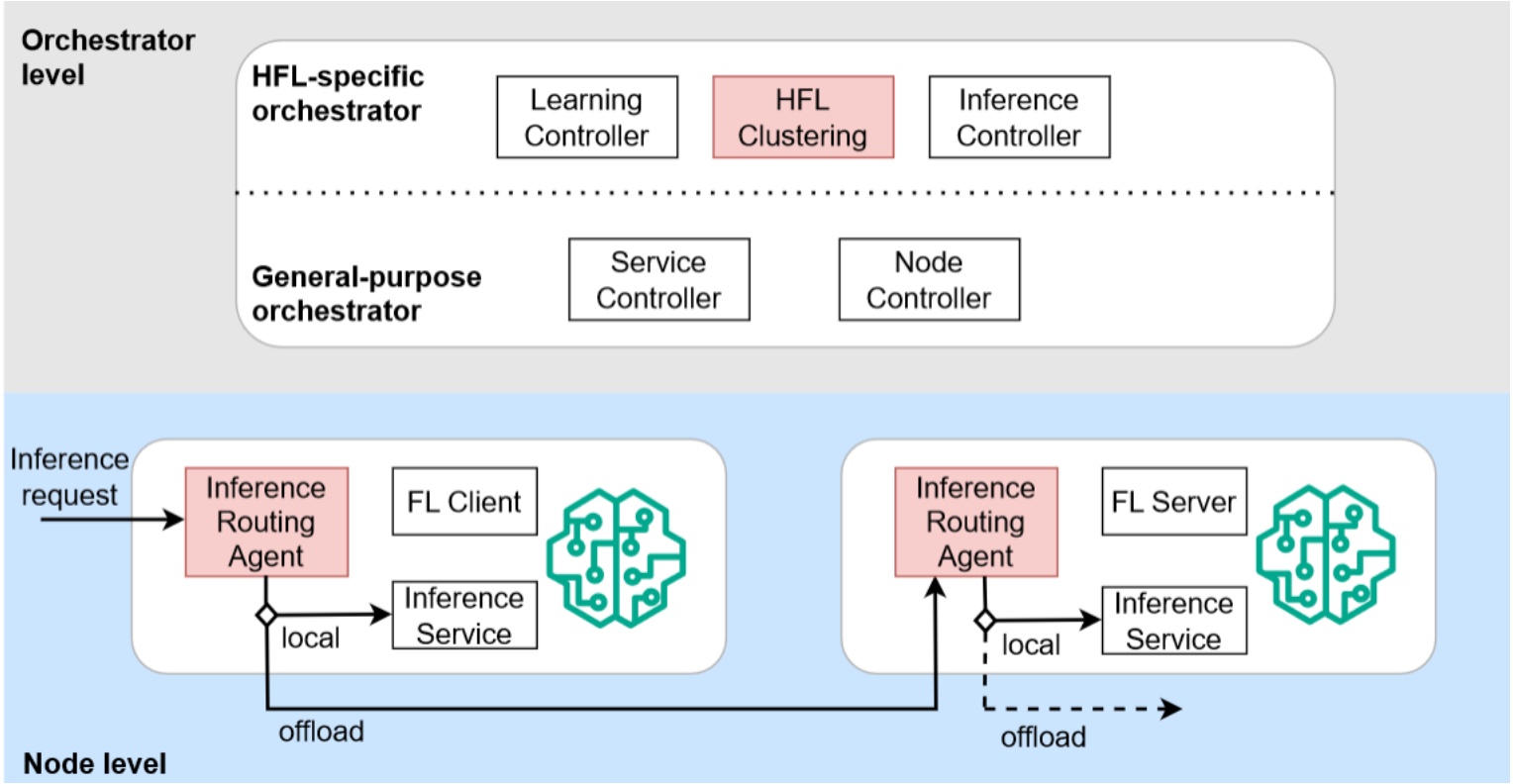
Results – load balancing



QEdgeProxy: conclusion

- QEdgeProxy effectively balances load and continuously maintains QoS per individual clients
- It outperforms both built-in Kubernetes load balancing and a state-of-the-art solution, serving a high rate of requests that meet the QoS requirements with minimal computational overhead

In addition: FL & inference



joint orchestration of distributed training and inference serving in a hierarchical FL environment

A. Lackinger, P. Frangoudis, I. Čilić, A. Furutanpey, I. Murturi, I.Podnar Žarko, S. Dustdar. Inference Load-Aware Orchestration for Hierarchical Federated Learning. 2024 IEEE 49th Conference on Local Computer Networks (LCN), October 2024. https://arxiv.org/abs/2407.16836

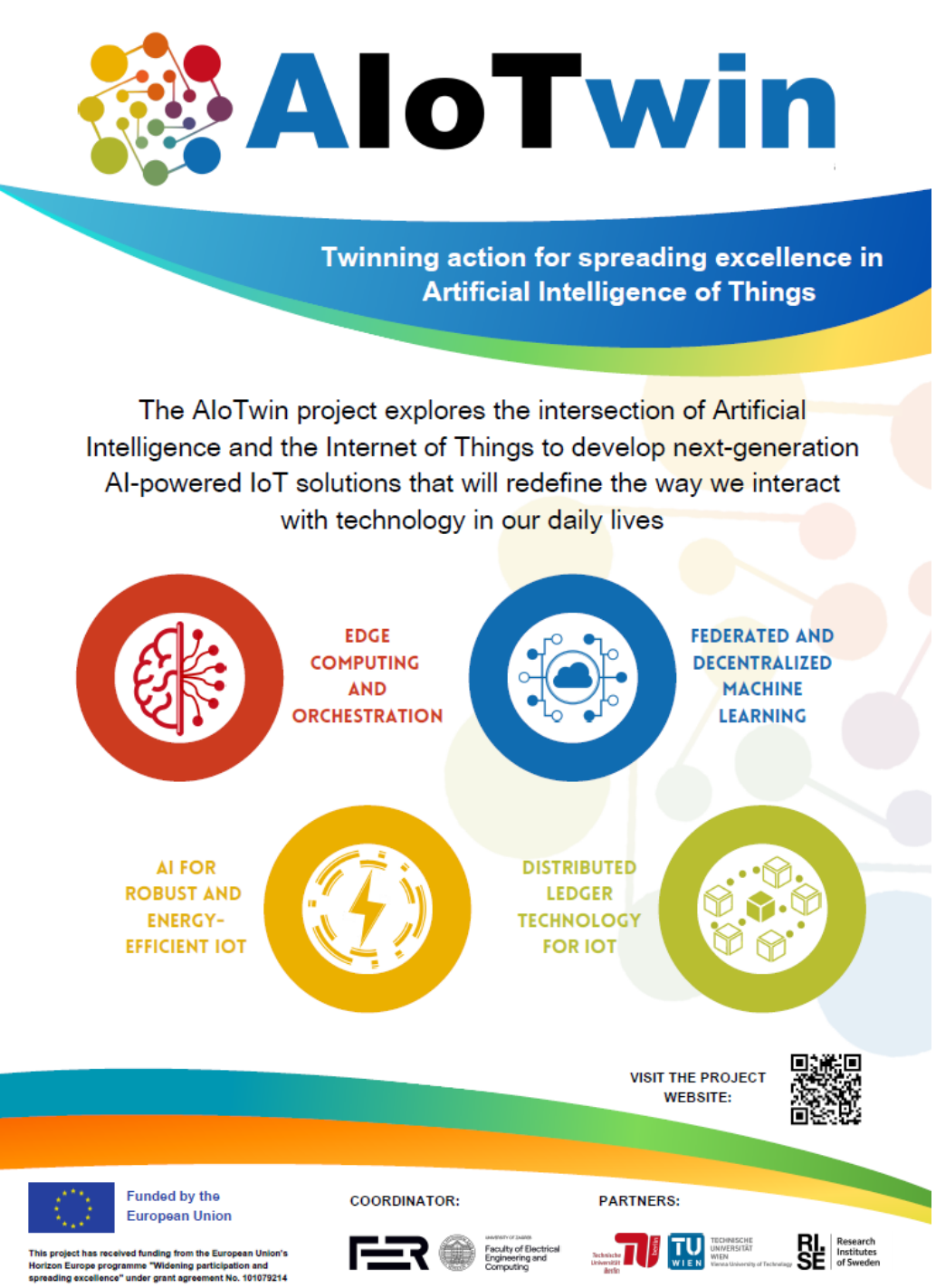
AIoTwin

Future Work

Future Work

- Extend the orchestrator with:
  - support for scheduling and deployment of **inference** service across the continuum based on the placement of clients that access them
  - scheduler that takes into account the **energy consumption** of both learning and inference services
- Integration of the orchestrator with QEdgeProxy to route data to appropriate inference service instances
  - routing of inference requests from IoT devices to inference services based on different QoS requirements, such as request latency or accuracy of inference models
- Novel decision-making and reconfiguration algorithms and large-scale experiments

AloTwin team



Thank you!

Time for questions.

Follow us at:

- https://www.aiotwin.eu/
- https://www.linkedin.com/company/iotlabfer/
- https://twitter.com/IoTLabFER
- https://github.com/aiotwin