# AIoTwin Orchestration Middleware

Ivan Čilić, Katarina Vuknić, Ana Petra Jukić, Ivana Podnar Žarko

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

2nd AIoTwin Summer School, September 16, 2024

Funded by the European Union

# Outline

- Artificial Intelligence of Things, AIoT
- Requirements and architecture
- Learning pipeline: adaptive orchestration of FL workflows
  - Hierarchical FL
  - FL configuration
  - Architecture and design
  - Pipeline reconfiguration
  - Framework for adaptive orchestration of FL workflows
- Hands on session: using framework for adaptive orchestration of FL workflows to run FL

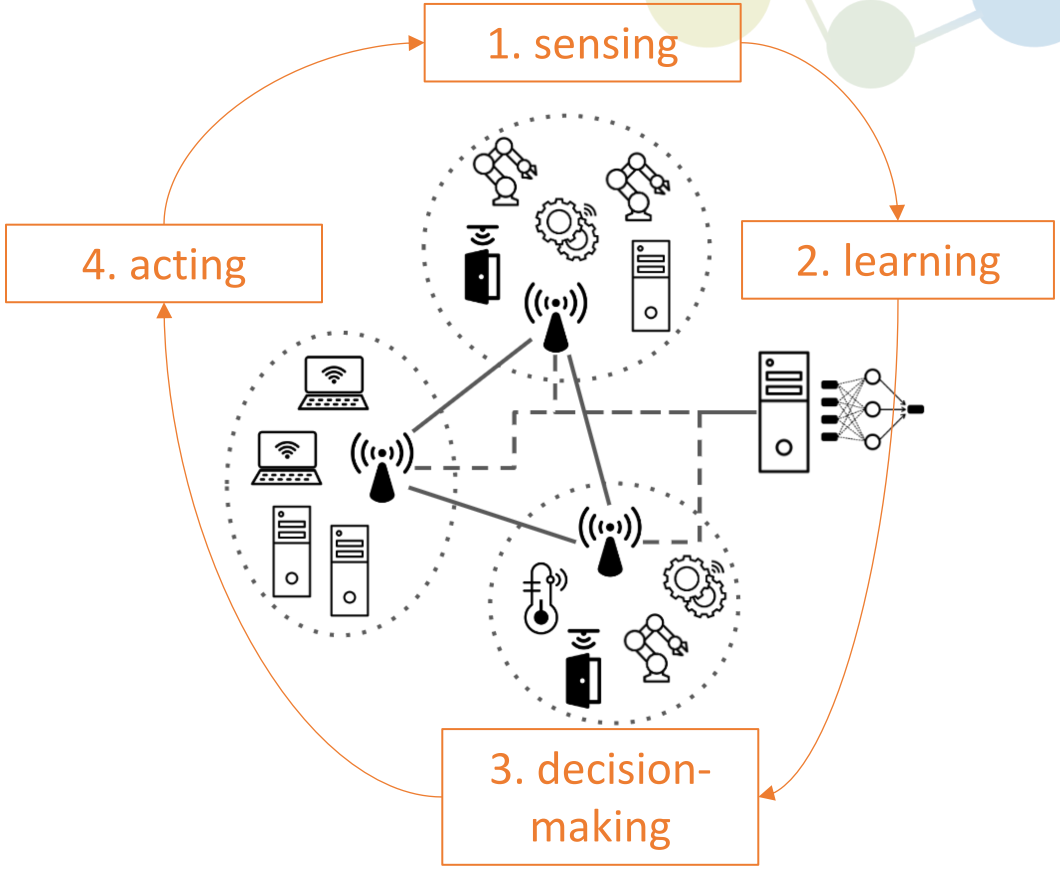
# Artificial Intelligence of Things, AIoT

Definition, example, challenges

Edge orchestration

# Artificial Intelligence of Things, AIoT

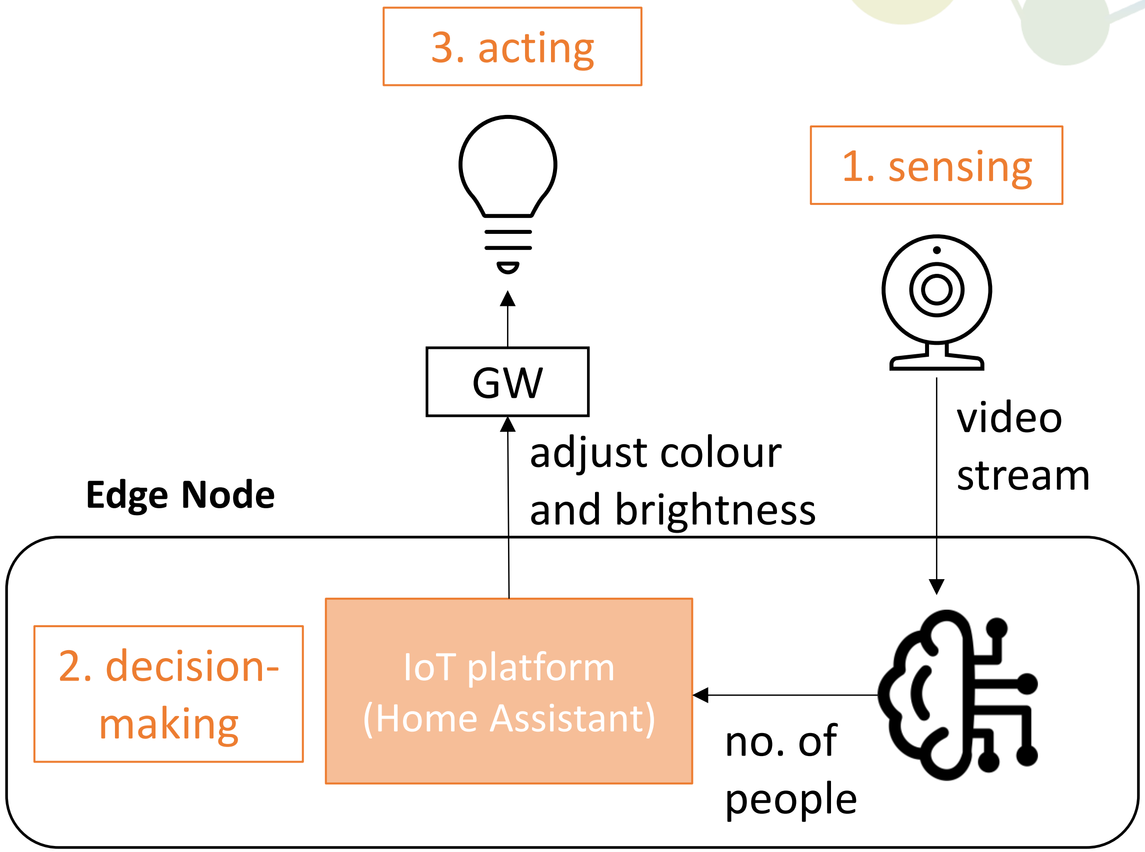
- IoT evolution
- Brings AI into smart physical spaces
- Boundaries between physical and digital world disappear
- Smart physical spaces generate large amounts of streaming data (**sensing**) for **learning**, creation of new AI models
- AI models are increasingly used in smart physical spaces, **inference**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



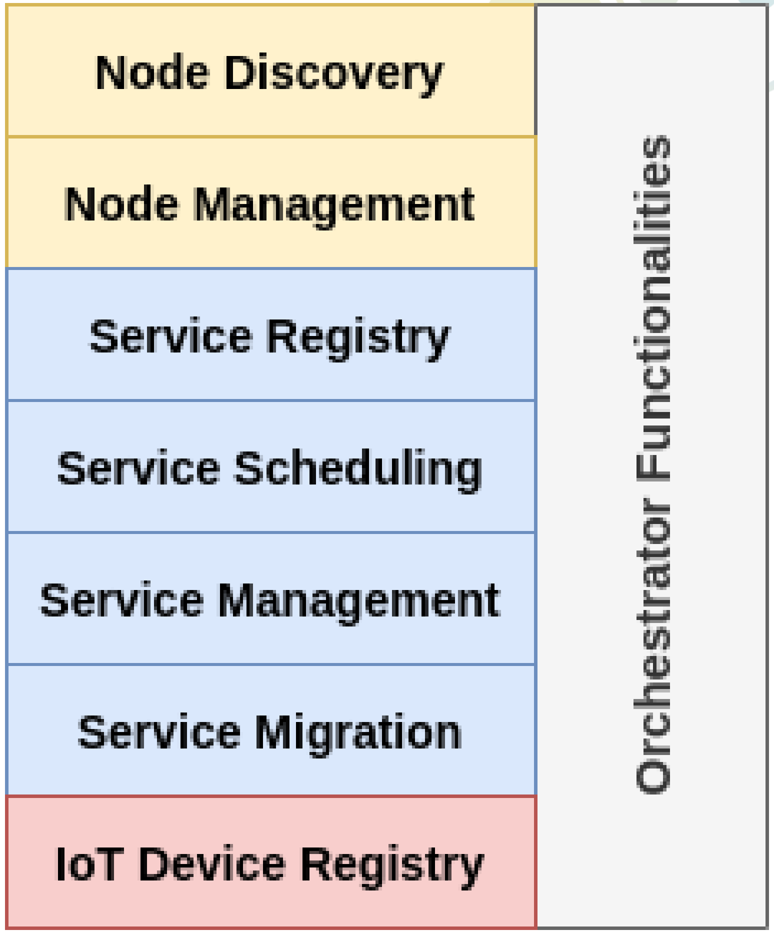
# AIoT challenges

- 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

# Edge orchestration

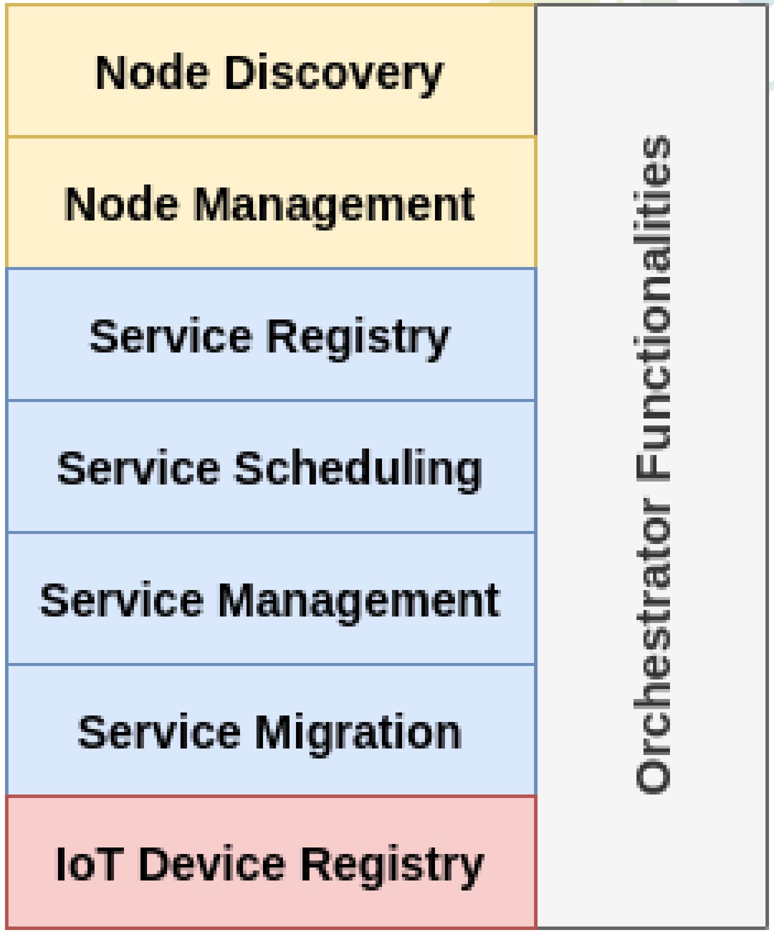
- Services running on edge nodes have to be orchestrated to ensure their high availability
  - technologies: microservices, containers, container orchestration tools
- Service orchestration is needed to
  - **schedule**
  - **deploy**
  - **manage**

  } 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



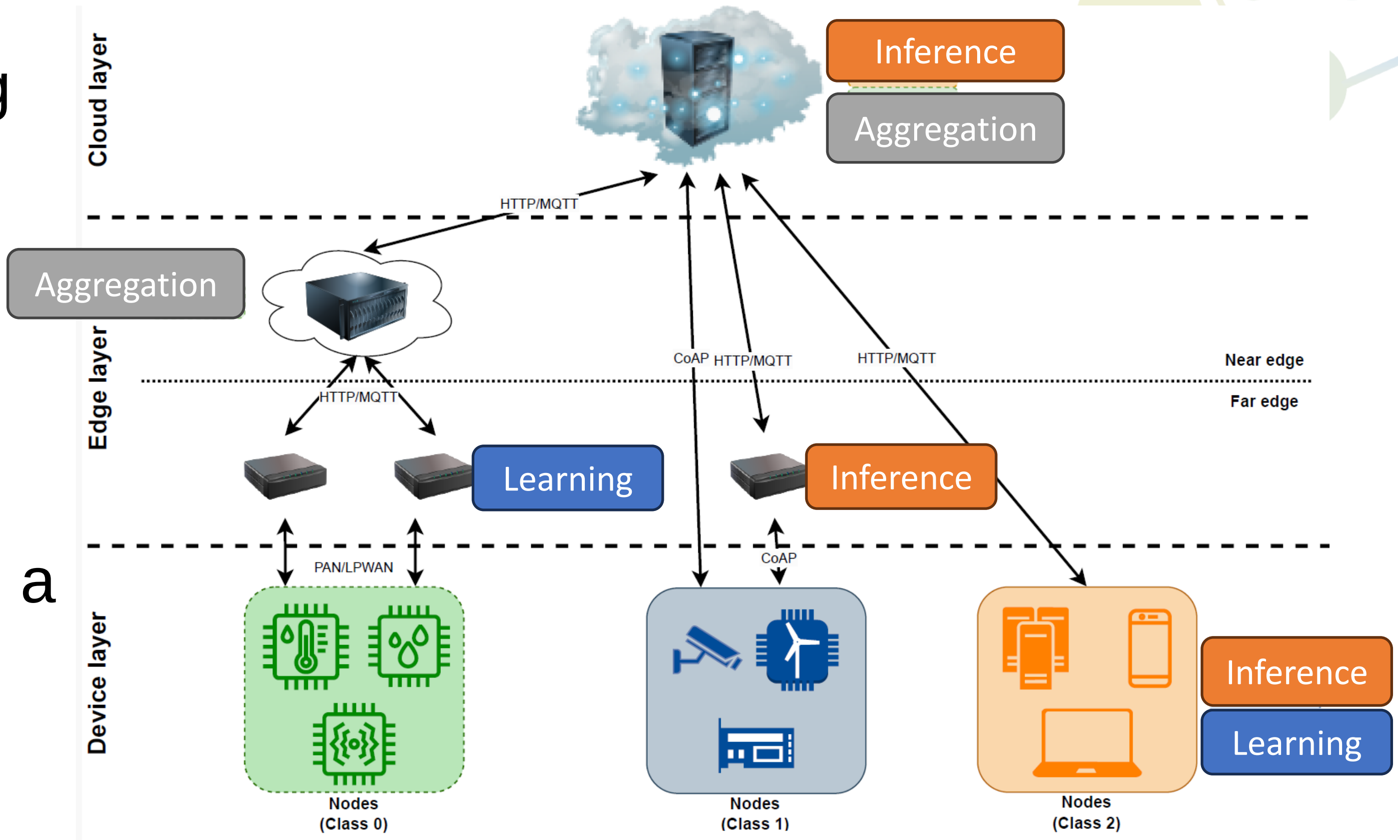
# Edge orchestration architecture

- 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



# 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?



AIoTwin

# AIoTwin Orchestration Middleware

## Requirements and Architecture

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

# Specified requirements

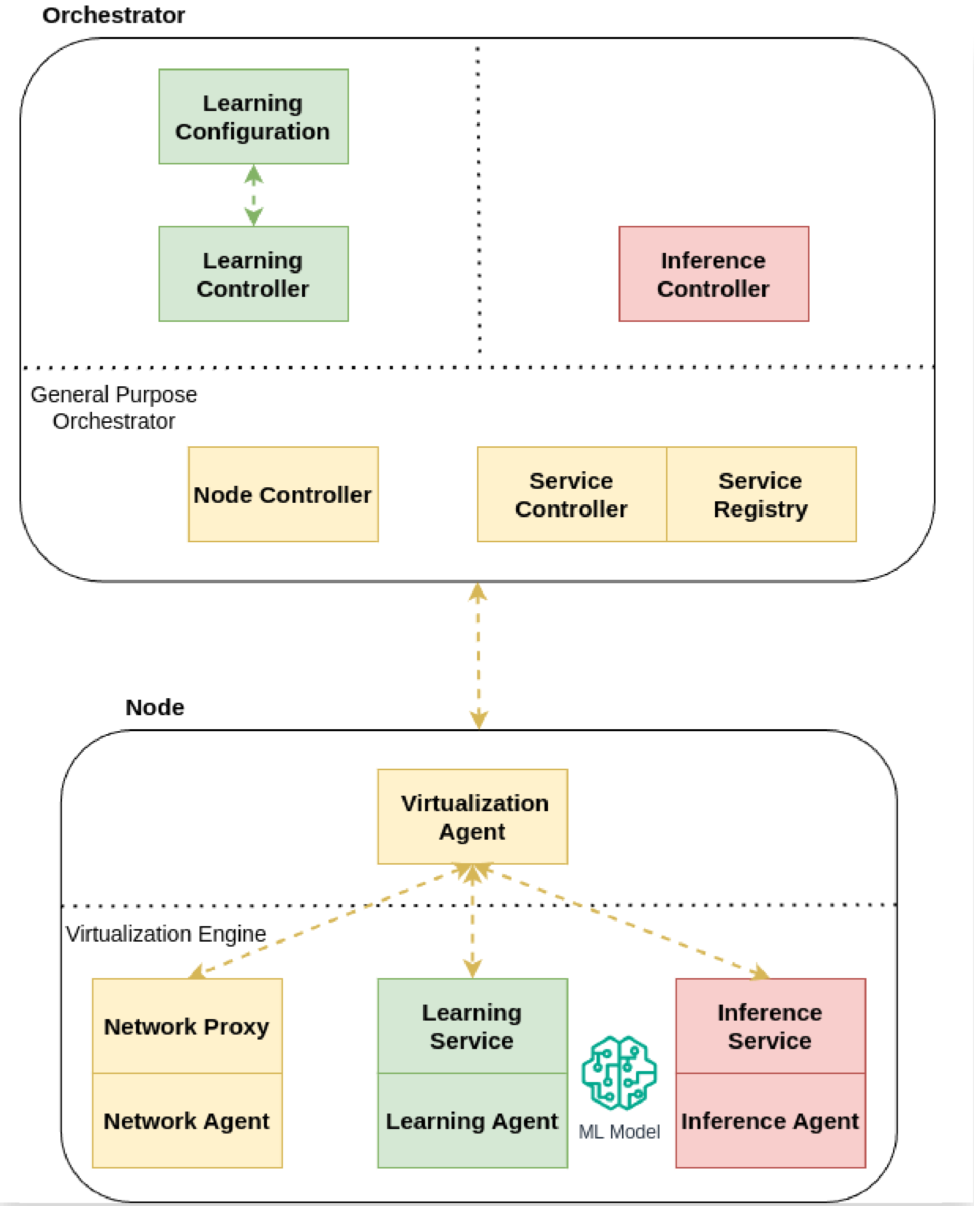
| No | Description |
|---|---|
| 1 | Efficiently manage and monitor resources on each node |
| 2 | Collect the information on the underlying network connecting nodes in the continuum |
| 3 | Deploy and manage services across the continuum |
| 4 | Collect the distribution of data available on node for ML |
| 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 |

GPO ● FL-related ● Inference-related

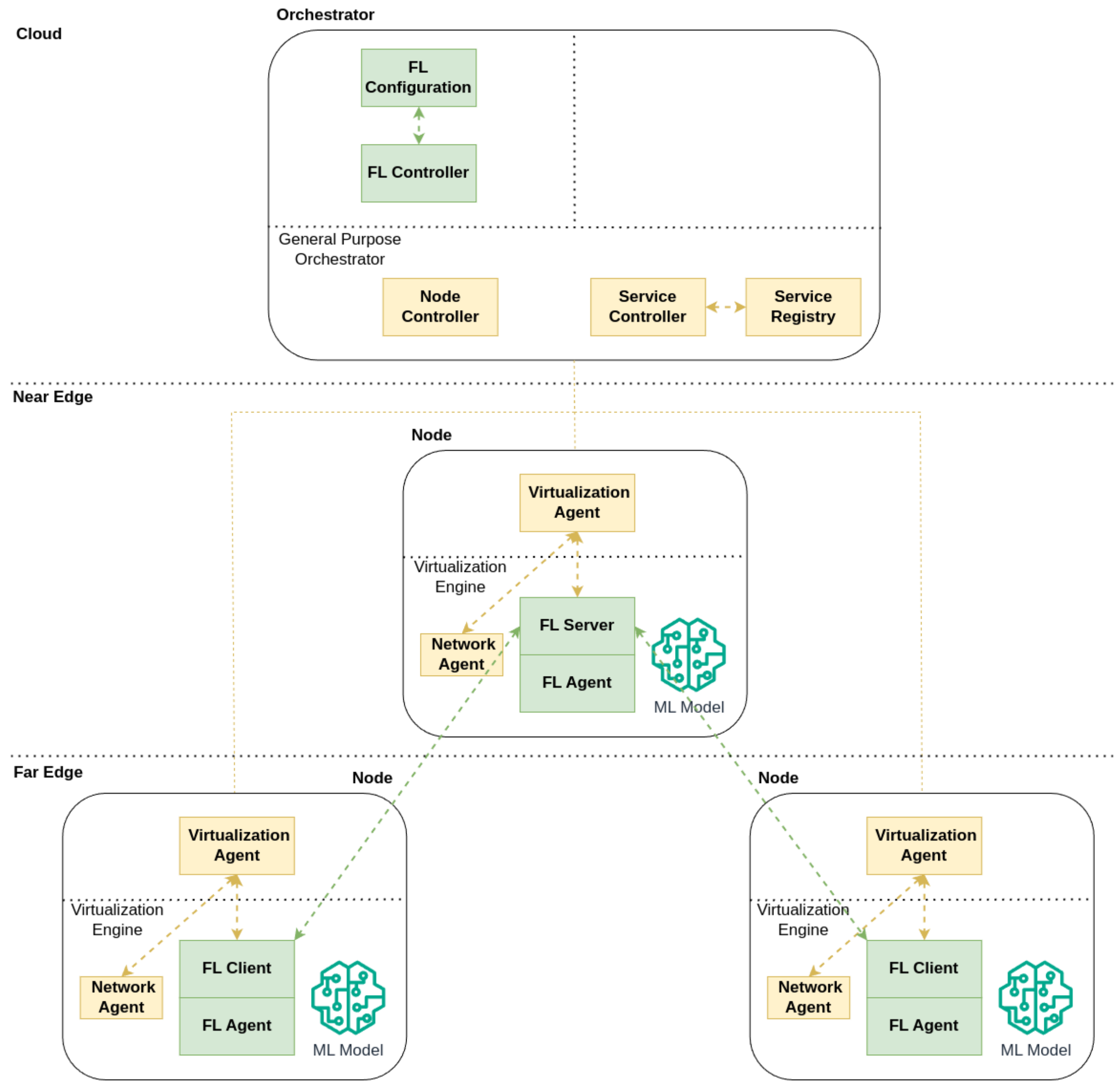
# General Architecture for Orchestration of ML Pipelines

- 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



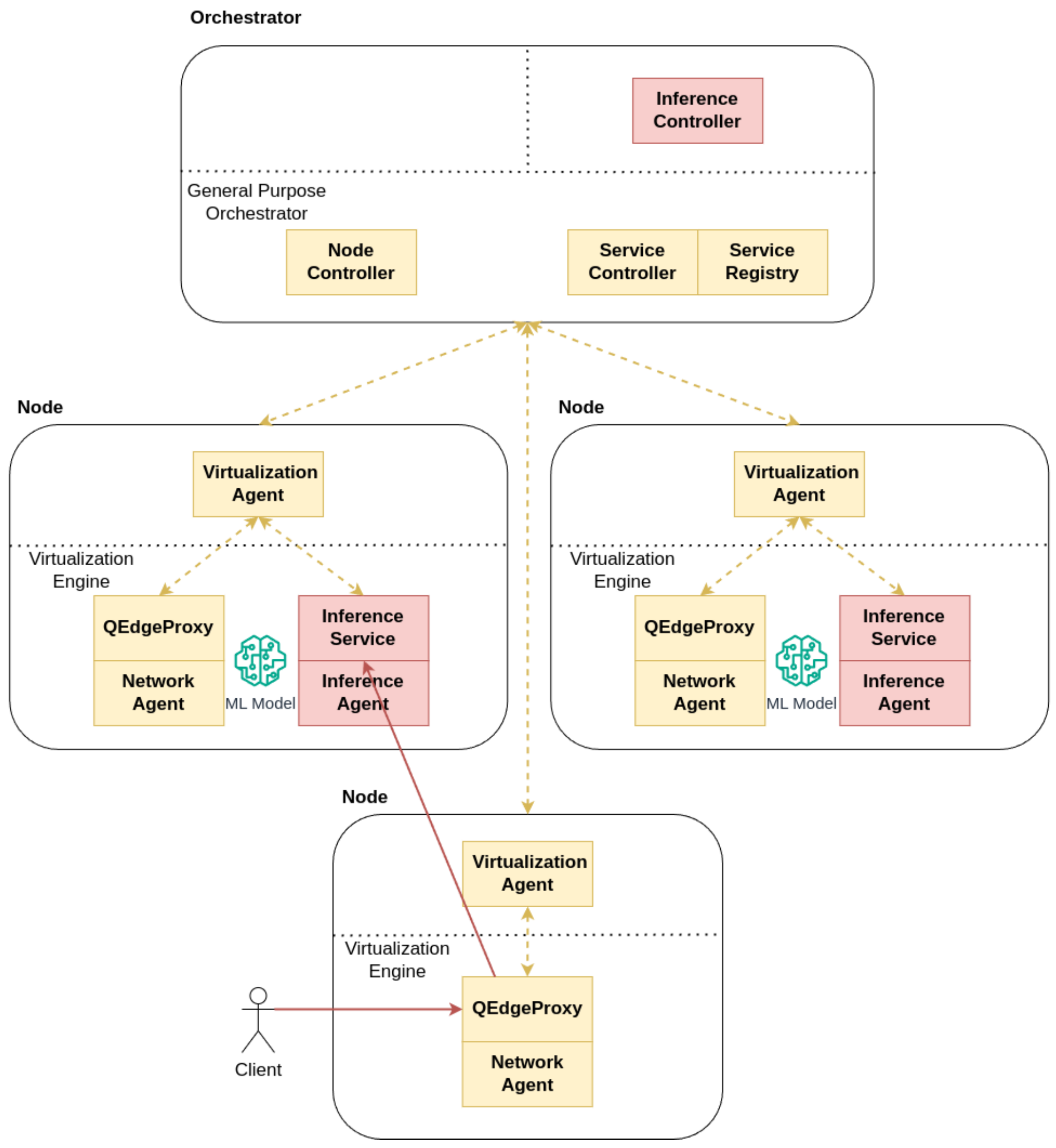
# Adaptive orchestration of **FL pipelines**: the architecture

- 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



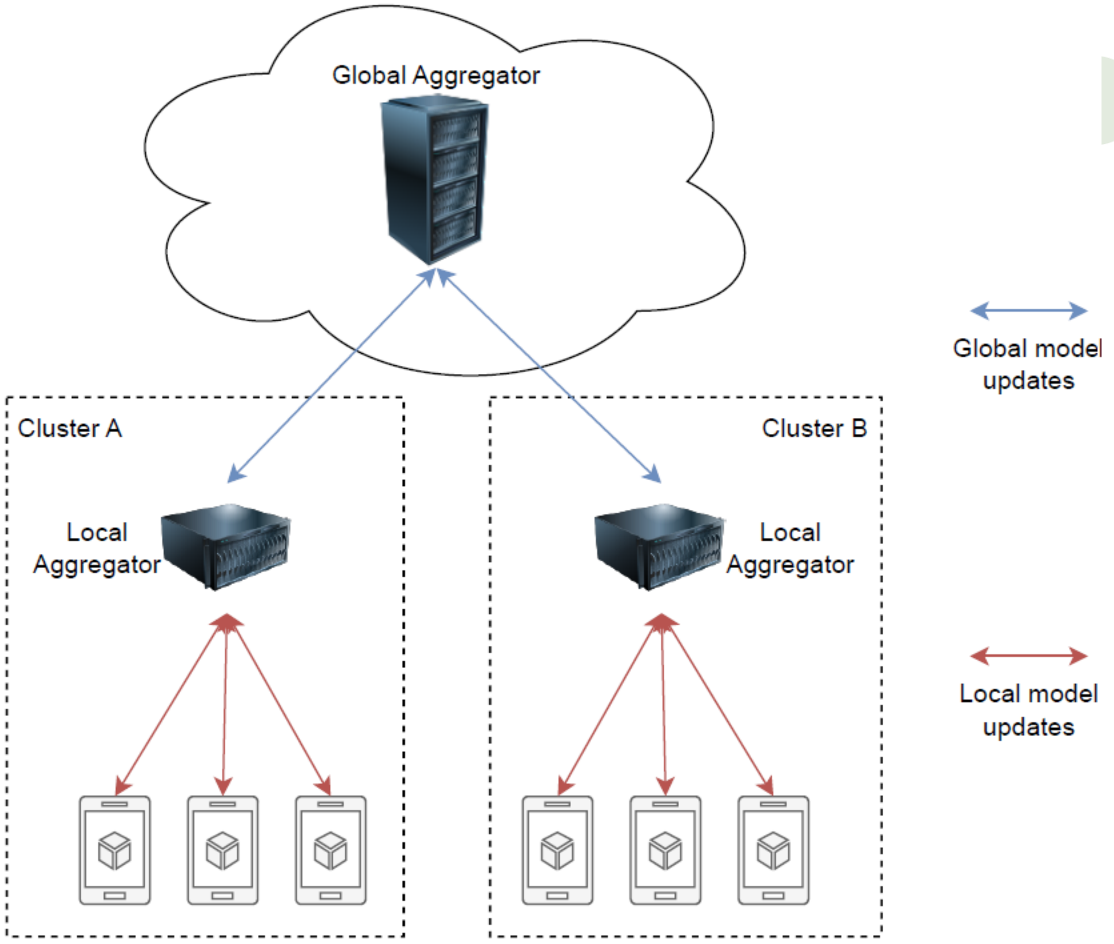
# AIoTwin Orchestration Middleware

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.

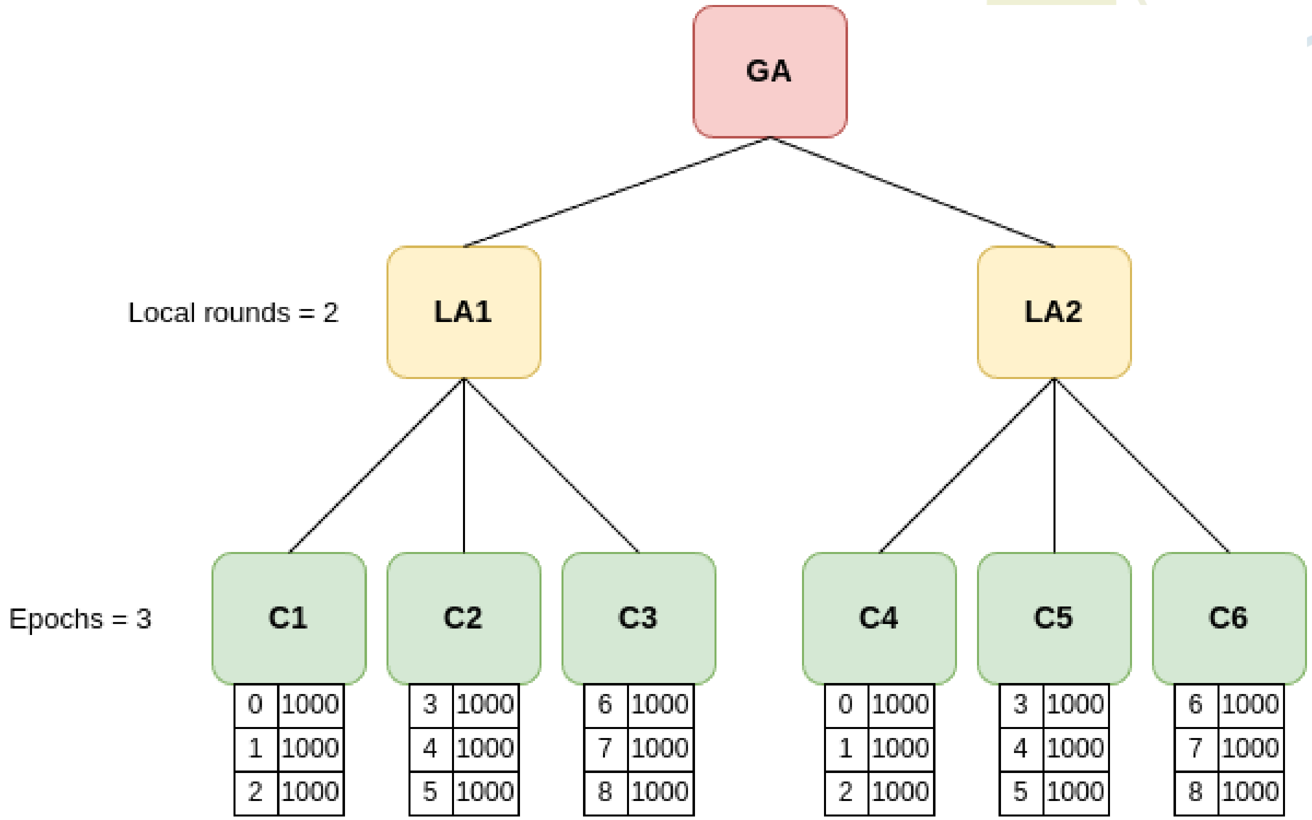
# Hierarchical Federated Learning

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



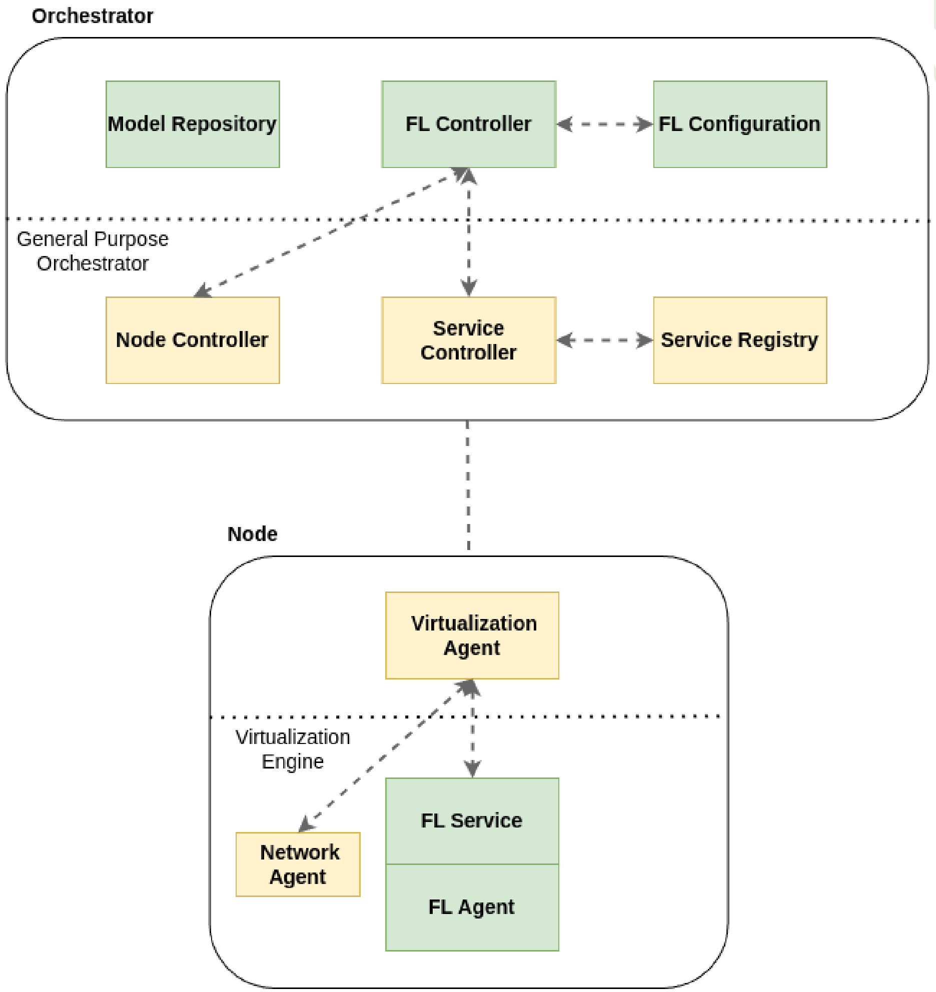
# Hierarchical FL configuration

- How should we organize clients into clusters?
  - Data distribution
  - Communication costs
- Aggregation configuration?
  - Aggregation algorithm
  - Aggregation frequency
  - Synchronous vs asynchronous

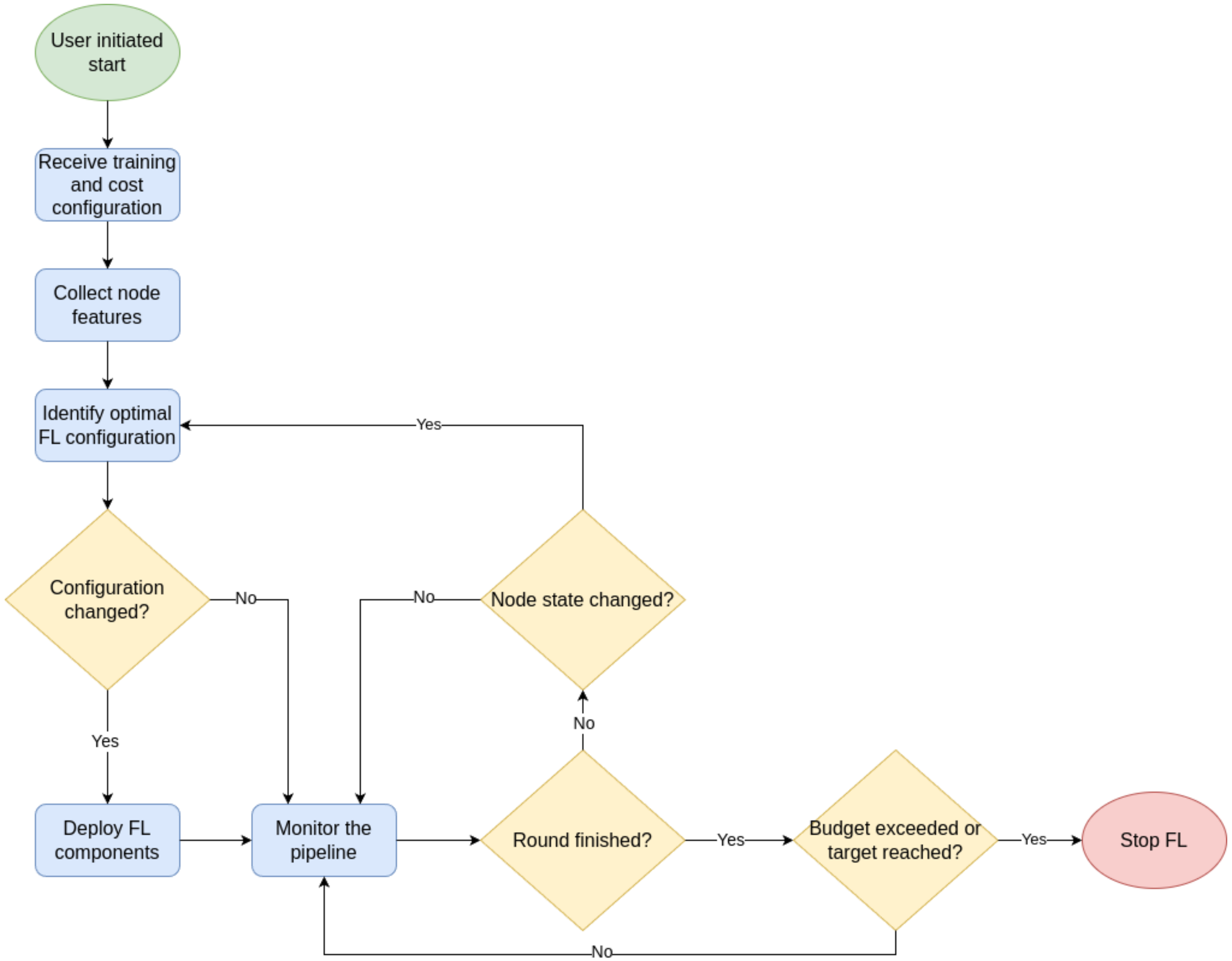


# Architecture for adaptive orchestration of FL workflows

- Dynamic edge environment
- An adaptive orchestration mechanism is needed to
  - deploy the entities of the FL pipeline (clients, local/global aggregators),
  - monitor the execution of the pipeline, and
  - perform reconfiguration when needed.



# Orchestration workflow



# Orchestration workflow: steps 1/2

1. Receive training and cost configuration
   - Training configuration
     - ML model, training parameters (batch size, learning rate…)
   - Cost configuration
     - Cost can be expressed in terms of communication, computation, time, or energy
     - Two cost configuration types
       - Total available budget
       - Minimize cost to reach target accuracy
2. Collect node features
   - Infrastructure-specific features
     - Node resources and underlying network
   - FL-specific features
     - Node role (client, local/global aggregator)
     - If node is a client:
       - Data distribution
       - Historical training behavior (training time, resources used during training)

# Orchestration workflow: steps 2/2

3. Identify optimal FL configuration
    - Configuration output: cluster organization, aggregation frequency…
    - Orchestration is independent of the configuration strategy
        - For example: clustering to minimize communication cost with tradeoff to data balancing [1]
4. Deploy FL components
    - Nodes download the FL services and FL pipeline starts
5. Monitor the pipeline
    - Infrastructure monitoring
        - Node states and their resources, network state, etc.
    - FL performance monitoring
        - Accuracy, loss, etc.
    - Cost monitoring

[1] Y. Deng et al., "Share: Shaping data distribution at edge for communication-efficient hierarchical federated learning", ICDCS 2021

AIoTwin

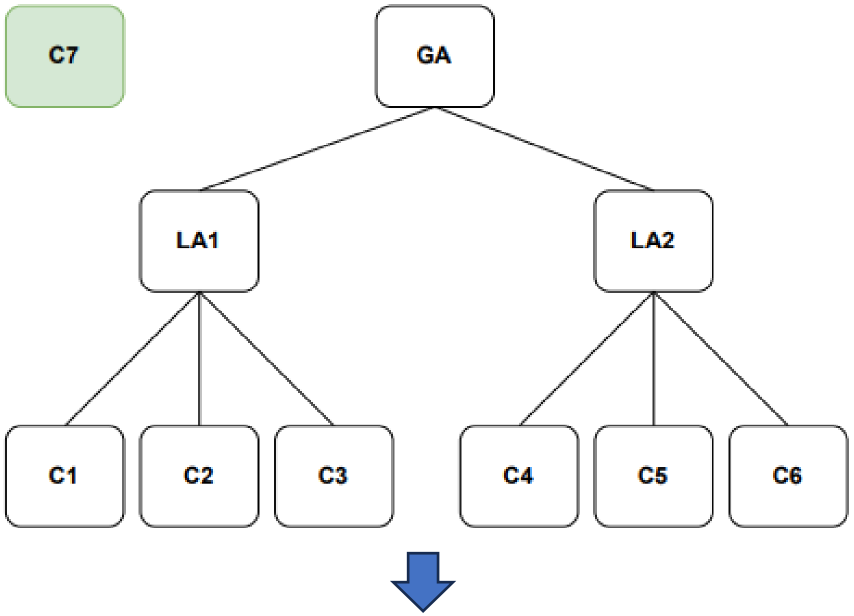
# AIoTwin Orchestration Middleware

## Adaptive orchestration of federated learning workflows: Reconfiguration

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.

# Pipeline reconfiguration

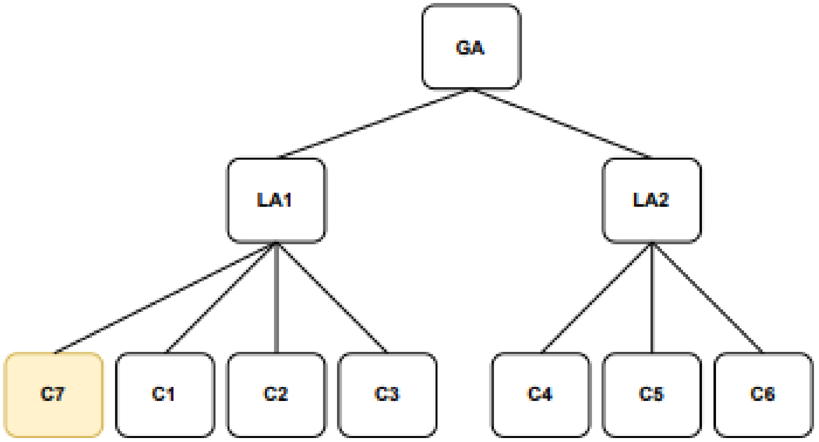
- Key characteristic of adaptive orchestration: **dynamically adjusting to changes during the FL runtime**
  - Adjustment = reconfiguration
- Reconfiguration triggers
  - Reactive: upon the occurrence of an event (e.g. node left)
  - Proactive: before the occurrence of an event (e.g. node is predicted to become overloaded)
- Reconfiguration steps
  1. Identify new optimal configuration
  2. Identify the differences between new and current configuration to define **reconfiguration changes** ($\Delta R$)
  3. Apply changes to the FL pipeline



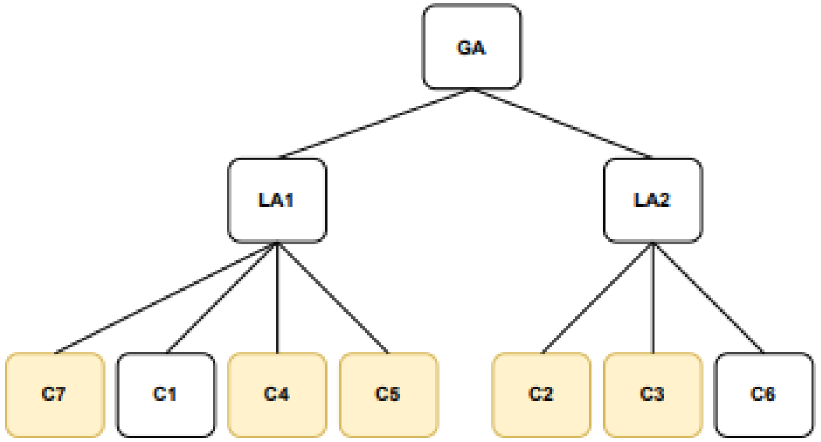

ΔR = 1

ΔR = 5

Configuration A



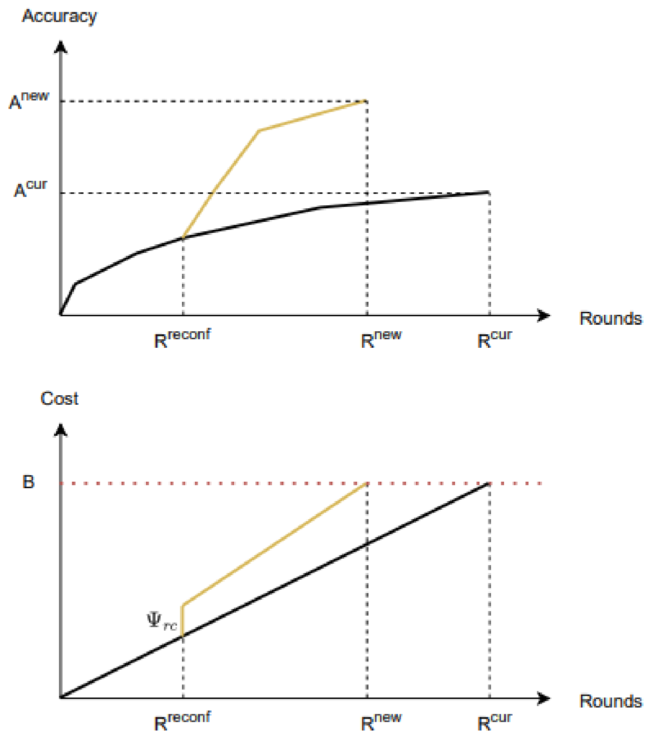

Configuration B



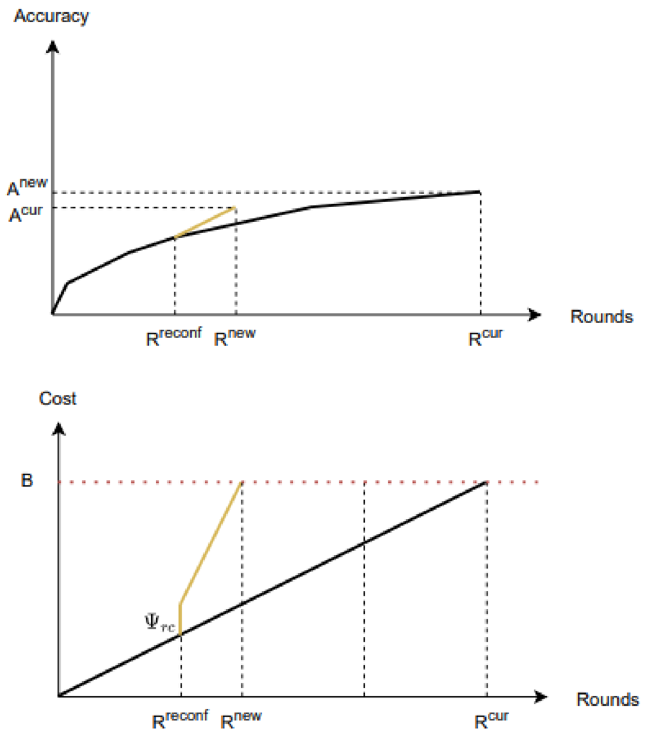
# Reconfiguration cost

- Reconfiguration comes with a cost $\Psi_{rec}$ that can be expressed with two parameters:
  - Reconfiguration change cost $\Psi_{rc}$
    - Cost for applying all reconfiguration changes
    - $\Psi_{rc} = \sum_{i=1}^{\Delta R} \psi_{rc}(i), \Psi_{rc} \geq 0$
  - Post reconfiguration cost $\Psi_{pr}$
    - Difference of cost per global round between new and current configuration
    - $\Psi_{pr} = \Psi_{gr}^{new} - \Psi_{gr}^{cur} = \Delta\Psi_{gr}, \Psi_{pr} \in (-\infty, +\infty)$

# Reconfiguration decision: communication budget

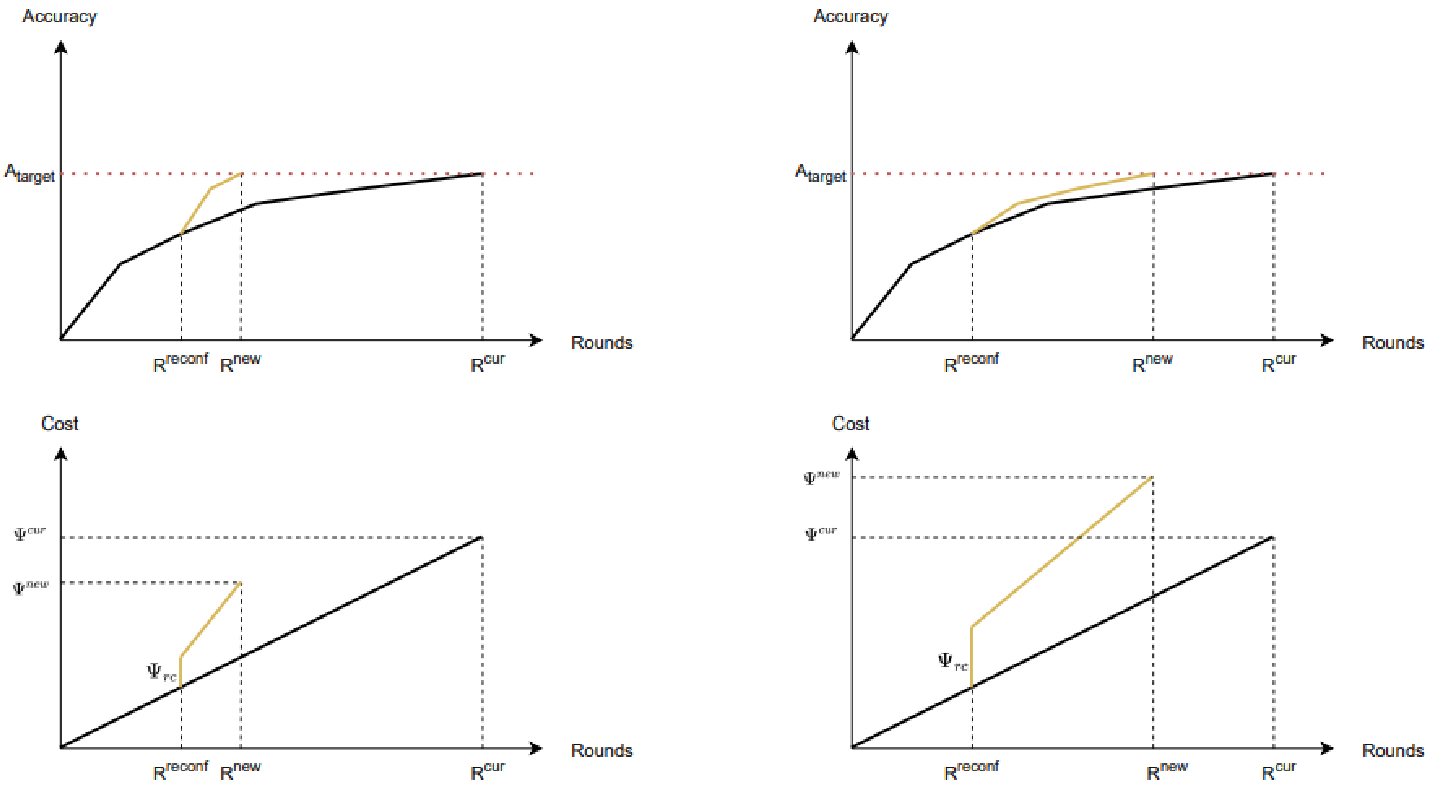



(a) Scenario A




(b) Scenario B

# Reconfiguration decision: cost minimization

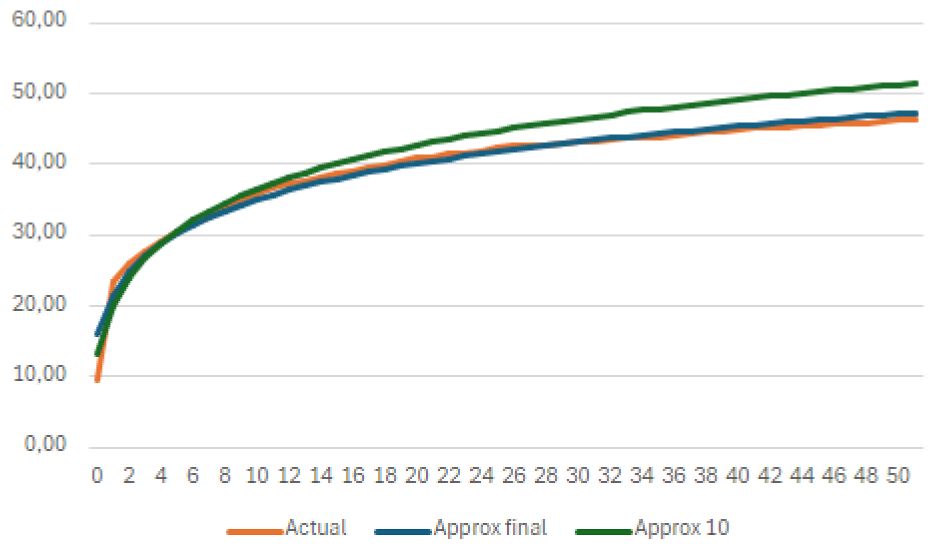



(a) Scenario A (b) Scenario B

AIoTwin

# Reconfiguration decision: proactive approach

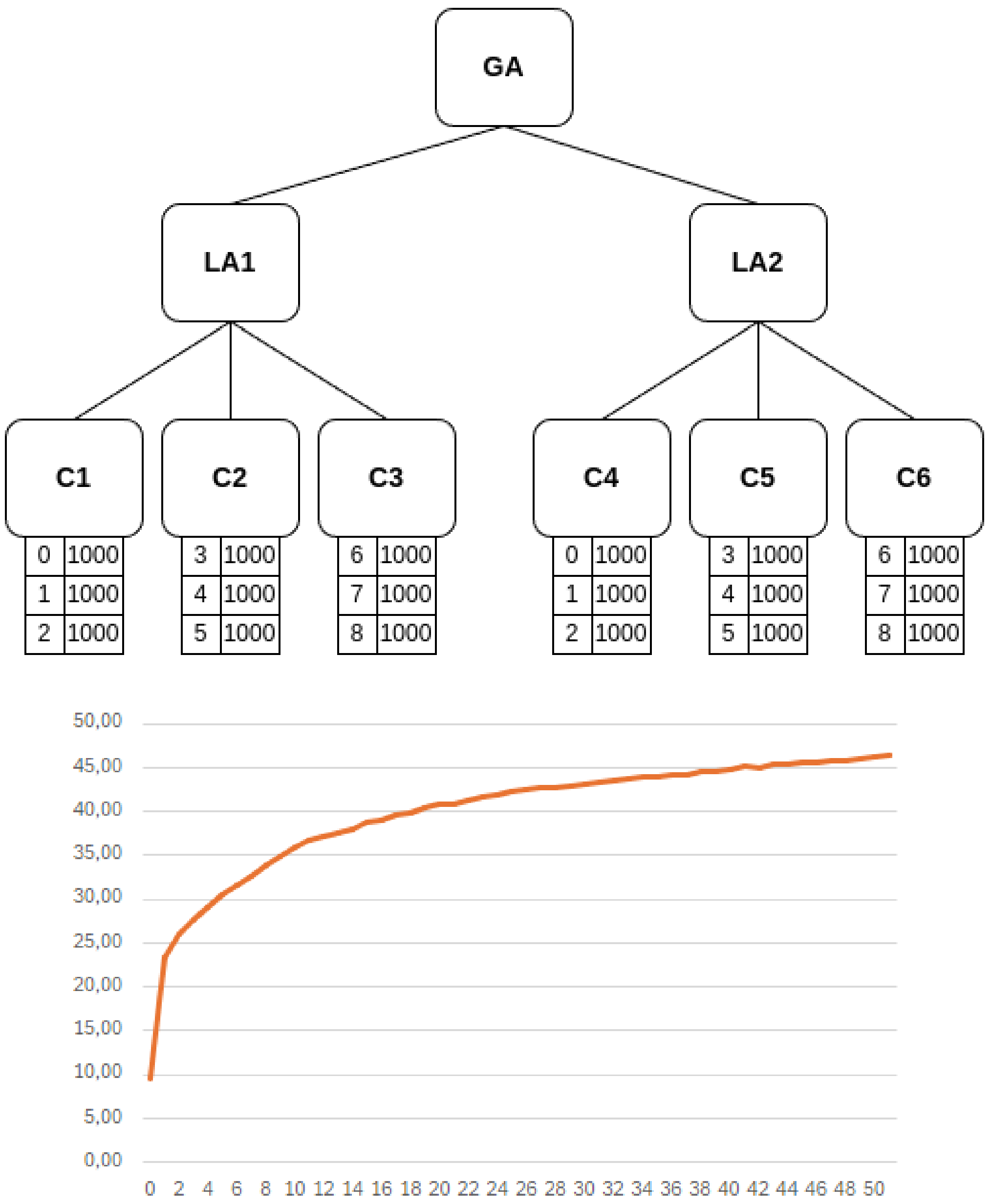
- Several methods to calculate node utility [2]:
  - Data sample-based utility measurement — $|D_i|$
    - Can be calculated before training
  - Model-based utility measurement — $||w_i^{after} - w_i^{before}||_2$
    - Can be calculated only after some training epochs
- Our tested approach:
  1. Calculate reconfiguration cost and get remaining rounds with new configuration
  2. Calculate function that described performance trend (regression)
  3. Calculate node utility from the data distribution
  4. Reconfigure if performance improvement is predicted

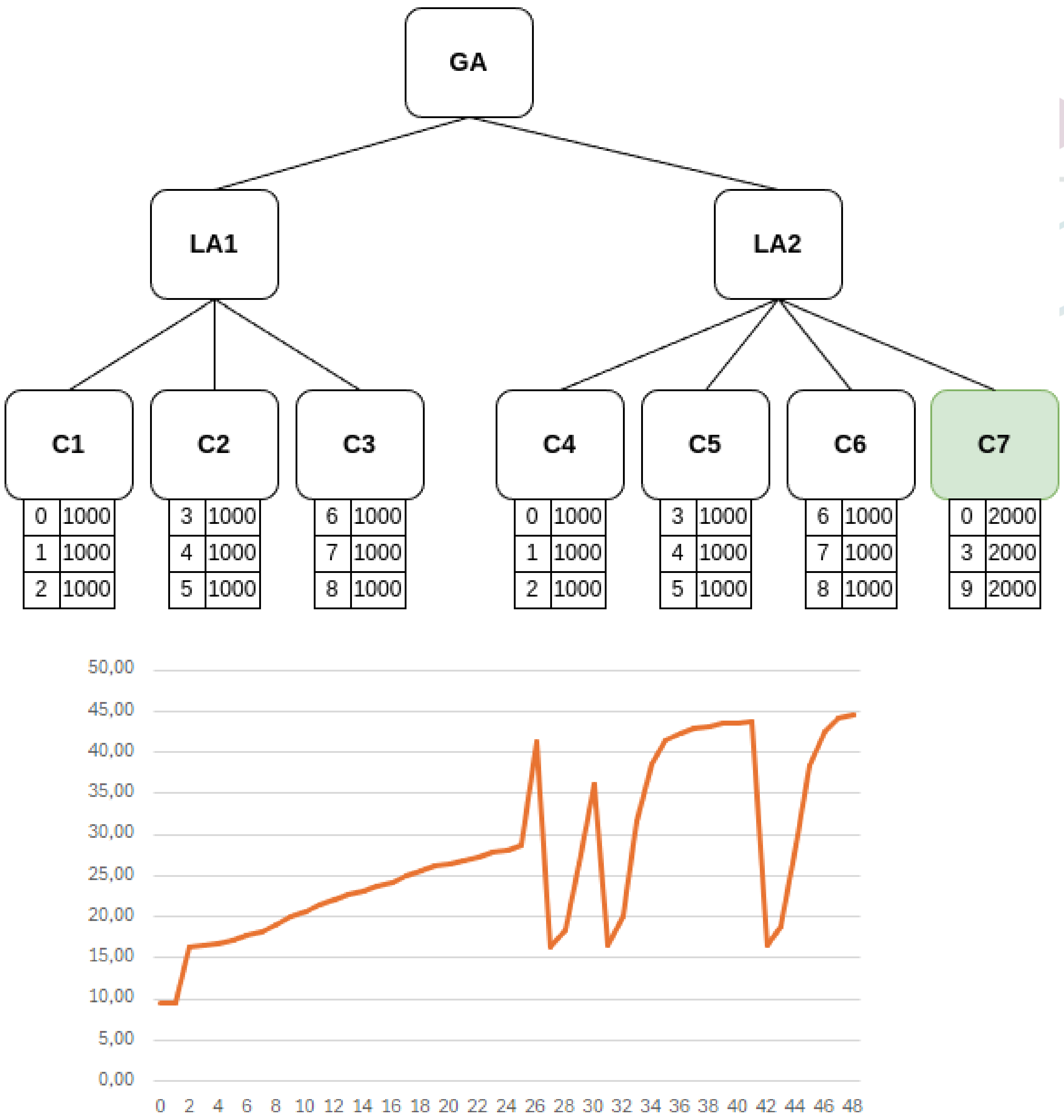



[2] L. Fu et al., "Client Selection in Federated Learning: Principles, Challenges, and Opportunities", IEEE Internet of Things Journal, 2023

# Proactive approach problems

- Various factors, besides the dataset size or unseen class data, affect the performance when introducing a new node
- Obtaining data distribution might violate privacy requirements
- Adding a new node can even introduce performance degradation because
  - New clusters are imbalanced
  - Model overfits to the new data (or unseen class)
  - Classes in the new node's dataset may be similar or completely different
    - So we need the information not only about the number of classes but also their characteristics
- Conclusion
  - Too many parameters that are hard to generalize to support different models and datasets

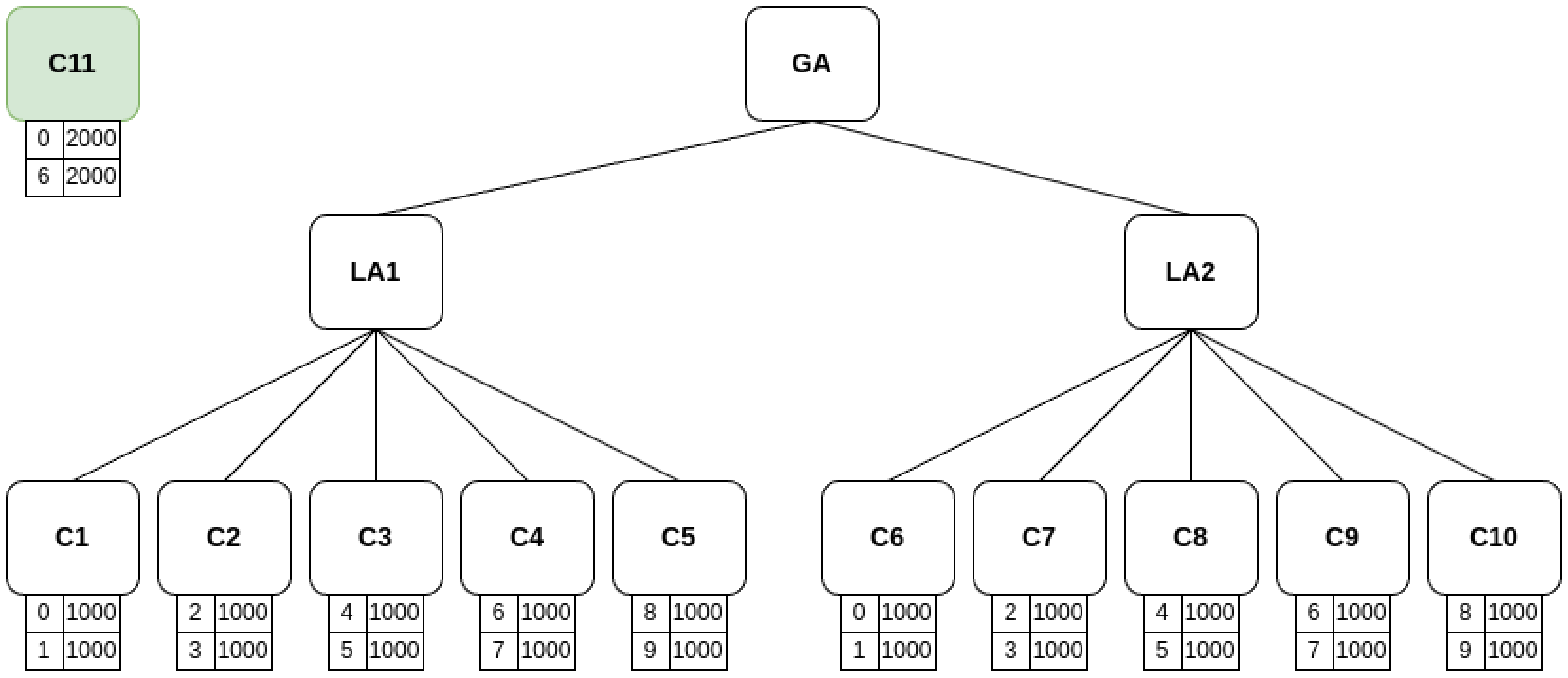

GA
LA1
LA2
C1
C2
C3
C4
C5
C6
0 1000
1 1000
2 1000
3 1000
4 1000
5 1000
6 1000
7 1000
8 1000
0 1000
1 1000
2 1000
3 1000
4 1000
5 1000
6 1000
7 1000
8 1000
50,00
45,00
40,00
35,00
30,00
25,00
20,00
15,00
10,00
5,00
0,00
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50

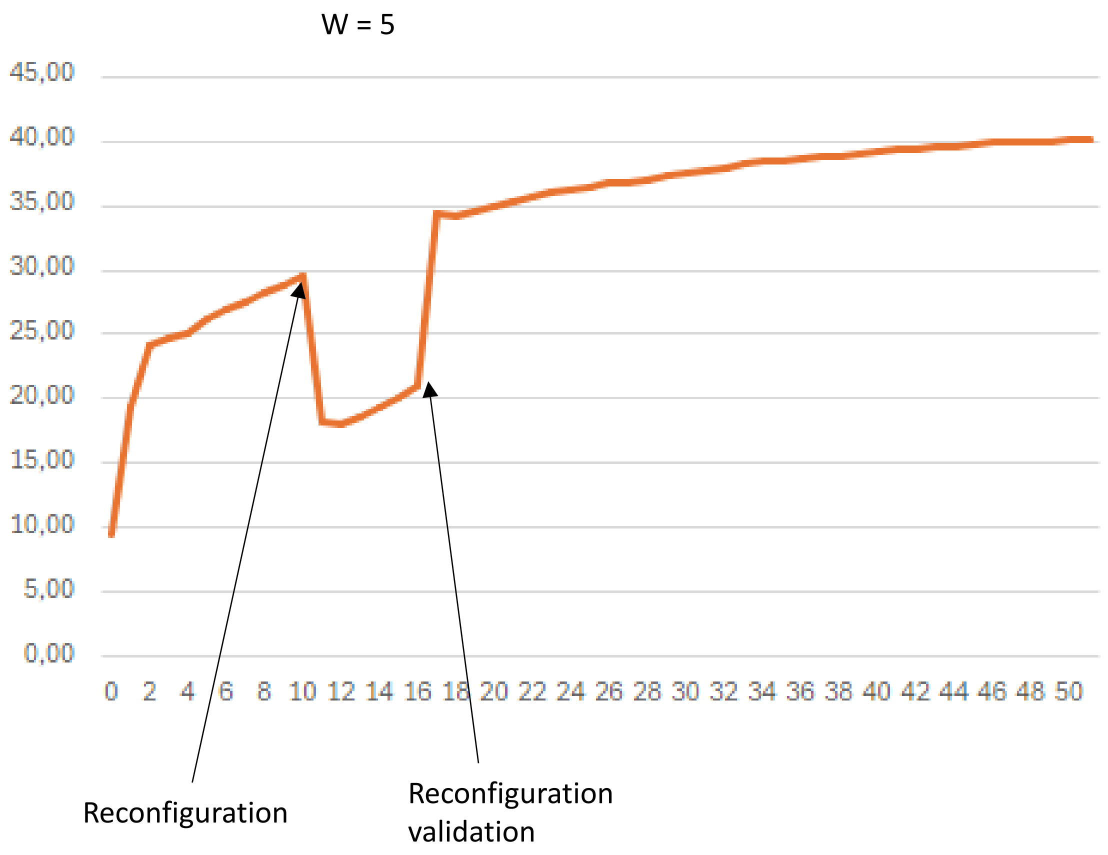

GA
LA1
LA2
C1
C2
C3
C4
C5
C6
C7
0 1000
1 1000
2 1000
3 1000
4 1000
5 1000
6 1000
7 1000
8 1000
0 1000
1 1000
2 1000
3 1000
4 1000
5 1000
6 1000
7 1000
8 1000
0 2000
3 2000
9 2000
50,00
45,00
40,00
35,00
30,00
25,00
20,00
15,00
10,00
5,00
0,00
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48

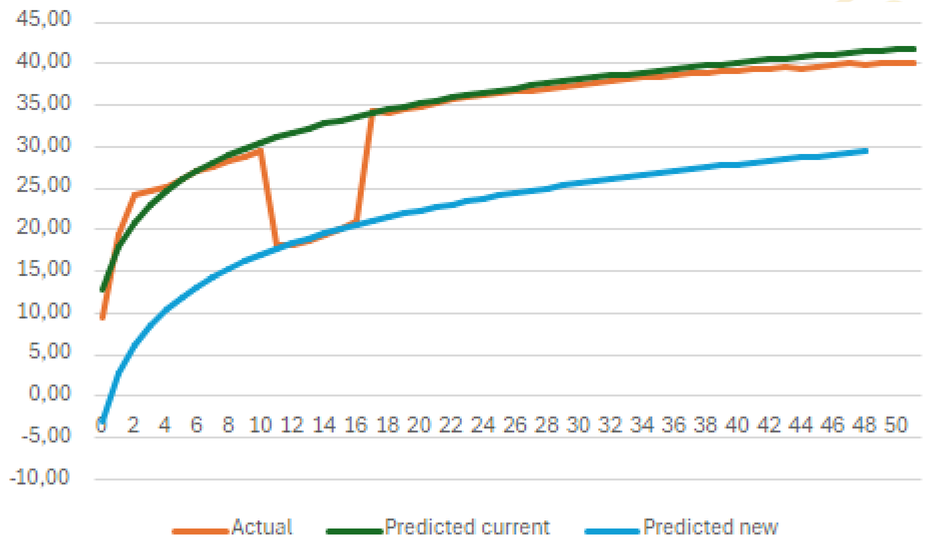
# Reconfiguration decision: reactive approach

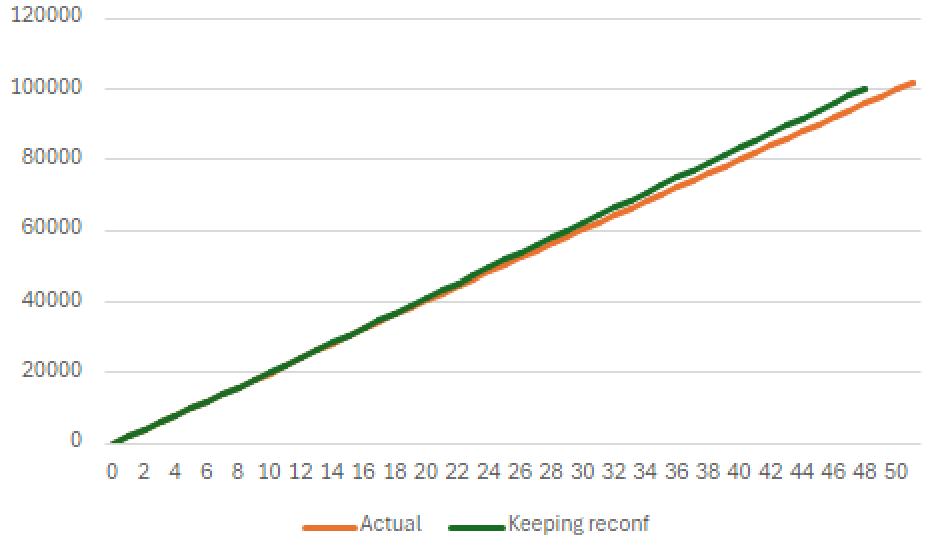
- Reconfiguration validation algorithm (total budget):
  1. Calculate reconfiguration cost
  2. Calculate function that describes the performance trend (regression)
  3. Perform reconfiguration
  4. Wait for W (reconfiguration validation window) rounds
     a) Get revert reconfiguration cost
     b) Calculate remaining rounds with initial configuration
     c) Calculate remaining rounds with new configuration
     d) Calculate function that described the performance trend of new configuration
     e) If predicted value new < predicted value current

        Revert configuration


C11
0 2000
6 2000
GA
LA1
LA2
C1
0 1000
1 1000
C2
2 1000
3 1000
C3
4 1000
5 1000
C4
6 1000
7 1000
C5
8 1000
9 1000
C6
0 1000
1 1000
C7
2 1000
3 1000
C8
4 1000
5 1000
C9
6 1000
7 1000
C10
8 1000
9 1000

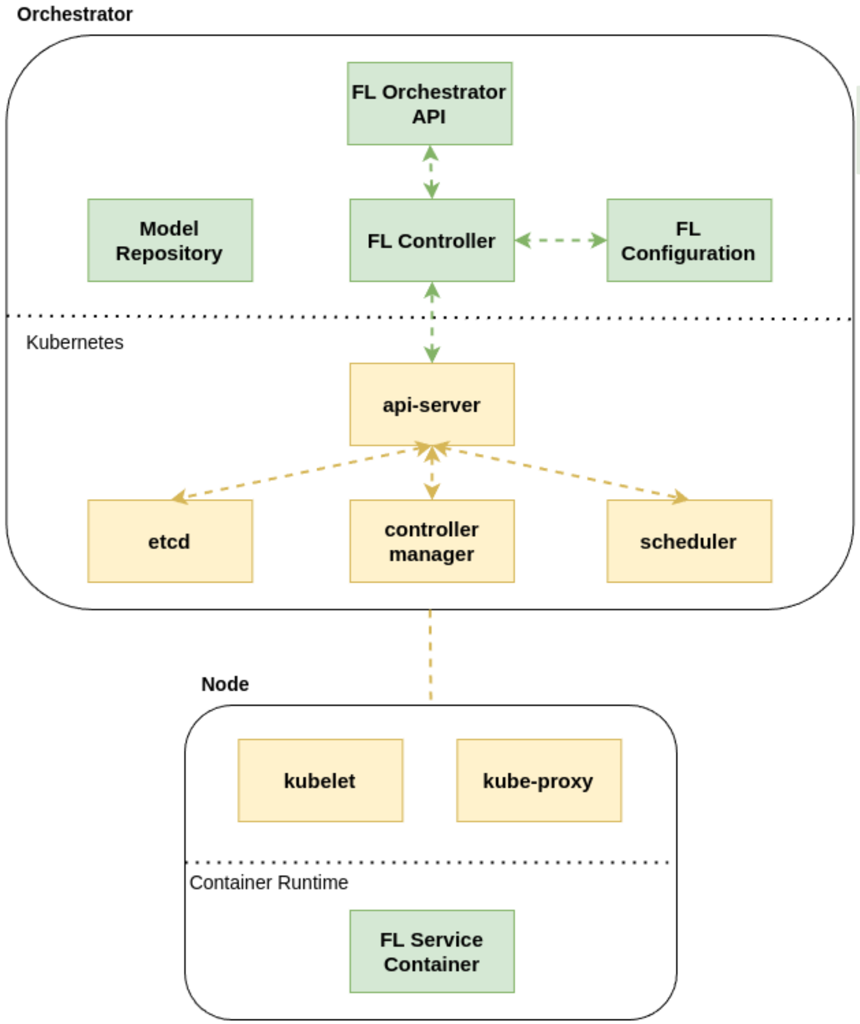

W = 5
45,00
40,00
35,00
30,00
25,00
20,00
15,00
10,00
5,00
0,00
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
Reconfiguration
Reconfiguration validation


45,00
40,00
35,00
30,00
25,00
20,00
15,00
10,00
5,00
0,00
-5,00
-10,00
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
Actual
Predicted current
Predicted new


120000
100000
80000
60000
40000
20000
0
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
Actual
Keeping reconf

# Implementation: Framework for 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



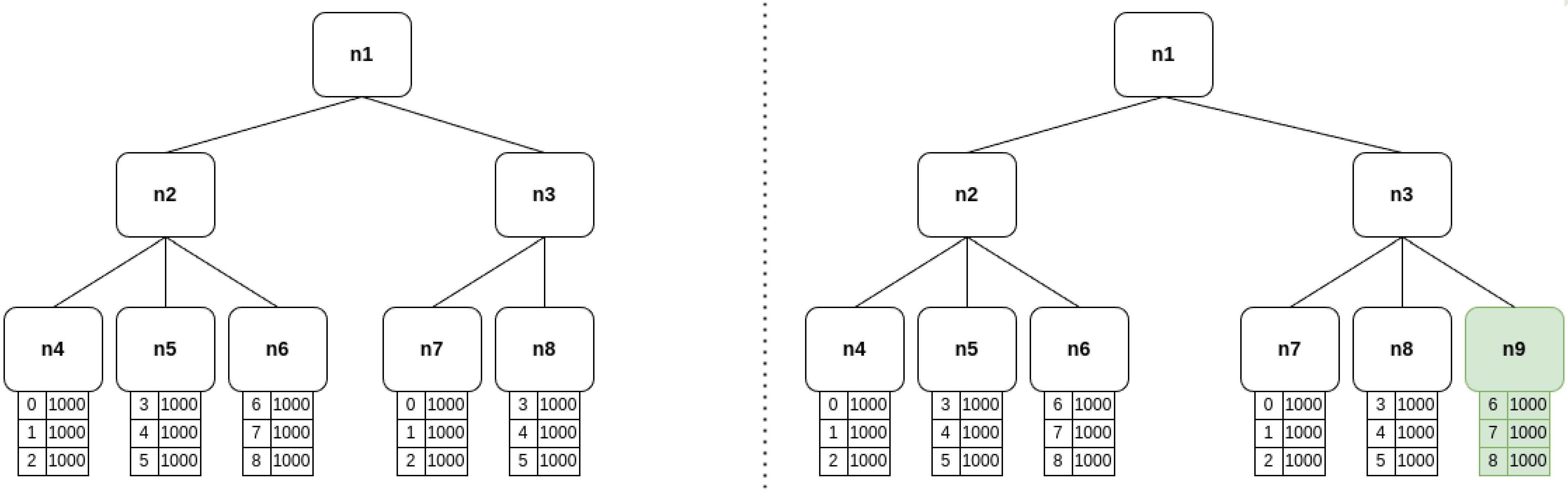
# AIoTwin Orchestration Middleware

Hands on session: Adaptive FL orchestration and reconfiguration validation

# Experimental environment

- K3s cluster consisting of 9 nodes
  - Each node is a VM with 2 CPU cores and 2 GB of RAM
- FL tasks are using only CPU's
- Deployment options
  - Simulated infrastructure
    - A node can host multiple FL services
    - FL entities and underlying network are defined with a configuration file
  - Actual infrastructure
    - One cluster node = one FL service
    - Network costs can be real or manually defined

# Reconfiguration validation: improvement



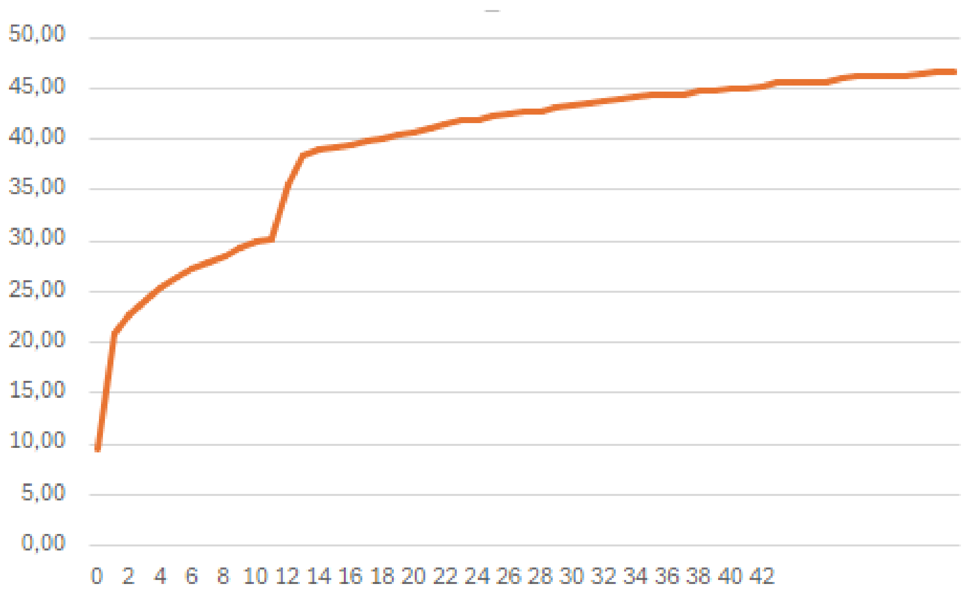
```
2024-09-11T08:48:41.604Z [INFO]  fl-orch: Starting FL with config minCommCost, modelSize 3.300000, and cost type totalBudget
Optimal clusters: [n4 n5 n6] [n7 n8]
Best comm cost:  990
Global aggregator ::
        &{Id:n1 InternalAddress:0.0.0.0:8080 ExternalAddress:fl-ga-svc-n1:8080 ParentAddress: Port:8080 NumClients:2 Rounds:100 LocalRounds:0}
Local aggregators ::
        &{Id:n2 InternalAddress:0.0.0.0:8080 ExternalAddress:fl-la-svc-n2:8080 ParentAddress:fl-ga-svc-n1:8080 Port:8080 NumClients:2 Rounds:100 LocalRounds:2}
        &{Id:n3 InternalAddress:0.0.0.0:8080 ExternalAddress:fl-la-svc-n3:8080 ParentAddress:fl-ga-svc-n1:8080 Port:8080 NumClients:2 Rounds:100 LocalRounds:2}
Clients ::
        &{Id:n4 ParentAddress:fl-la-svc-n2:8080 ParentNodeId:n2 Epochs:2 DataDistribution:map[0:1000 1:1000 2:1000]}
        &{Id:n5 ParentAddress:fl-la-svc-n2:8080 ParentNodeId:n2 Epochs:2 DataDistribution:map[3:1000 4:1000 5:1000]}
        &{Id:n6 ParentAddress:fl-la-svc-n2:8080 ParentNodeId:n2 Epochs:2 DataDistribution:map[6:1000 7:1000 8:1000]}
        &{Id:n7 ParentAddress:fl-la-svc-n3:8080 ParentNodeId:n3 Epochs:2 DataDistribution:map[0:1000 1:1000 2:1000]}
        &{Id:n8 ParentAddress:fl-la-svc-n3:8080 ParentNodeId:n3 Epochs:2 DataDistribution:map[3:1000 4:1000 5:1000]}
Epochs:  2
Local rounds:  2
```

```
2024-09-11T09:01:50.002Z [INFO]  fl-orch: New event:
2024-09-11T09:01:50.002Z [INFO]  fl-orch: Nodes added: [0xc0004cc660]
2024-09-11T09:01:50.002Z [INFO]  fl-orch: Node removed: []
Optimal clusters: [n5 n4 n6] [n7 n9 n8]
Best comm cost:  1320
2024-09-11T09:01:50.003Z [INFO]  fl-orch: Reconfiguration change cost: 165.00
2024-09-11T09:01:50.003Z [INFO]  fl-orch: Post reconfiguration cost: 660.00
2024-09-11T09:01:50.007Z [INFO]  fl-orch: reconfiguration evaluation set for round: 16
2024-09-11T09:01:50.007Z [INFO]  fl-orch: Starting reconfiguration:
```

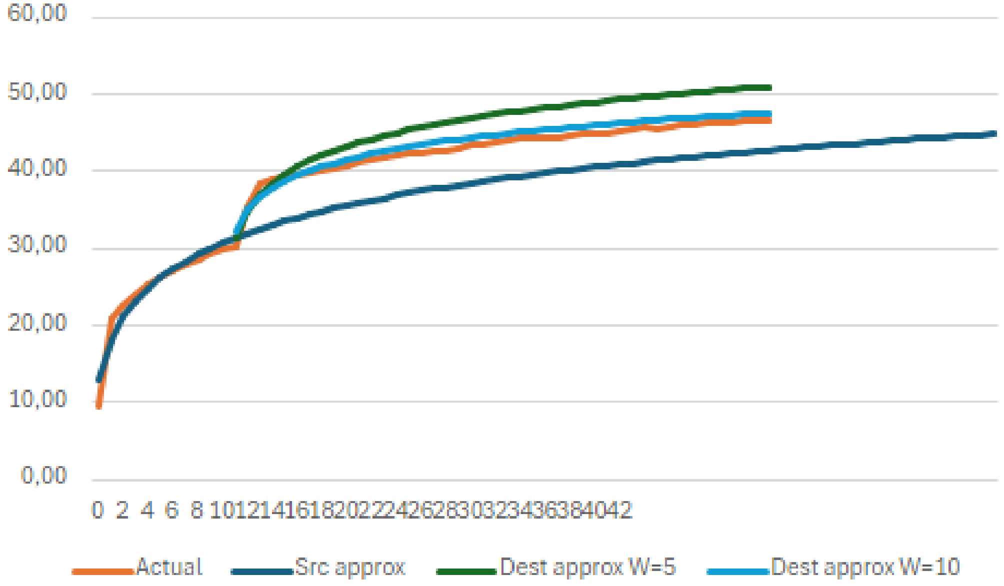
```
2024-09-11T09:09:29.588Z [INFO]  fl-orch: Starting reconf evaluation...
2024-09-11T09:09:29.588Z [INFO]  fl-orch: End accuracy: 39.42
2024-09-11T09:09:29.588Z [INFO]  fl-orch: Predicted accuracy: 33.97
2024-09-11T09:09:29.588Z [INFO]  fl-orch: Reconf change cost: 0.00
2024-09-11T09:09:29.588Z [INFO]  fl-orch: Remaining budget: 74755.00
2024-09-11T09:09:29.588Z [INFO]  fl-orch: Start rounds remaining: 56.00
2024-09-11T09:09:29.588Z [INFO]  fl-orch: End rounds remaining: 37.00
2024-09-11T09:09:29.588Z [INFO]  fl-orch: Start accuracy final: 44.84
2024-09-11T09:09:29.588Z [INFO]  fl-orch: End accuracy final: 50.74
2024-09-11T09:09:29.588Z [INFO]  fl-orch: Reconfiguration introduces performance improvement. Continuing with new configuration...
```

```
2024-09-11T09:58:12.040Z [INFO]  fl-orch: Communication budget exceeded!
Total cost: 100485.00
Final accuracy: 46.62
```

https://wandb.ai/aiotwins/k8sreal_7nodes_v2/runs/56nwu4le?nw=nwuserivancilic



50,00
45,00
40,00
35,00
30,00
25,00
20,00
15,00
10,00
5,00
0,00
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42

60,00
50,00
40,00
30,00
20,00
10,00
0,00
0 2 4 6 8 101214161820222426283032343638404 2
Actual
Src approx
Dest approx W=5
Dest approx W=10

# Reconfiguration validation: degradation