UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI INGEGNERIA INFORMATICA, MODELLISTICA, ELETTRONICA E SISTEMISTICA
DIMES

Dipartimento di Eccellenza 2018-2022

Funded by the European Union

MLSysOps

Machine Learning for System Operation and Application Orchestration in the IoT-Edge-Cloud Continuum

Raffaele Gravina

University of Calabria

June 29, 2024

SHANGHAI MARITIME UNIVERSITY

UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI INGEGNERIA INFORMATICA, MODELLISTICA, ELETTRONICA E SISTEMISTICA
DIMES

Dipartimento di Eccellenza 2018-2022

About me

- BSc and MSc in Computer Engineering from the University of Calabria, Italy (2003, 2007)
- Researcher at the **Wireless Sensor Network Lab - Berkeley**, California (2008-2010)
- PhD degree in Computer and Systems Engineering from the University of Calabria (2012)
- PostDoc at UNICAL (2013-2015)
- Assistant Professor of Computer Engineering at **UNICAL** (2016-2022)
- **Associate Professor** at UNICAL since 2022
- Member of **SPEME Research Group**
- Co-founder and CTO of health-care sector at SenSysCal Srl, a spin-off which develops Internet-of-Things and sensor-based m-Health

- Research interests:
  - **Wearable Computing Systems**
  - Internet of Things
  - Pattern Recognition and Machine Learning
  - **Device-Edge-Cloud Continuum**

The University of Calabria




- Established in **1968**
- Campus located on over 350K m^2, with Departments, Lecture Halls, Laboratories, Libraries, Theaters, and Residential Centers.
- **Biggest campus in Italy** and one of the "greenest" in Europe
- 25.0000 students (over 1.000 are internationals)
- 800 professors
- 650 administrative personnel
- 14 Departments
- 80 Degree courses (15 delivered in English)
- 10 PhD Schools
- Technology transfer with the support of the spin-off incubator and Liaison Office
- News! UNICAL gained 50 positions in the QS ranking → ranked #901-950 in 2025!

UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI INGEGNERIA INFORMATICA, MODELLISTICA, ELETTRONICA E SISTEMISTICA
DIMES

Dipartimento di Eccellenza 2018-2022

The SPEME Group

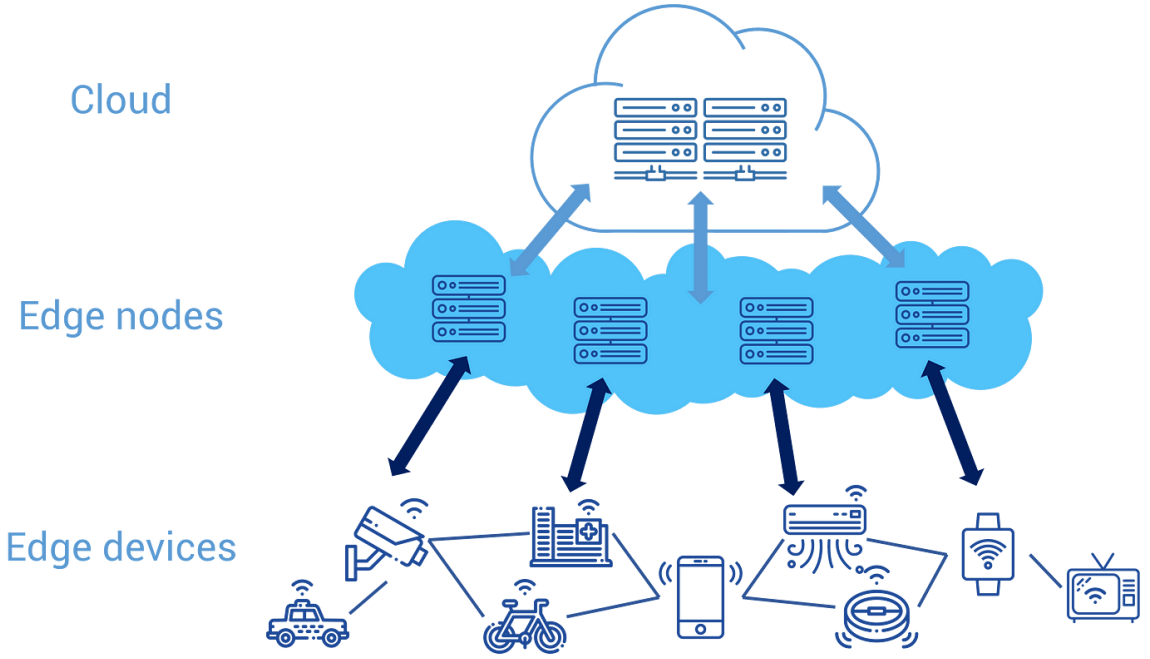
- **Smart PErvasive and Mobile systems Engineering** (SPEME)
- Part of the Department of Computer, Modelling, Electronics, and Systems Engineering (DIMES)
- 30+ people
  - 8 faculty members,
  - 2 adjunct professors/researchers
  - 5 PostDocs
  - **17 PhD students**,
  - + several MS students and external collaborators
- Areas of expertise:
  - Internet of Things
  - Wearable Computing Systems
  - Edge Intelligence
  - Wireless Sensor Networks
  - Multi-Agent Systems
  - Large-scale Decentralized, Mobile and Cloud Computing
  - Vehicular Area Networks
  - Machine Learning
  - Process Mining

Website:
https://speme.dimes.unical.it

Outline

- Introduction
- A Framework for system and application deployment in the edge-cloud continuum
- High-level modeling
- Architecture & APIs
- Use case scenario

Introduction



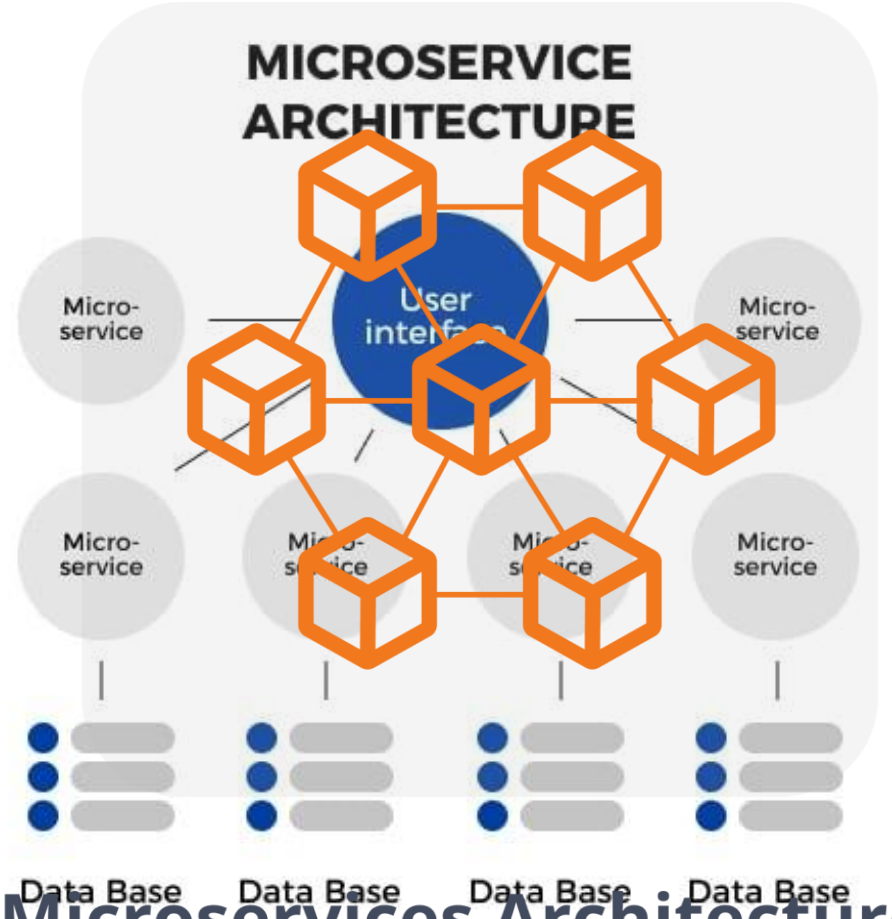
Continuum systems

- Applications are moving outside the cloud and start involving nodes and resources toward or directly at the edge of the Internet
  - including powerful but also resource-constrained IoT devices
- Scale, heterogeneity, dynamics and complexity increases
- Practically impossible to monitor/manage by a human

*The vision of **autonomic computing systems**, which can manage themselves with little or no involvement of the application/system administrator, becomes more relevant than ever before!*

Application deployment architectures

Monolithic Architecture



Microservices Architecture

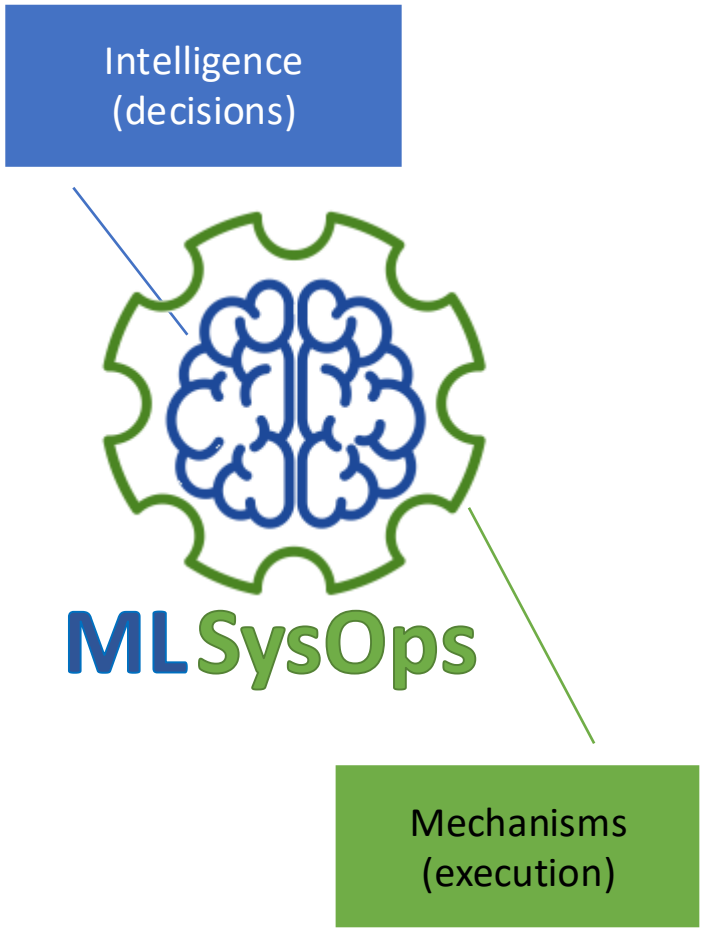
kubernetes

docker

Research challenges

*Autonomic system management and configuration in the **cloud-edge-IoT continuum** using **AI/ML methods***

- Focus on modular, distributed applications
  - Comprised out of independently executable components
- Different management aspects
  - Deployment, computing, storage, communication/networking, trust
- Disassociation of management from control
  - Develop AI/ML-ready (policy-neutral) mechanisms
  - Take decisions using suitable ML models
- Key AI/ML properties
  - Distributed / hierarchical approach
  - Continual learning / efficient model retraining
  - Explainability



Objectives

1. Deliver an open AI-ready, agent-based framework for holistic, trustworthy, scalable, and adaptive system operation across the heterogeneous cloud-edge continuum.

2. Develop an AI architecture supporting explainable, efficiently retrainable ML models for end-to-end autonomic system operation in the cloud-edge continuum.

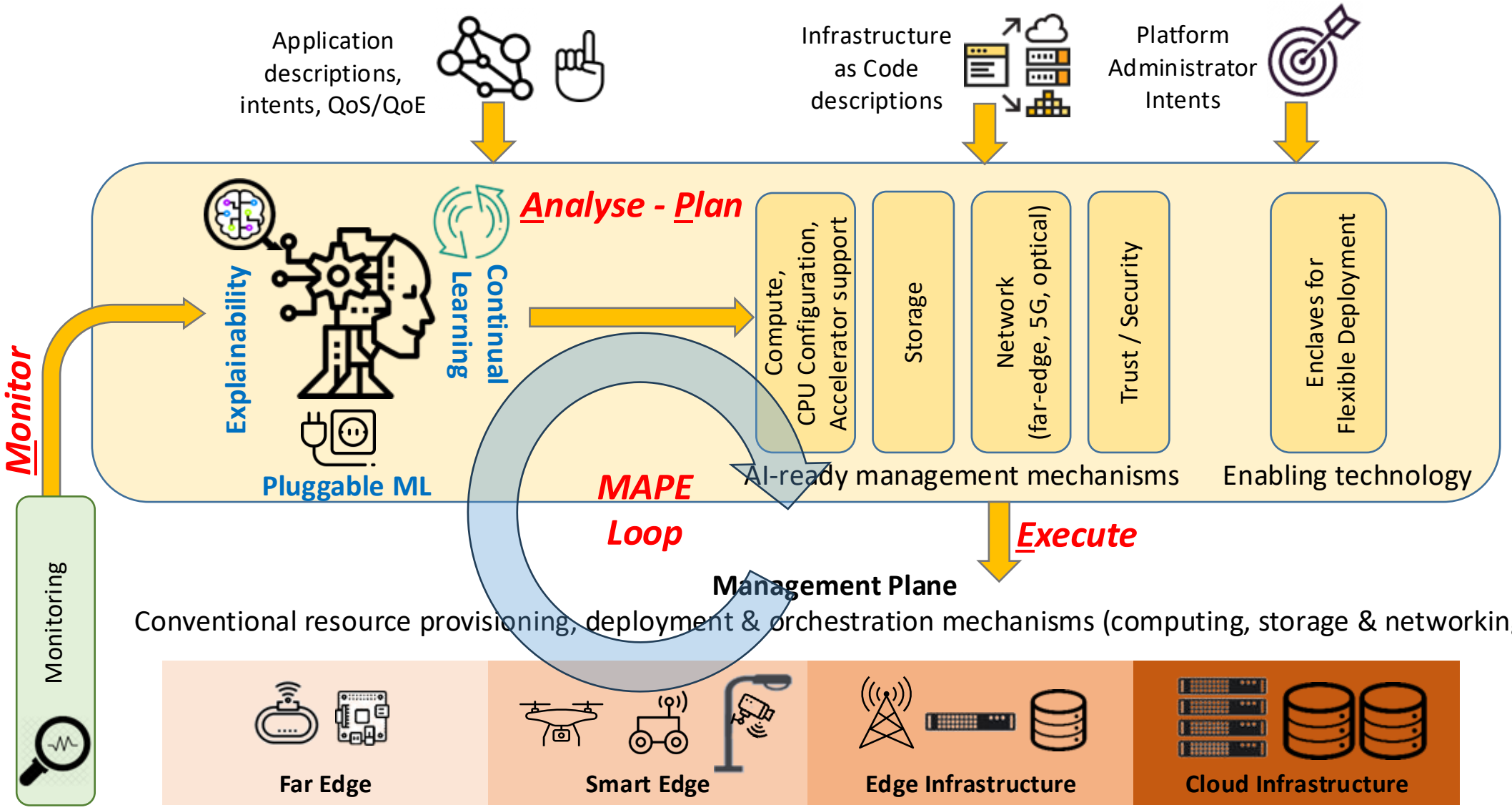
3. Enable efficient, flexible, and isolated execution across the heterogeneous continuum.

4. Support green, resource-efficient, and trustworthy system operation, while satisfying application QoS/QoE requirements.

UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI INGEGNERIA INFORMATICA, MODELLISTICA, ELETTRONICA E SISTEMISTICA
DIMES

Dipartimento di Eccellenza 2018-2022

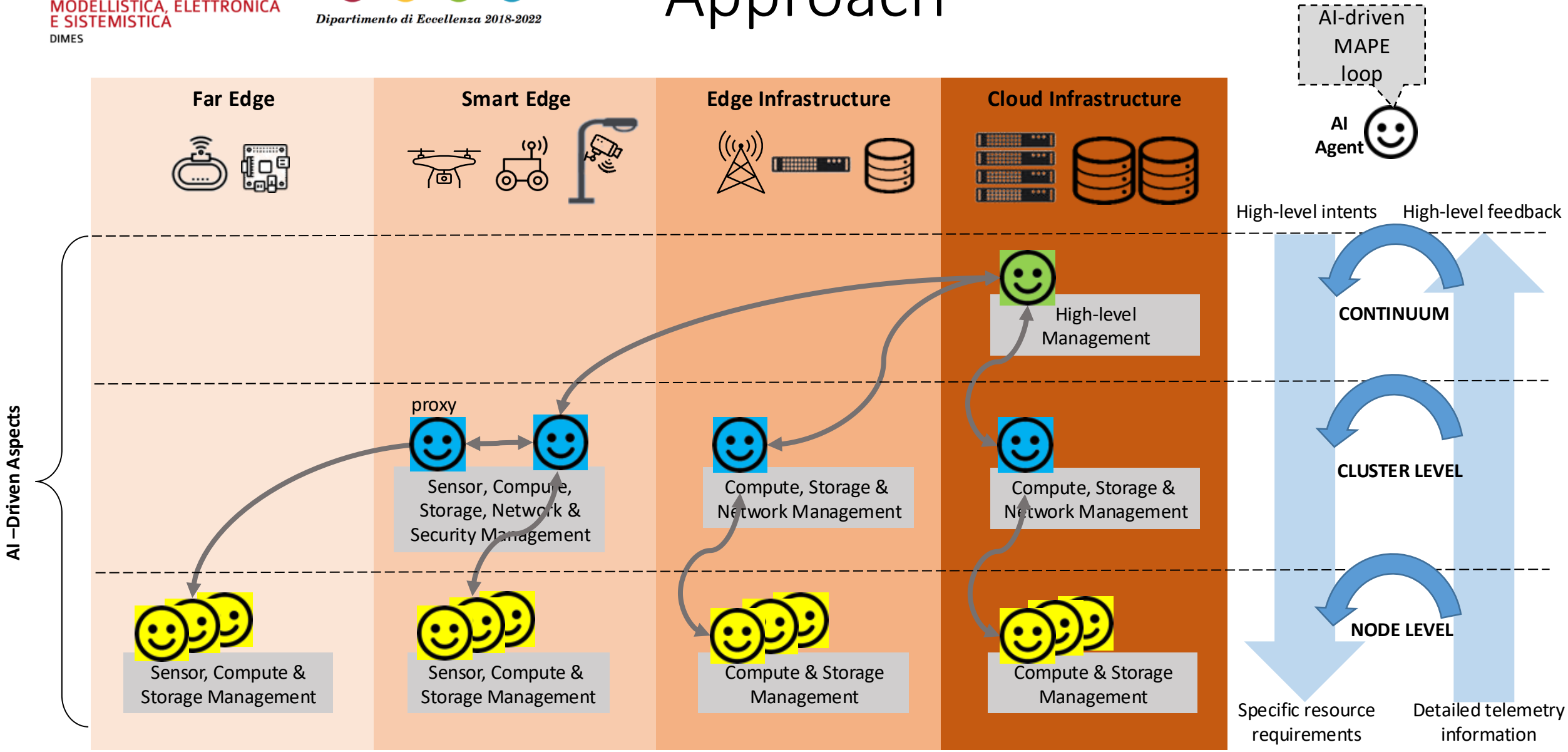
Concept



UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI INGEGNERIA INFORMATICA, MODELLISTICA, ELETTRONICA E SISTEMISTICA
DIMES

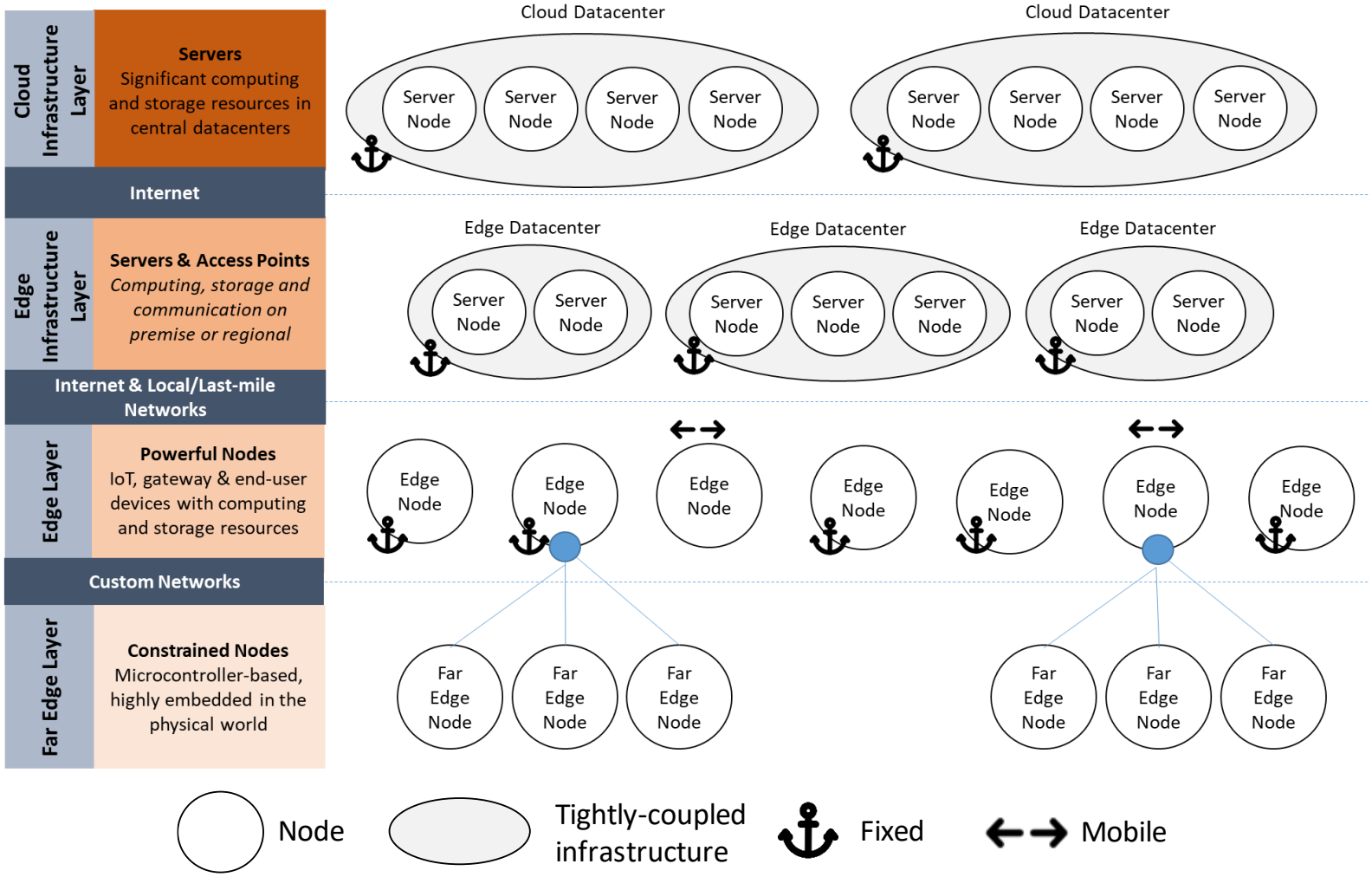
Dipartimento di Eccellenza 2018-2022

Approach

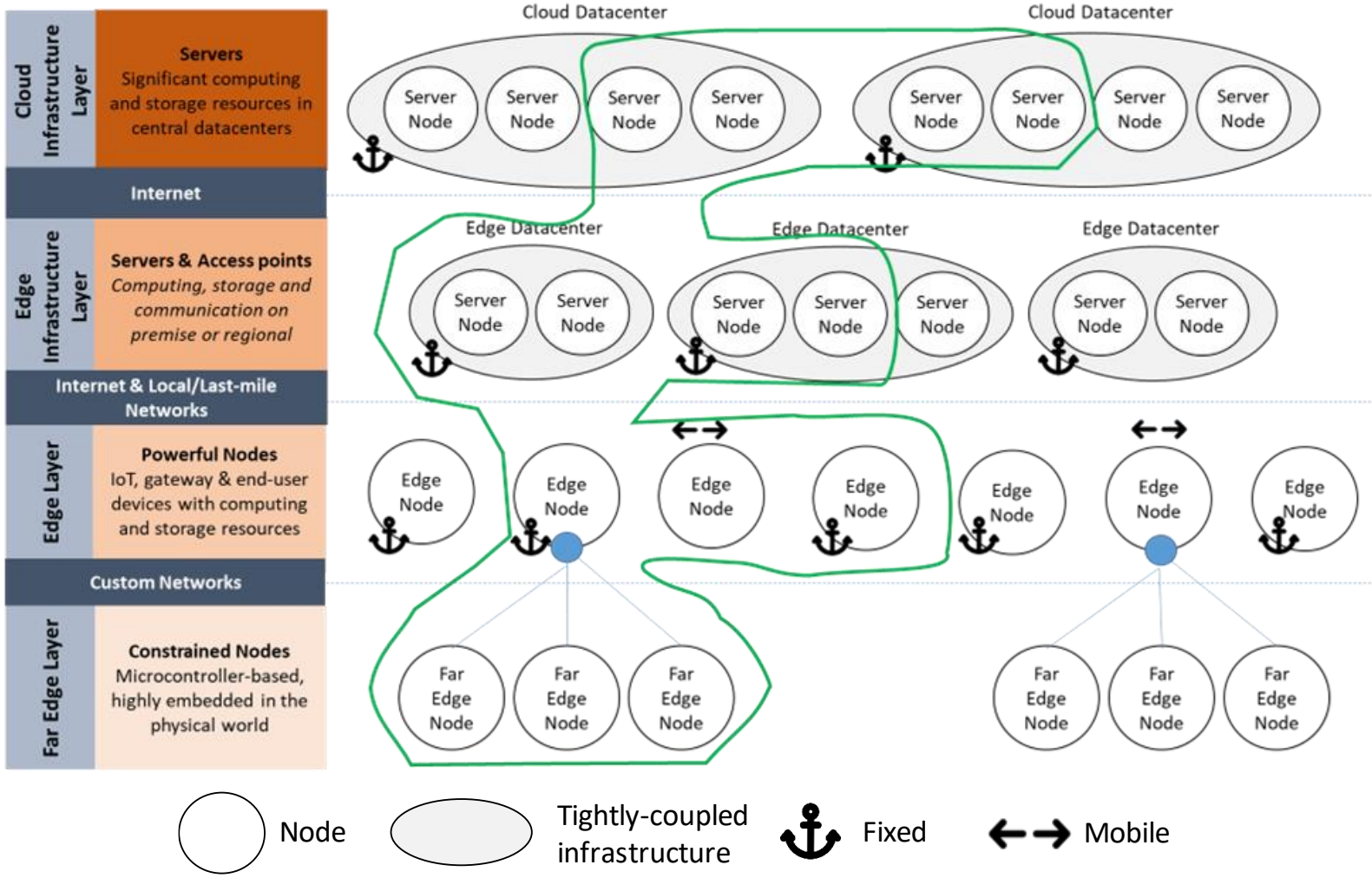


High-level modeling

System Model



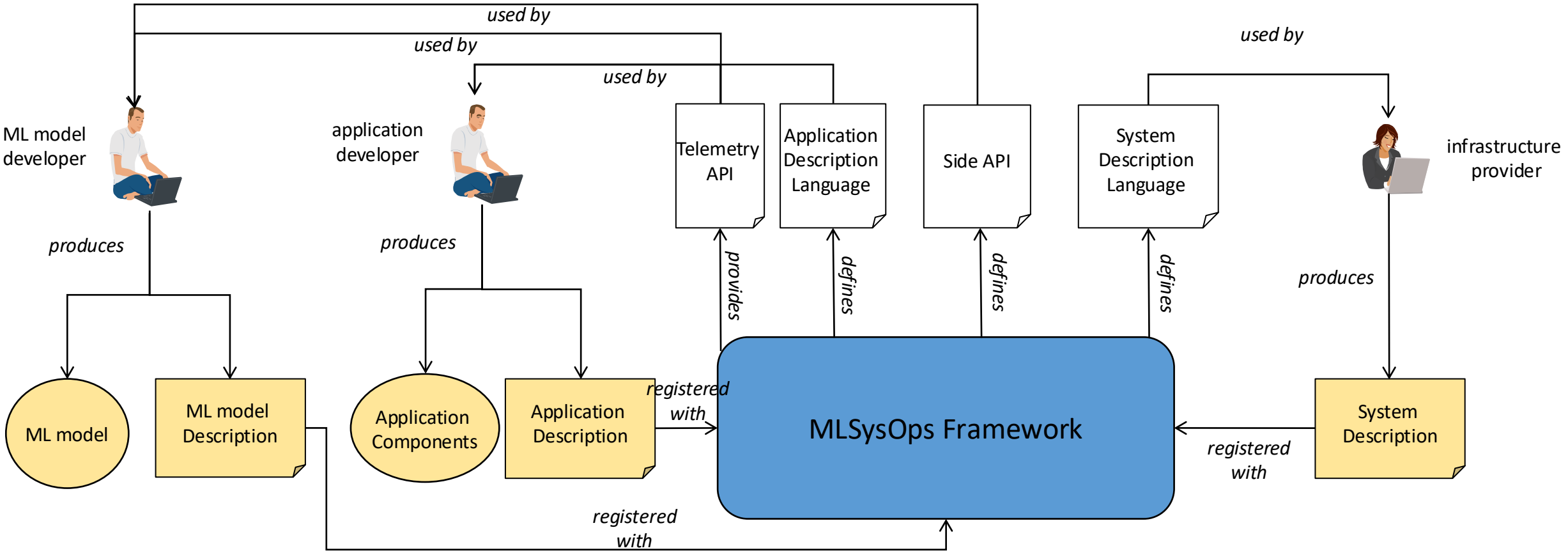
System Model - Role



PaaS within a given **slice**

- Deploys, orchestrates, runs and adapts distributed applications
- Manages system infrastructure
- Employs ML-based methods
- Exploits system resources to enable continual and explainable ML alongside application operation

Main Actors and Interactions

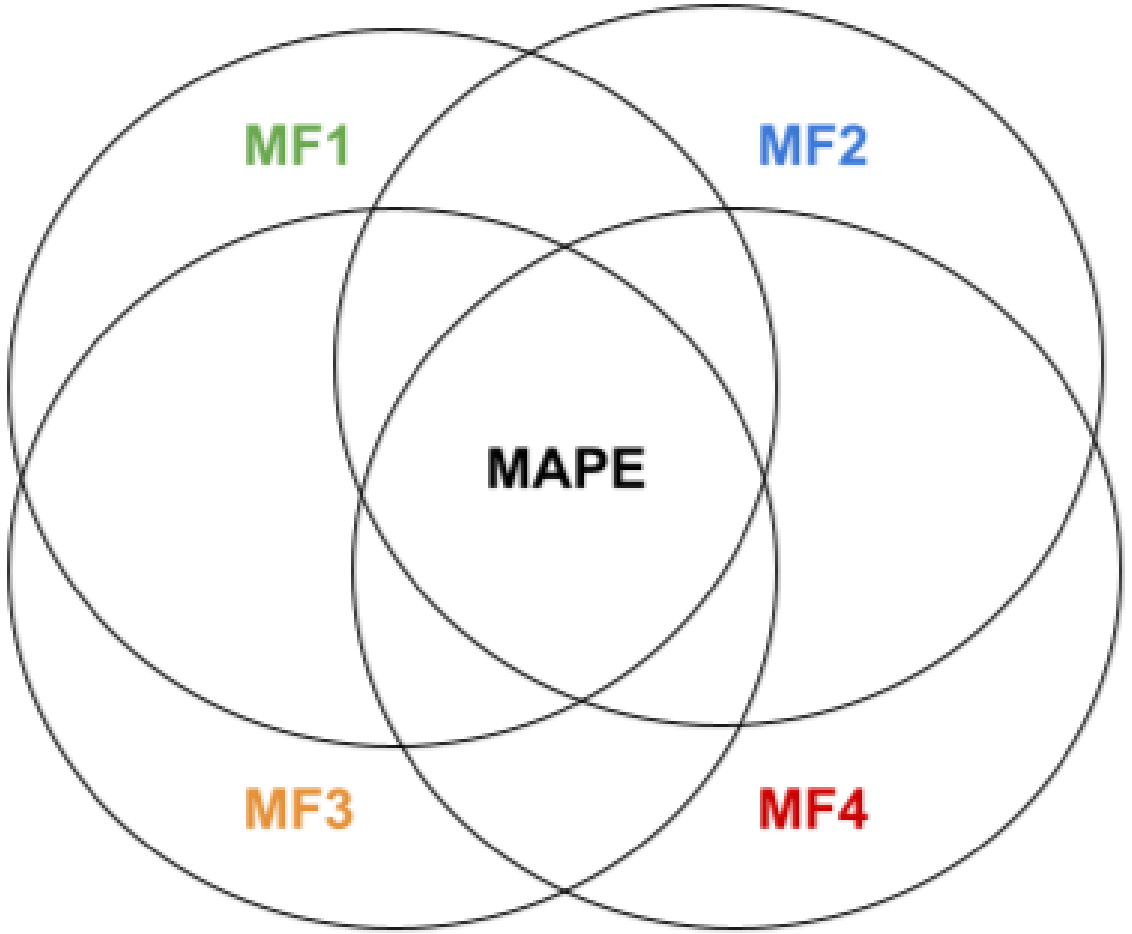


Application Model

- **Modular applications**, consisting of **components**
- Components & their interactions form an **application graph**
- Components are typically **packaged as containers**
  - Other enclaves – more appropriate – can be autogenerated on a case-by-case basis
  - They come with:
    - Resource requirements
    - QoS requirements
  - Different implementations may be available for select components
    - Corresponding to different points in the resource requirements vs. QoS space
- Applications are specified and registered to the system via structured application descriptions

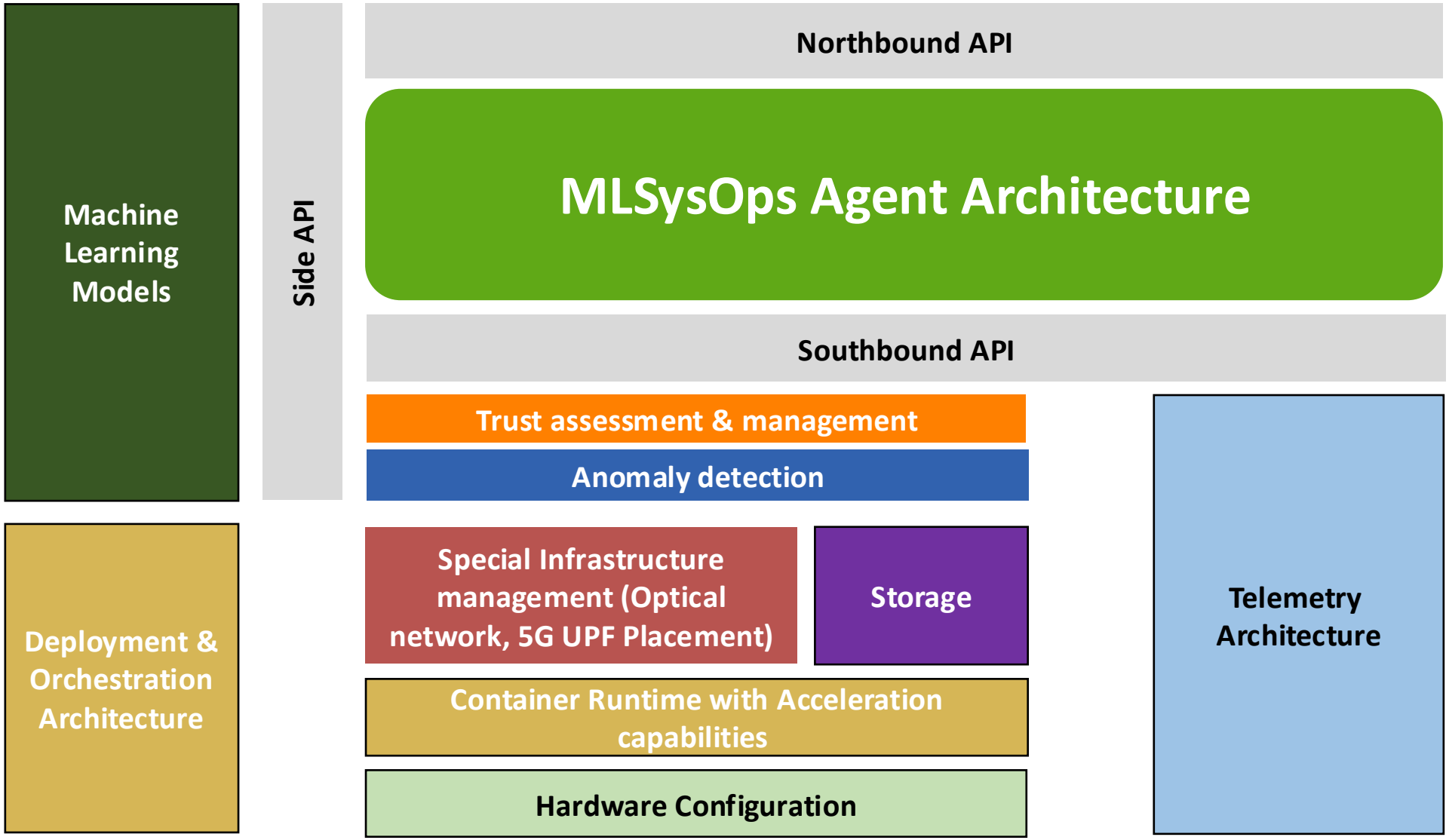
Functional Model

- MF1: **Management**
  - Monitor
- MF2: **Service & Control**
  - Monitor, Analyse, Plan, Execute
- MF3: **Communication & Interface**
  - Monitor, Execute
- MF4: **Interoperability**
  - Monitor, Analyse, Plan, Execute



Framework architecture

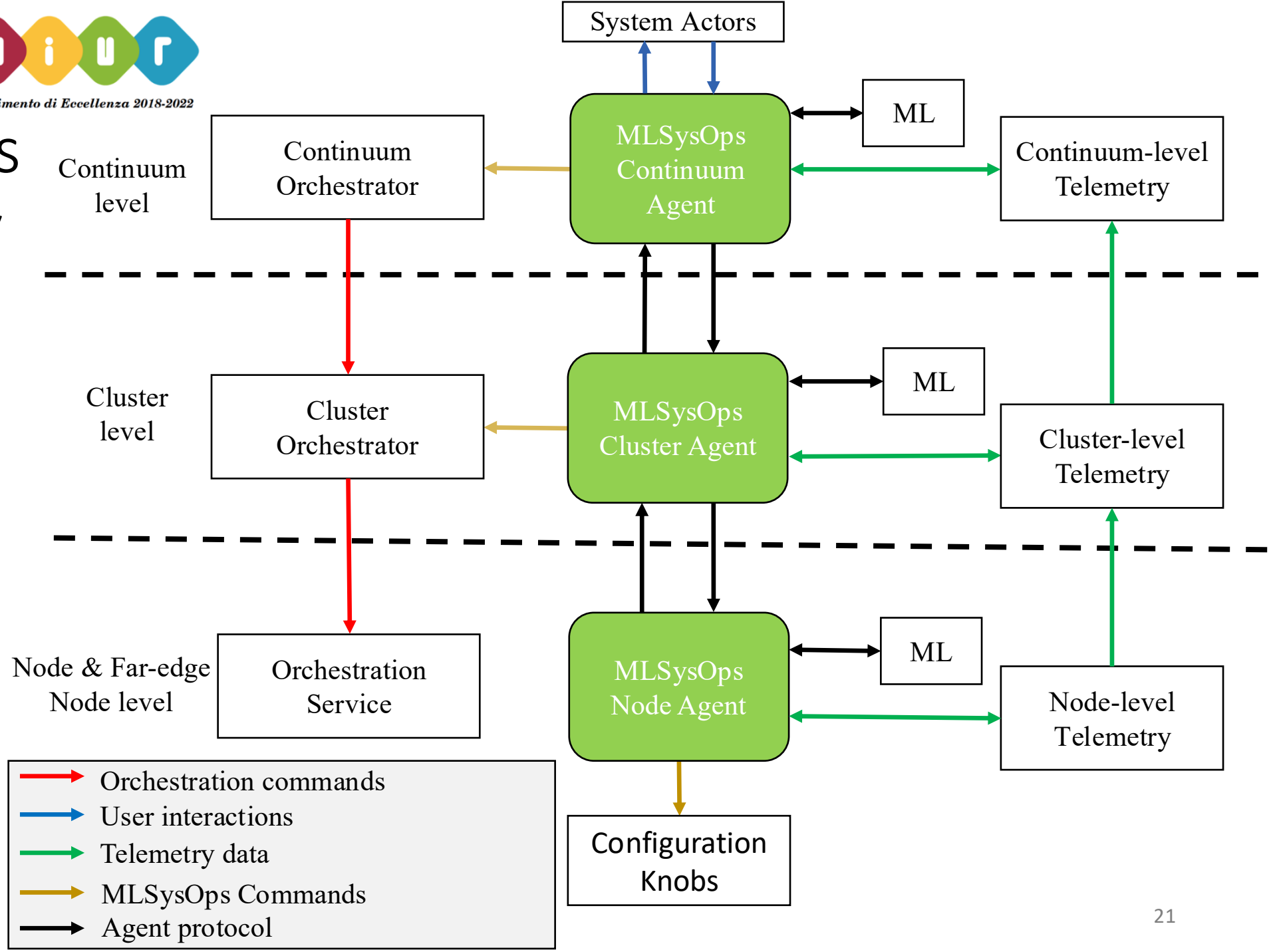
MLSysOps Software Architecture



UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI INGEGNERIA INFORMATICA, MODELLISTICA, ELETTRONICA E SISTEMISTICA
DIMES

Dipartimento di Eccellenza 2018-2022

MLSysOps Hierarchy

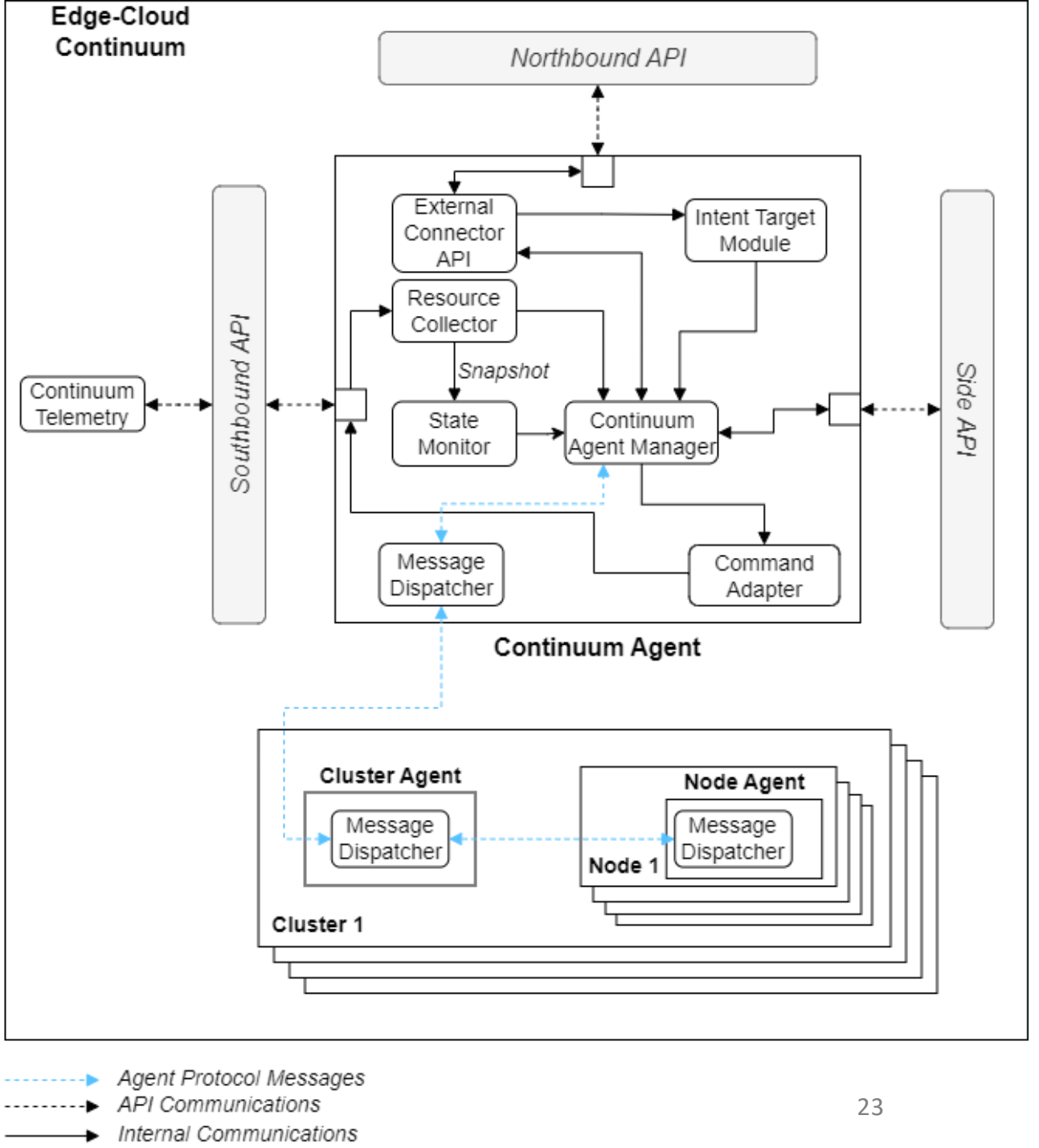


Agent-based implementation

Agent Architecture

Agent responsibilities:

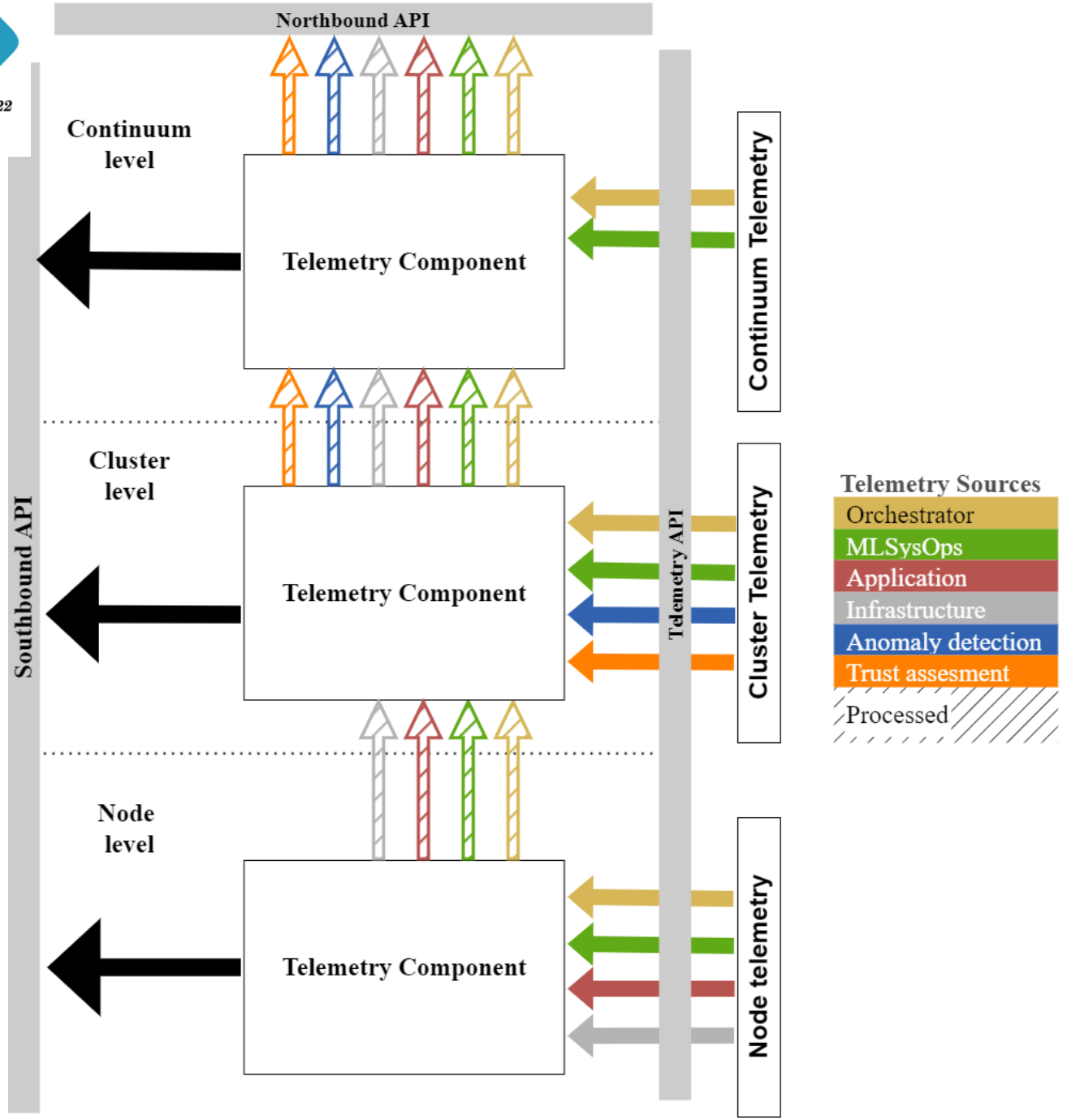
- High-Level Decision-Making
- Data Interaction and Forwarding
- Action Consistency Assessment
- Transforming Information into Action
- Model Retraining Management
- Anomaly Detection and Trust Evaluation
- Responsiveness to System Administrators' commands
- Transition to Conventional Techniques
- Logging for Accountability



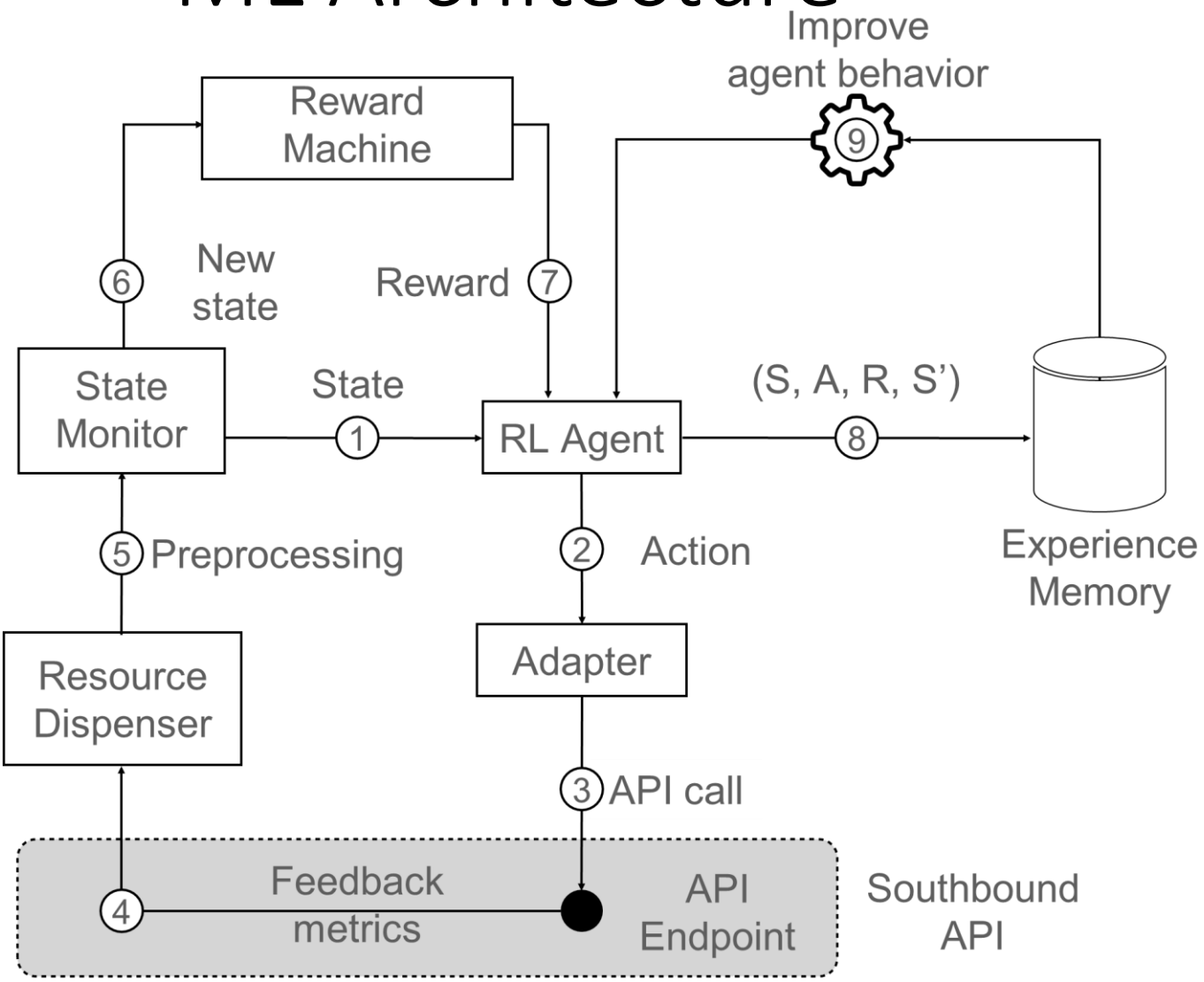
Agent Architecture – Inter-agent Message Types

Message Type	Examples	Description	Direction
Inform	Heartbeat	'Keep alive' message to notify higher-level agents of its current availability.	Upwards
	Telemetry Process Monitoring	Feedback on errors or the initiation/termination of telemetry streaming from lower-level agents.	Upwards
	Health Status Update	Details on the device's health status.	Upwards
Request	Heartbeat Request	Explicit heartbeat request	Downwards
	Perform Action Request	Command to child agents to perform a specific action.	Downwards
Agree	Perform Action	Acknowledgement of higher-level agents' perform action request.	Upwards
Refuse	Refuse to Perform Action	Refusing to perform a given action requested by higher-level agents.	Upwards
Cancel	Cancel Action	Cancel the request to perform a specific action.	Downwards
Failure	Failure to act.	Acknowledgement to notify an action was attempted, but the attempt failed.	Upwards

Telemetry Architecture



ML Architecture



1. The State Monitor sends the current state to the RL agent.
2. The RL agent chooses an action.
3. The act module translates the agent's action to the corresponding API call.
4. The underlying system sends the feedback metrics to the state monitor.
5. The Resource Dispenser collects the metrics, preprocesses them, cleans them, and sends them to the State Monitor.
6. The State Monitor creates a new state.
7. The Reward Machine calculates the reward and returns it to the RL agent.
8. The (state, action, reward, new action) tuple is saved in the experience memory.
9. Update the RL agent policy.

APIs

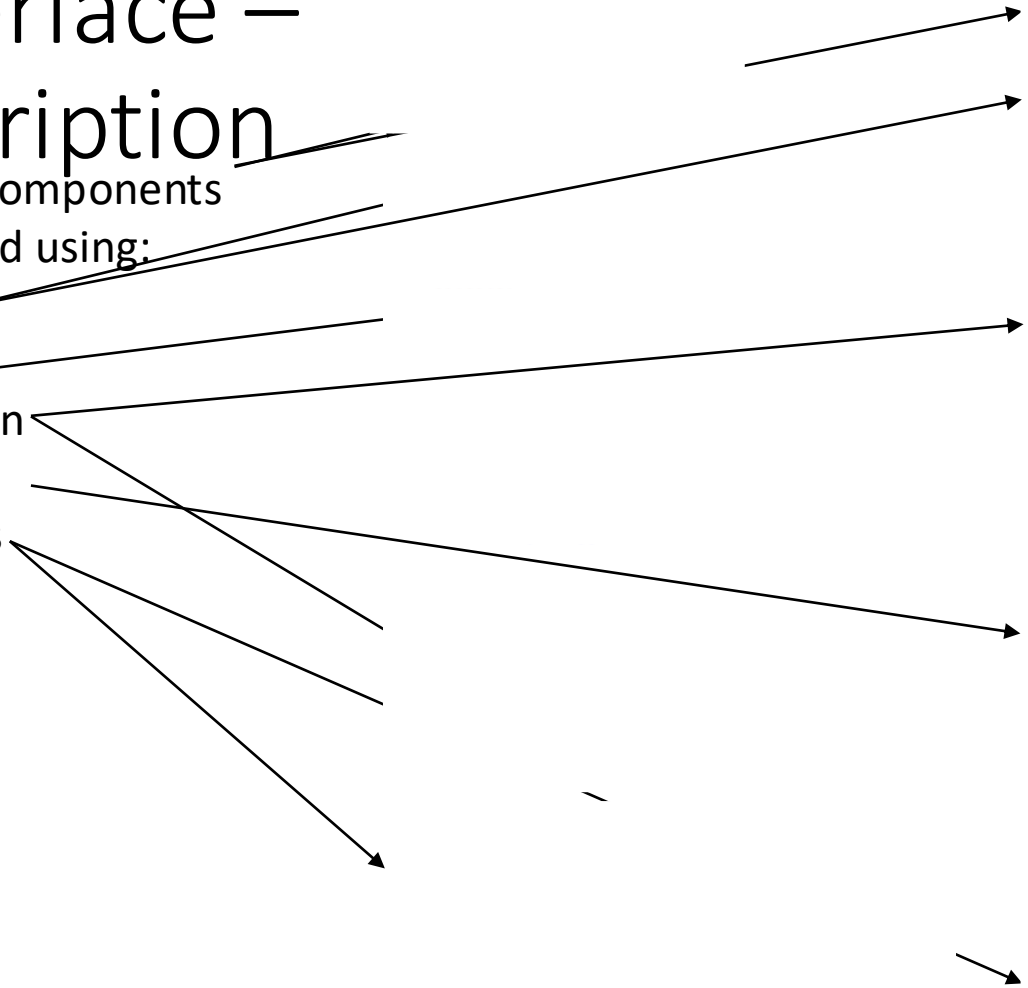
NorthBound Interface - APIs

API Call
runApp(AppDescription)
stopApp(AppDeploymentID)
listApps()
registerInfrastructure (list of ClusterDescriptions or list of DatacenterDescriptions or list of NodeDescriptions)
unregisterInfrastructure(list of ClusterIDs or list of DatacenterIDs or list of NodeIDs)
listInfrastructure(DatacenterID or ClusterID, or 0)
configTrust(list of NodeIDs, list of Indexes, list of weights)
getApplicationState(AppDeploymentID)
getApplicationPerformance(AppDeploymentID)
getNodeState(NodeID)
getDatacenterState(DatacenterID)
getClusterState(ClusterID)
setManagementMode(0[conventional], 1[ML])
GetApplicationLevelExplanations(currentConfiguration)
GetAdminLevelExplanations(currentConfiguration)
SetSystemTarget(list of clusterIDs, list of DatacenterIDs, list of NodeIDs, targets: [minNodeUsage, minEnergyConsumption, maxGreenEnergy])

- A set of **16 API calls**
- Used by
  - **application owners**
  - **infrastructure administrators**

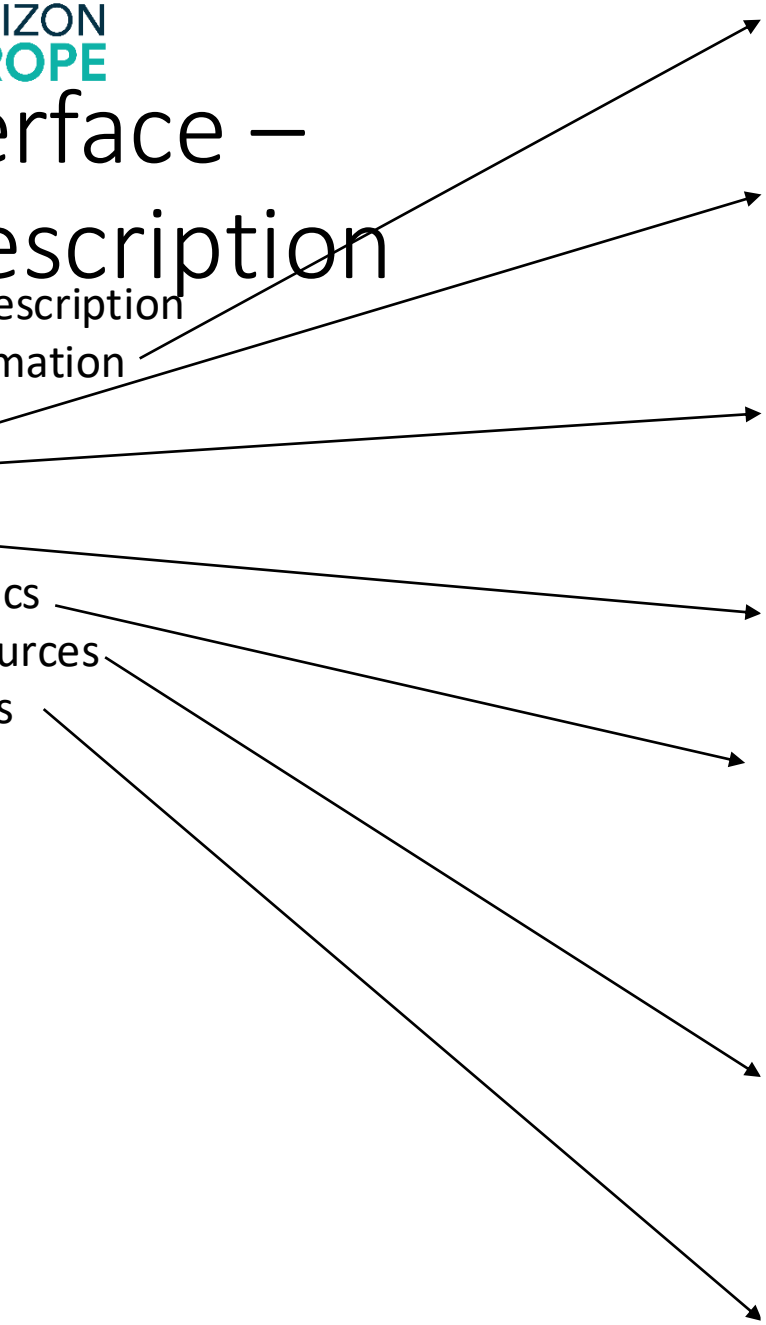
NorthBound Interface – Application Description

- Each application is a set of components
- Each component is described using:
  - Placement options
  - Sensor requirements
  - QoS-related information
  - Storage requirements
  - Resource requirements



Northbound Interface – Infrastructure Description

- Fixed edge node description
  - General information
  - Environment
  - Acceleration
  - Storage
  - Hardware specs
  - Network resources
  - Power sources



Northbound Interface – Infrastructure Description

- Sensor-equipped node description
  - Sensor information

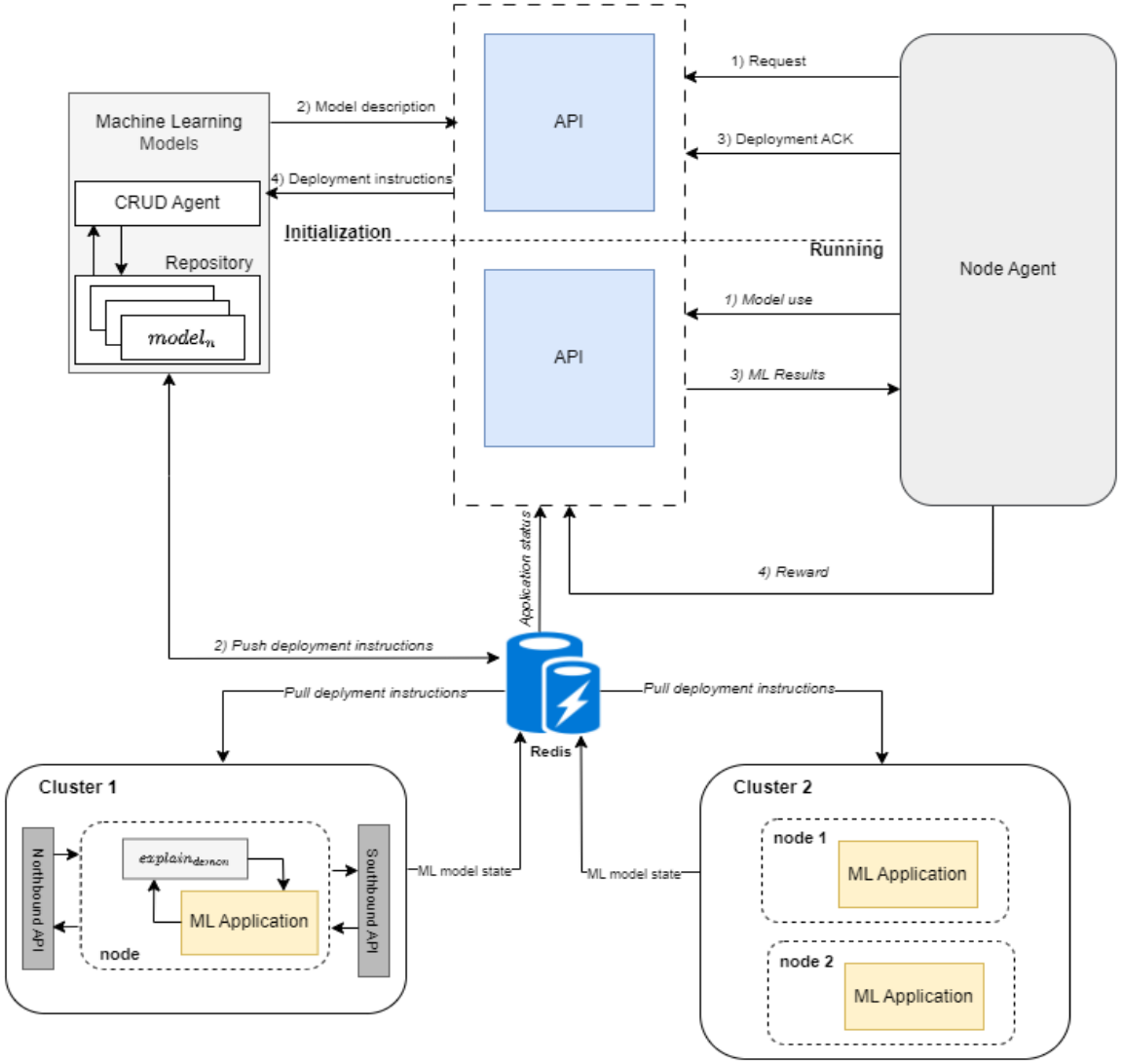
Side Interface

Initialization API

1. Request. Agent specifies the kind of ML model it needs.
2. Model description. The API returns a list of all the ML models that match the agent's needs.
3. Deployment ACK. Agent picks one model and sends back a request to API to deploy the selected ML model.
4. Deployment instructions. The API pushes the request to the queue for deployment and returns a deployment ID to the agent.

Model use API

1. Model use. Request inferences or predictions. Input data will be required for this API call.
2. ML results. API returns prediction results.
3. Reward. Based on the return results and action performed, a reward will be returned to API to train the RL agent to improve its performance.



Use case
Indoor Localization and Tracking

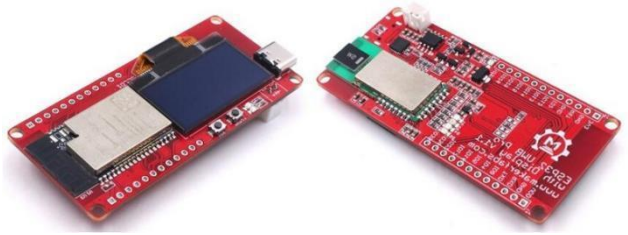
Objectives

- Show how MLSysOps Agents can communicate through the layers of the edge-cloud continuum.
- Show the capability of MLSysOps Agents to monitor and execute commands to orchestrate the application microservices.
- Show the capability of MLSysOps proxy Agents to configure application behavior at the far edge devices.
- Show how agents interact to adapt the computation performed to compute the tag locations as a function of the number of active tags.

Testbed Configuration

XMPP

Raspberry Pi 3



ESP32 UWB pro position sensor

Raspberry Pi Zero W

Local server

IoT containerized application

Agent Configuration

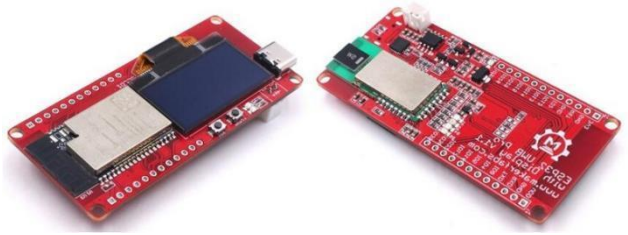
CLUSTER AGENT

XMPP

Raspberry Pi 3

NODE AGENT

Local server



PROXY AGENT

ESP32 UWB pro position sensor

NODE AGENT

Raspberry Pi Zero W

FAR EDGE

SMART EDGE

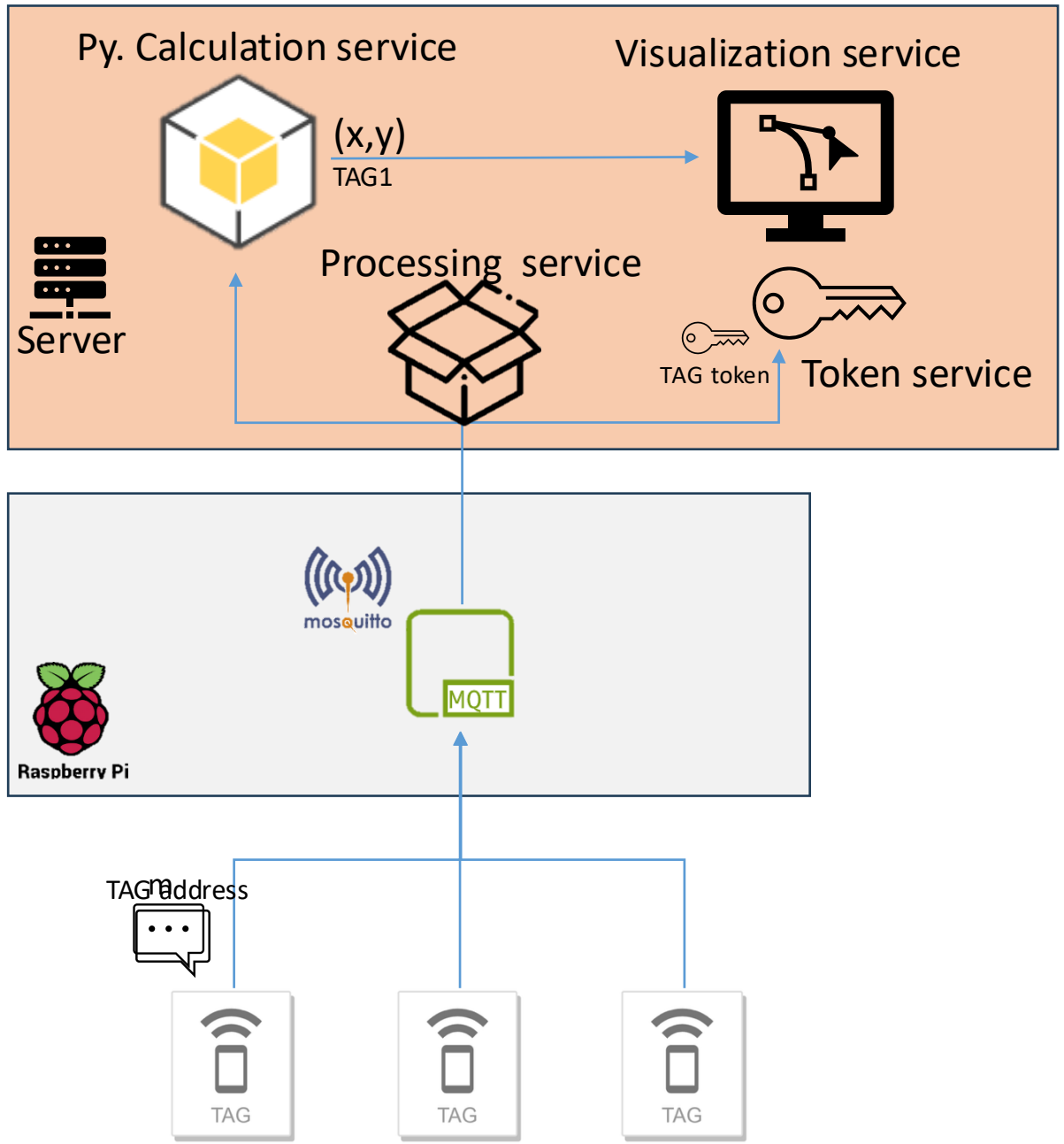
EDGE INFRASTRUCTURE

Application & infrastructure

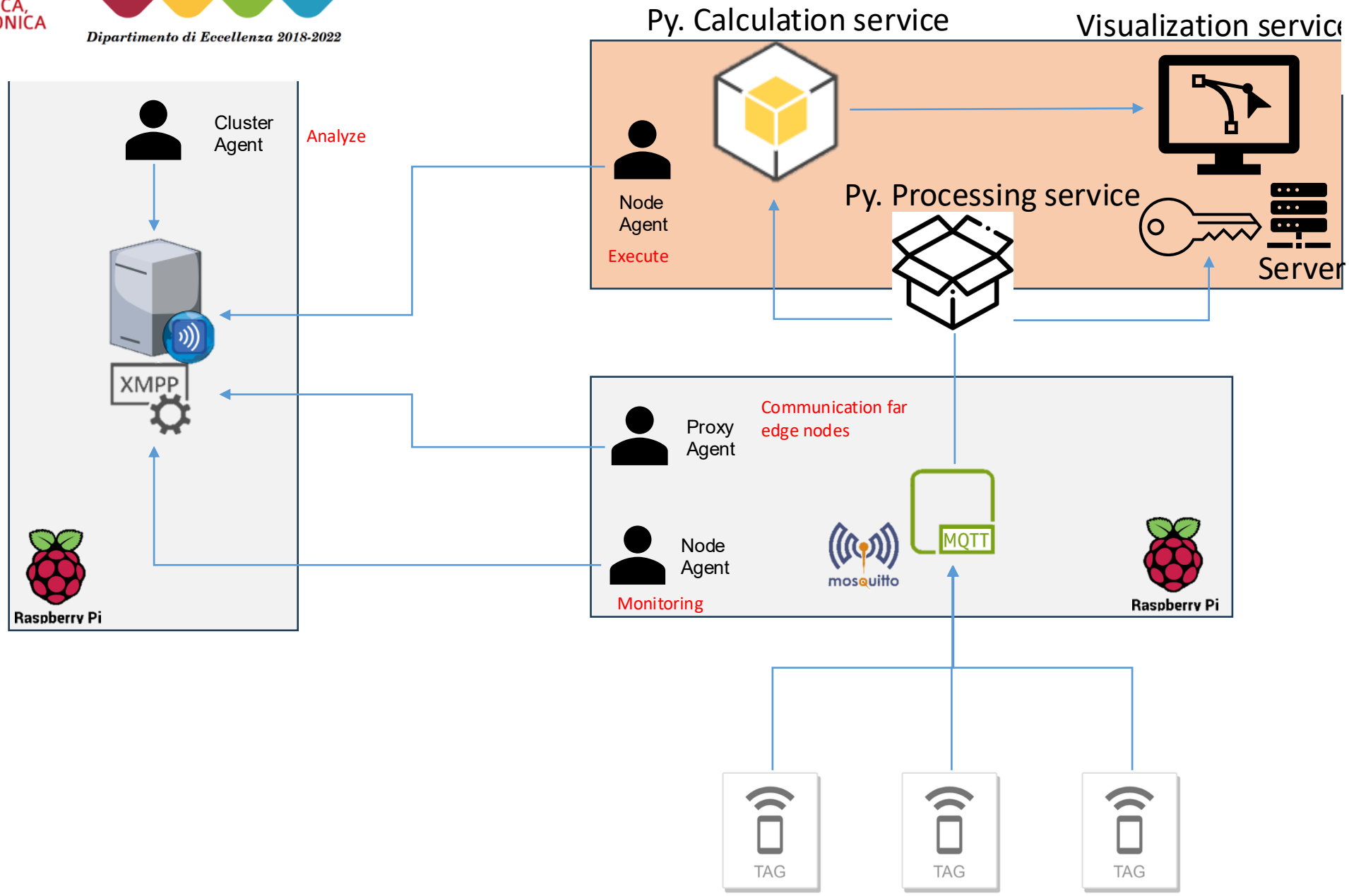
- Indoor localization system
  - anchors (radio beacons fixed in space) & tags (mobile devices)
  - tag positions are obtained through trilateration

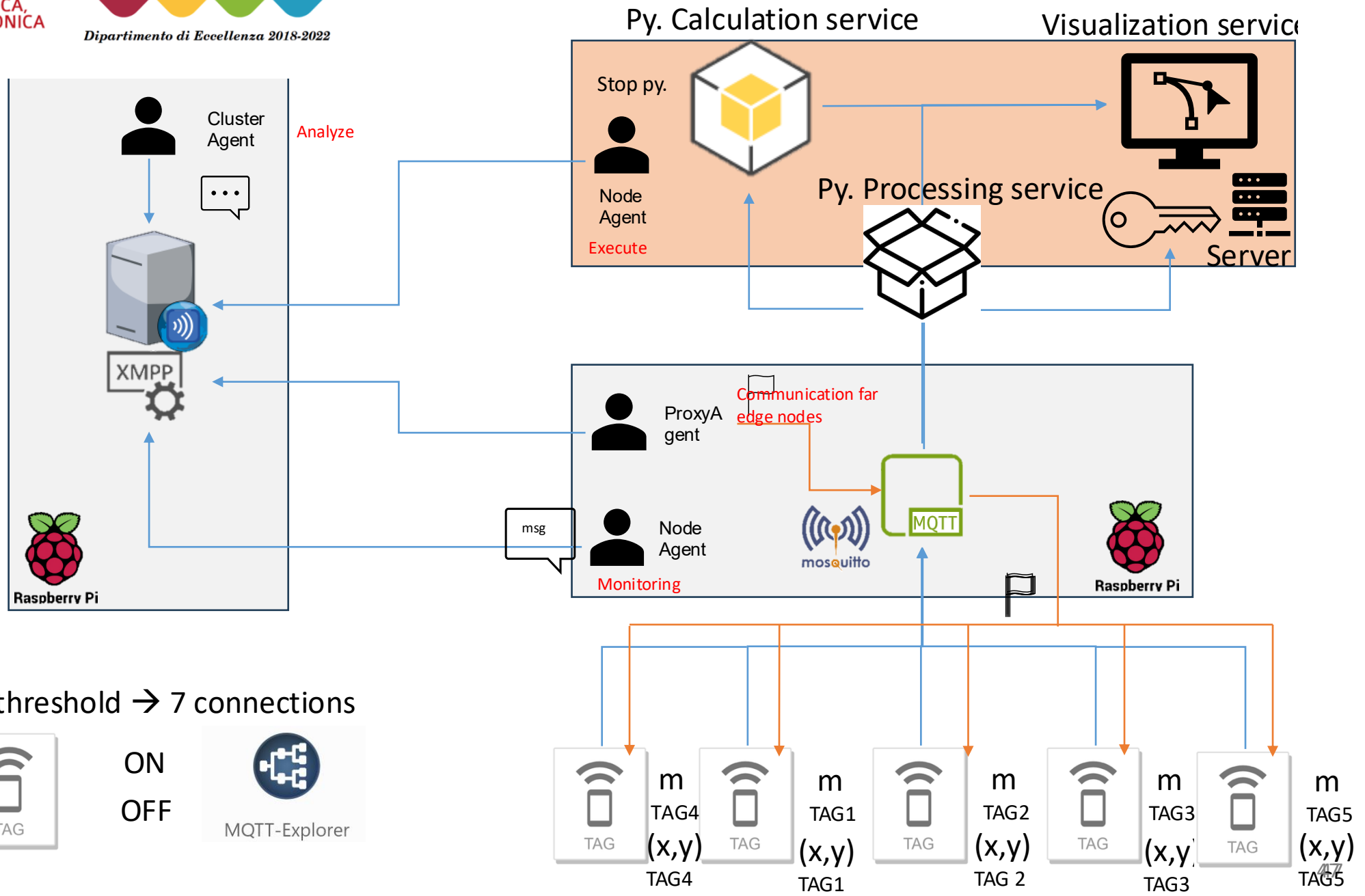
- System infrastructure
  - Level I: **IoT devices** (anchor and tags) running custom firmware
  - Level II: **Raspberry** hosting a MQTT broker for the communication between IoT devices and Server
  - Level III: **Server** hosting position calculation, visualization and data storage services

Application architecture and functionality



m = {TAG_ID, DIST_1, DIST_2, DIST_3, VOLTAGE}


Py. Calculation service
Visualization service
Cluster Agent
Analyze
Node Agent
Execute
Py. Processing service
Server
XMPP
Proxy Agent
Communication far edge nodes
Node Agent
Monitoring
mosquitto
MQTT
Raspberry Pi
Raspberry Pi
TAG
TAG
TAG



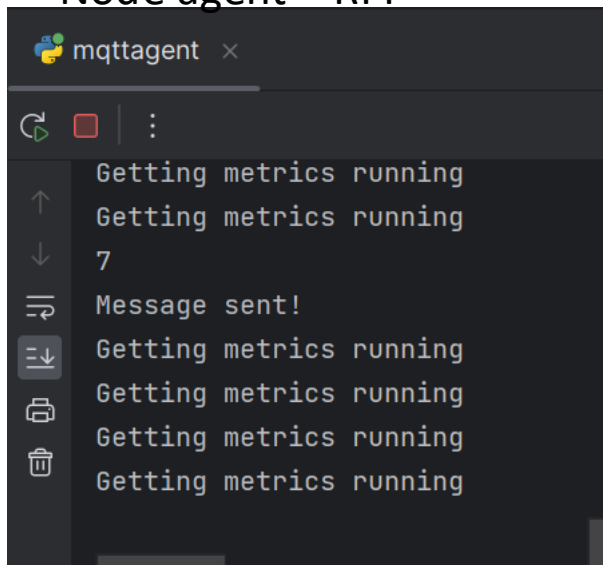
Note: Demo threshold → 7 connections

ON

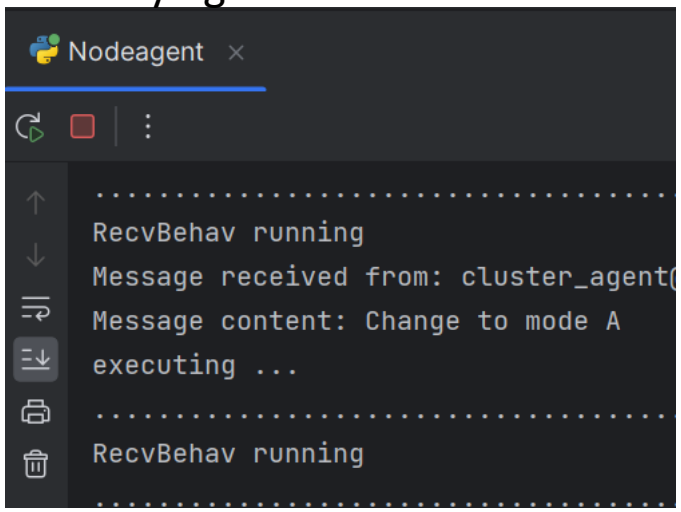
OFF

Agent messages

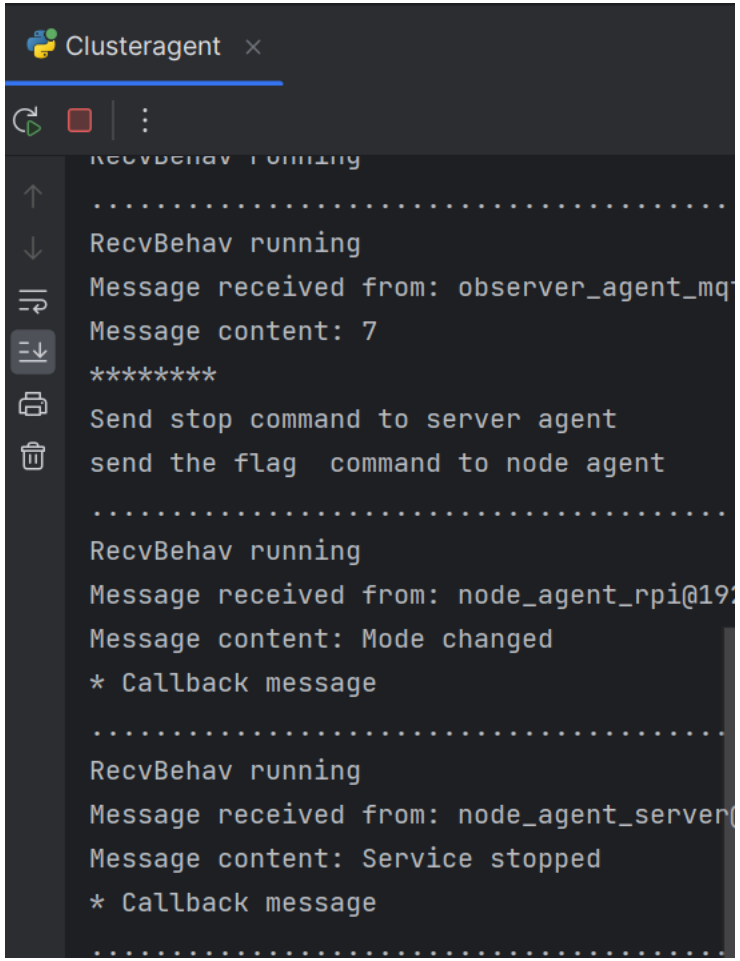
Node agent – RPI



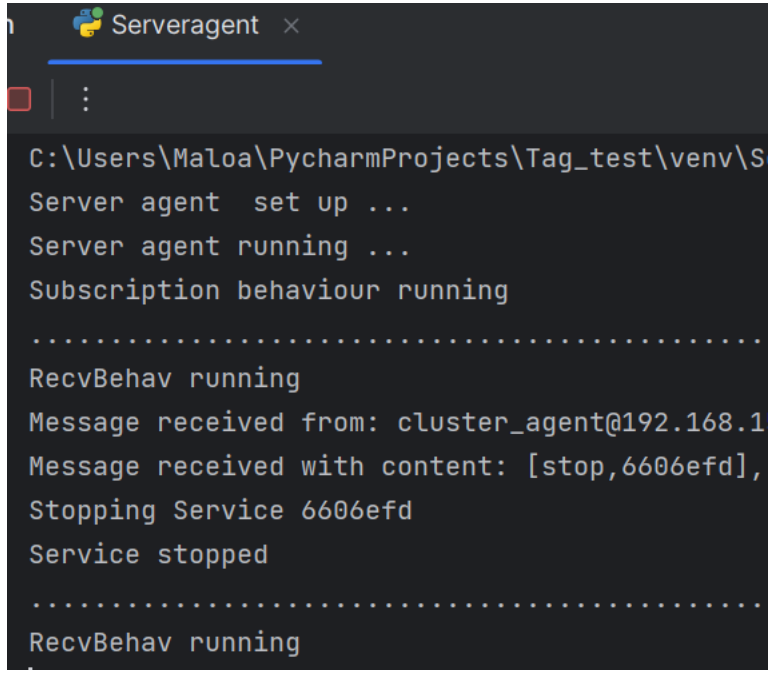
Proxy agent - RPI



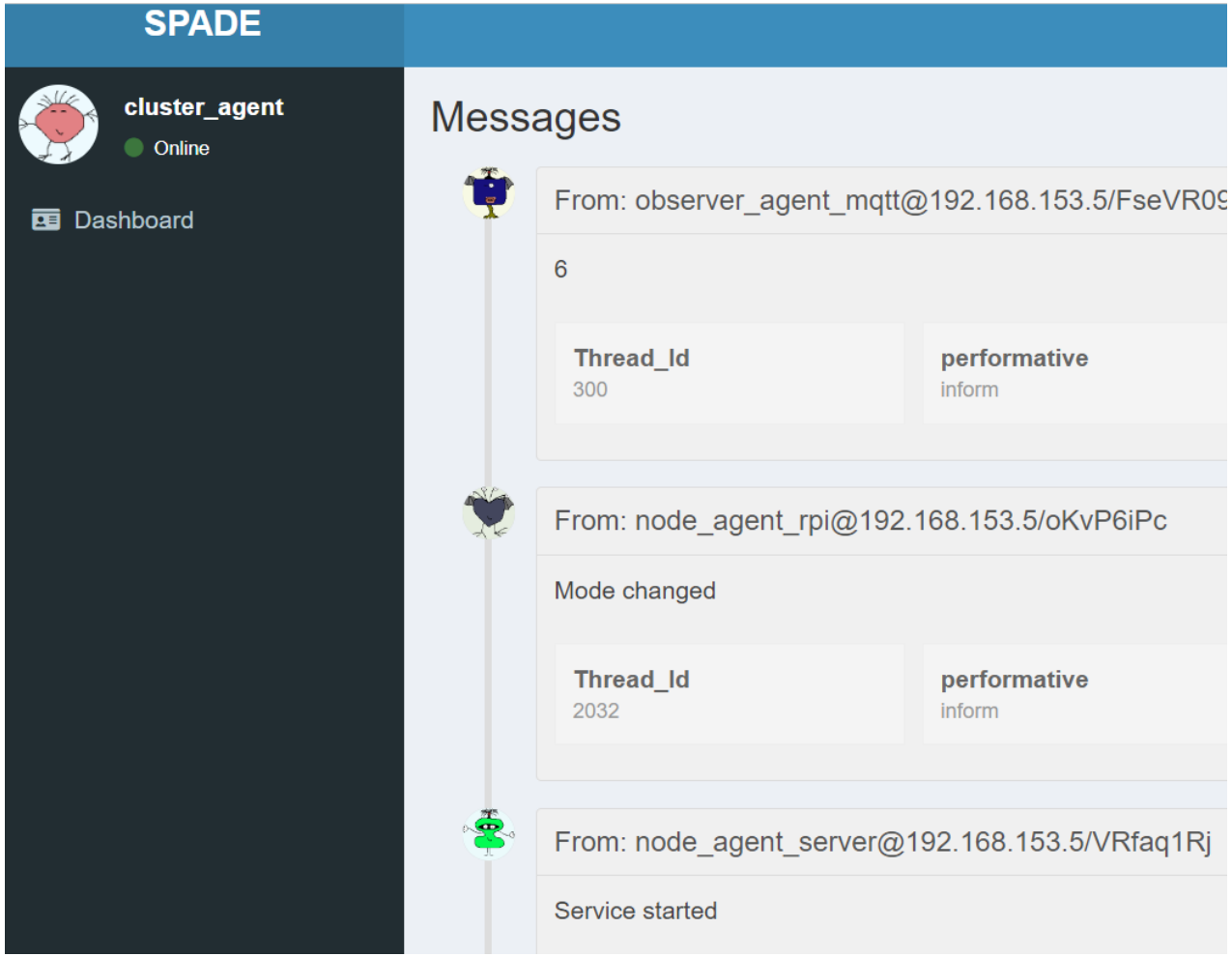
Cluster agent

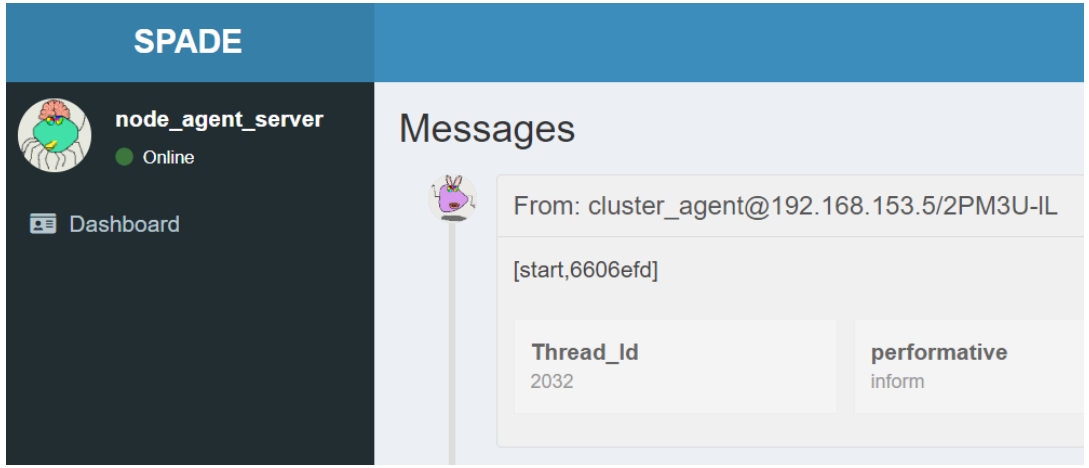


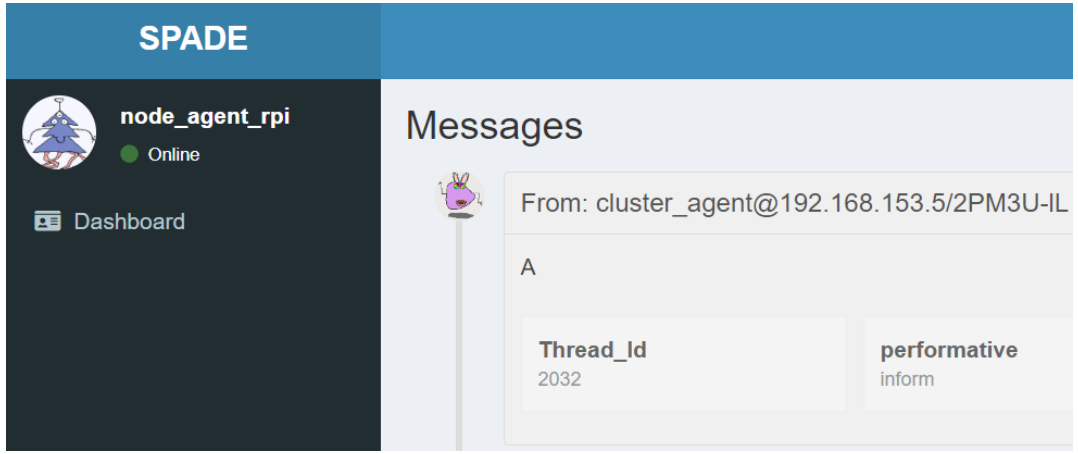
Node agent - Server



Agent graphical interface







Remarks

- Agents are able to exchange different messages between different layers of the edge-cloud continuum through XMPP service.
- As shown agents are able to monitor, analyze and execute commands to start and stop docker containerized services.
- Far edge devices can be integrated in the MAS by using the proxy agent.

Conclusions

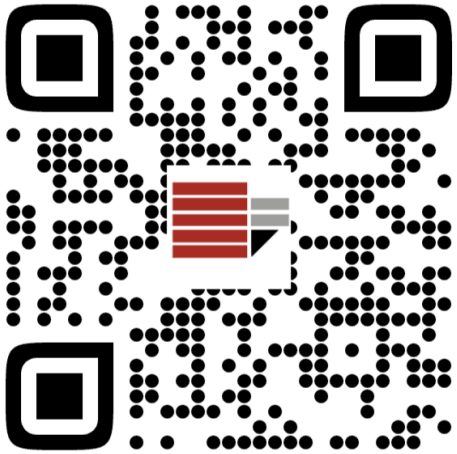
- Optimal System Management in the Device-Edge-Cloud Continuum is challenging
  - Continuous human/manual intervention is unrealistic
- AI/ML can effectively tackle this challenge
- MLSysOps proposes a concrete framework to support application deployment/orchestration and system operations in DEC continuum
- Agent paradigm is an effective programming approach to implement the ML-supported mechanisms

UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA
DIMES

Dipartimento di Eccellenza 2018-2022

THANKS!

Q&A



Funded by the European Union

MLSysOps

The research leading to these results has received funding from the European Community's Horizon Europe Programme under the MLSysOps Project, grant agreement #101092912.