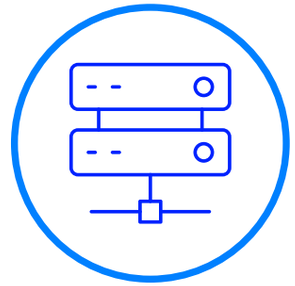


PULCEO in Action

Towards API-driven Cloud-Edge Orchestration
with PULCEO: A Proof of Concept

Sebastian Böhm and Guido Wirtz

University of Bamberg, Germany

https://github.com/spboehm/pulceo-misc

Cloud-Edge Orchestration

Edge computing: Placing of computational resources close to end-users.

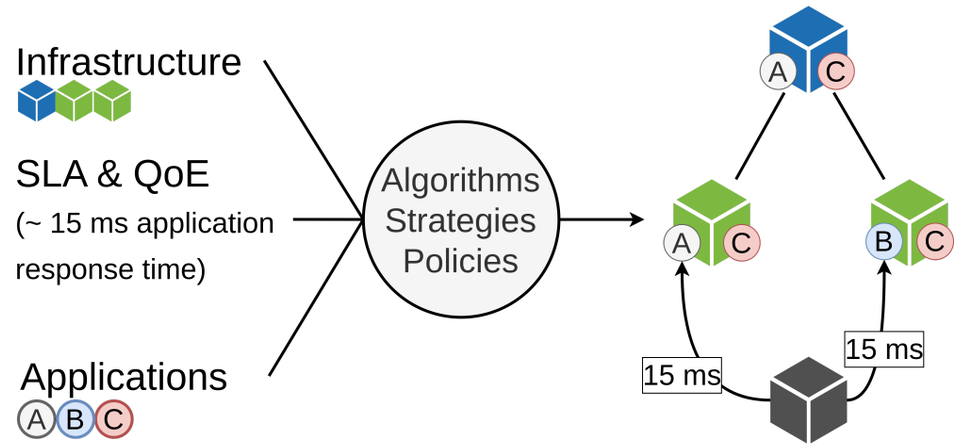
Many solutions exist for **service placement**

But, limited

- **Reproducibility**
- (General) **Applicability**

because of

- **custom** implementations
- missing **real-world experiments**



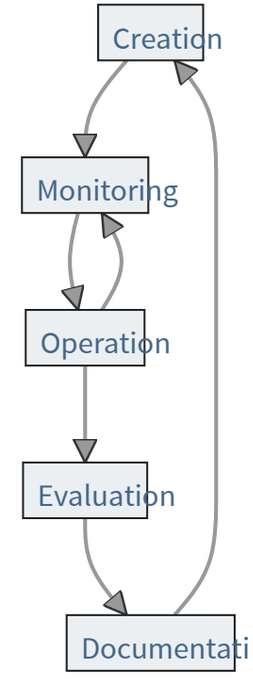
Similar **infrastructures**, **optimization goals**, and **orchestration** operations.

Simulations are prevalent: only 19 out of 99 solutions used a small test-bed.[1]

1. S. Smolka and Z. Á. Mann, “Evaluation of fog application placement algorithms: a survey,” Computing, vol. 104, no. 6, pp. 1397–1423, Jun. 2022.

Solution: Holistic Management

- **Creation of Topology:**
  - **Nodes:** Edge and fog devices, virtual machines, etc.
  - **Links:** Relations between nodes for measuring round-trip time and bandwidth
- **Monitoring:** Various kind of metrics (CPU, …), sampling rate, etc.
- **Operation:** Resource allocation and service placement
- **Evaluation:** Data export, large-scale data analytics, etc.
- **Documentation:** Publishing of reports and results, raw data, etc.
- (Deletion: Tearing down of environments)



PULCEO

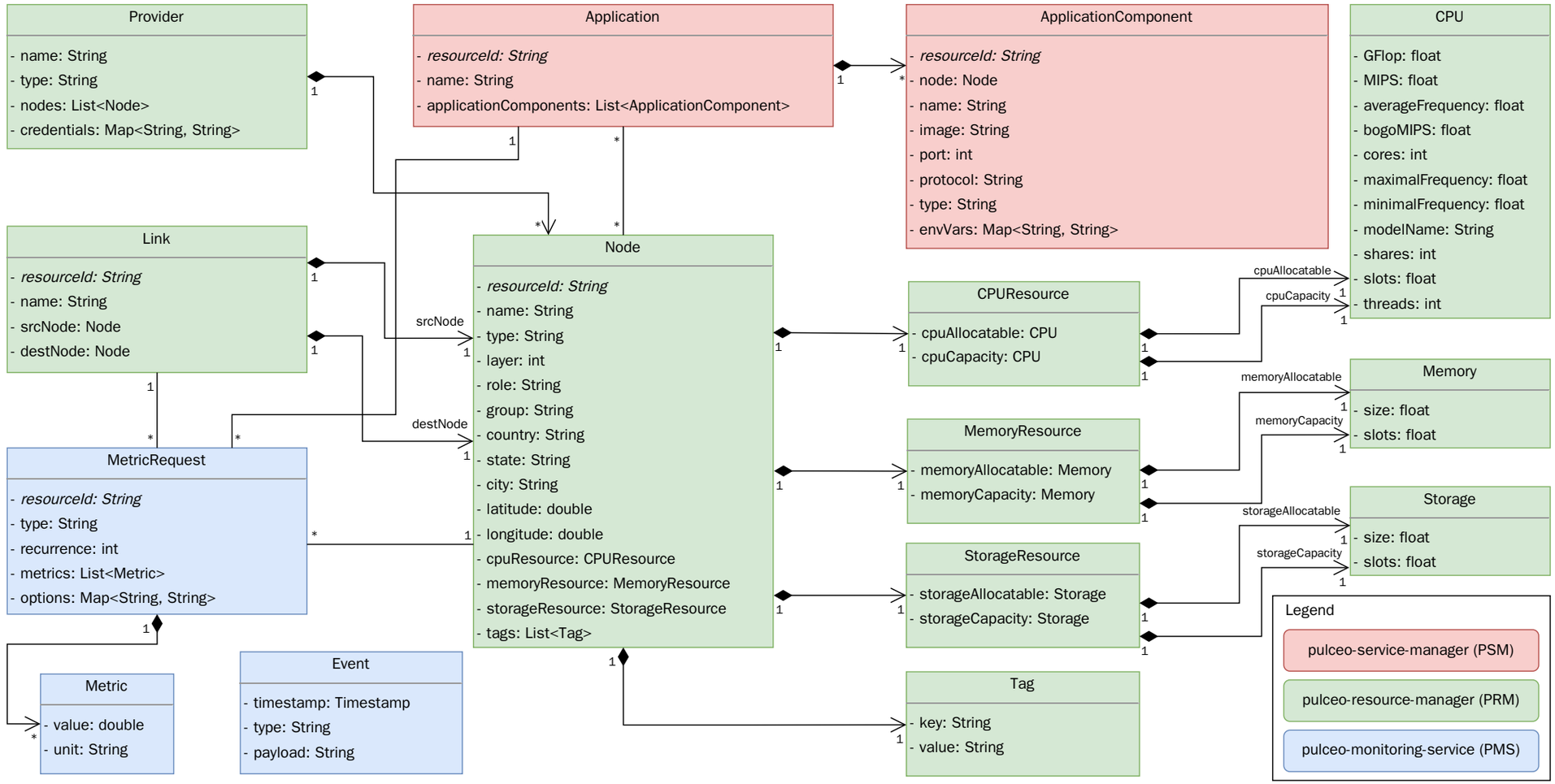
Platform for **U**niversal and **L**ightweight **C**loud-**E**dge **O**rchestration

Domain Model · Architecture · Node Agent

https://github.com/spboehm/pulceo-misc

PULCEO's Domain Model

Extracted from **27 peer-reviewed** service placement solutions[1]

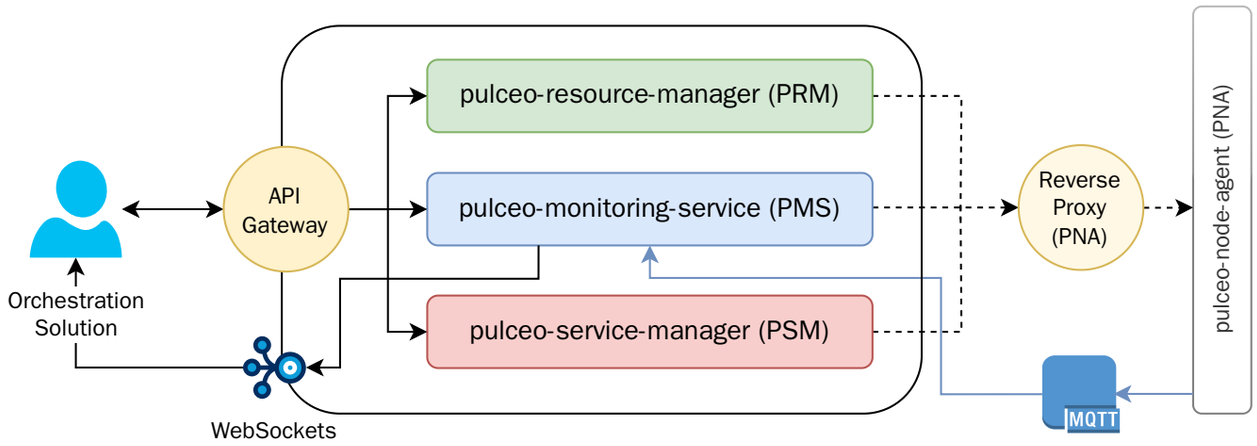


1. https://spboehm.github.io/pulceo-misc/

https://github.com/spboehm/pulceo-misc

PULCEO's Architecture

- Decoupled orchestration with a RESTful HTTP API exposed by an API Gateway
- Microservice architecture aligned to a scientific meta-study[1]
- Real-time data streaming via WebSockets

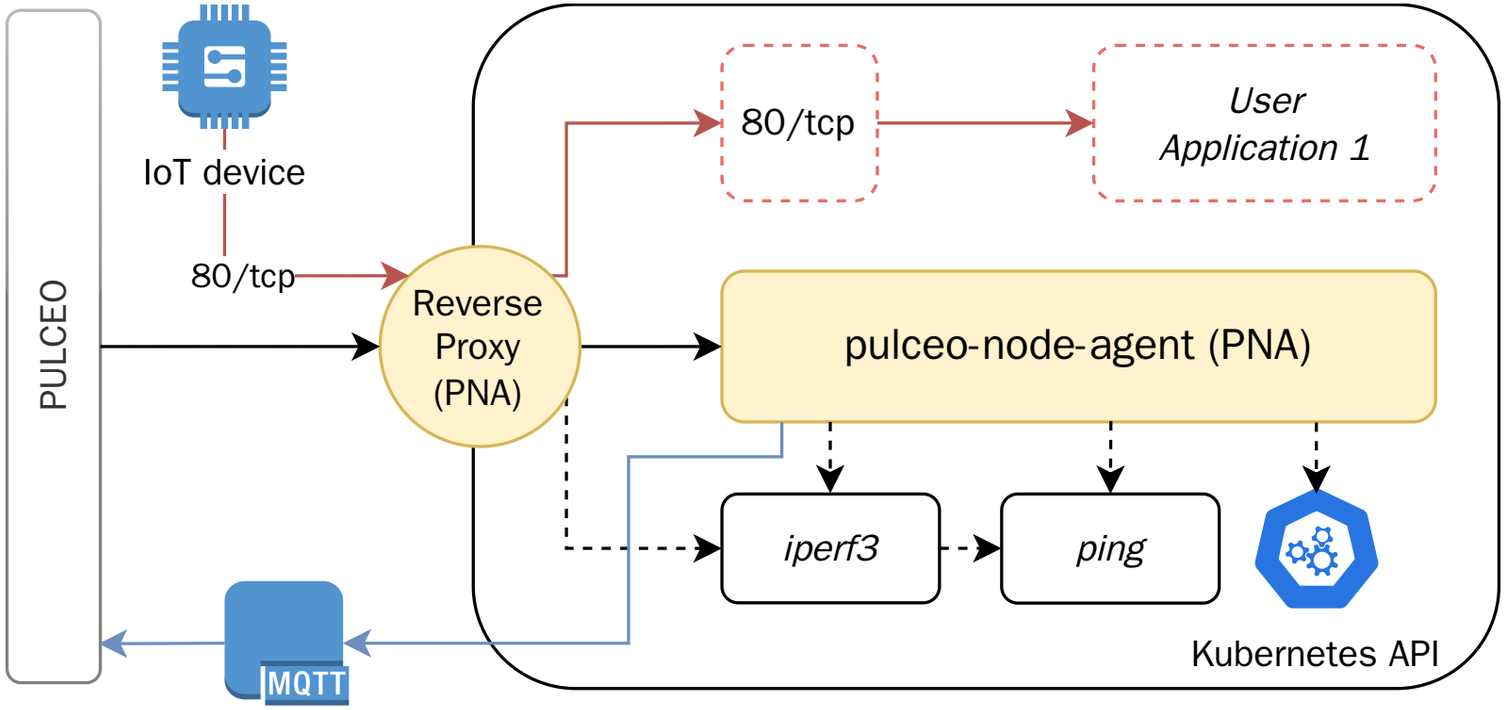


1. B. Costa, J. Bachiega, L. R. de Carvalho, and A. P. F. Araujo, "Orchestration in Fog Computing: A Comprehensive Survey," ACM Comput. Surv., vol. 55, no. 2, pp. 1–34, Feb. 2023, doi: 10.1145/3486221.

https://github.com/spboehm/pulceo-misc

PULCEO Node Agent Architecture

- RESTful HTTP API for instructions
- Monitoring data transmitted via MQTT
- Latency and bandwidth measurement with *ping* and *iperf3*
- Standalone Kubernetes clusters as container manager



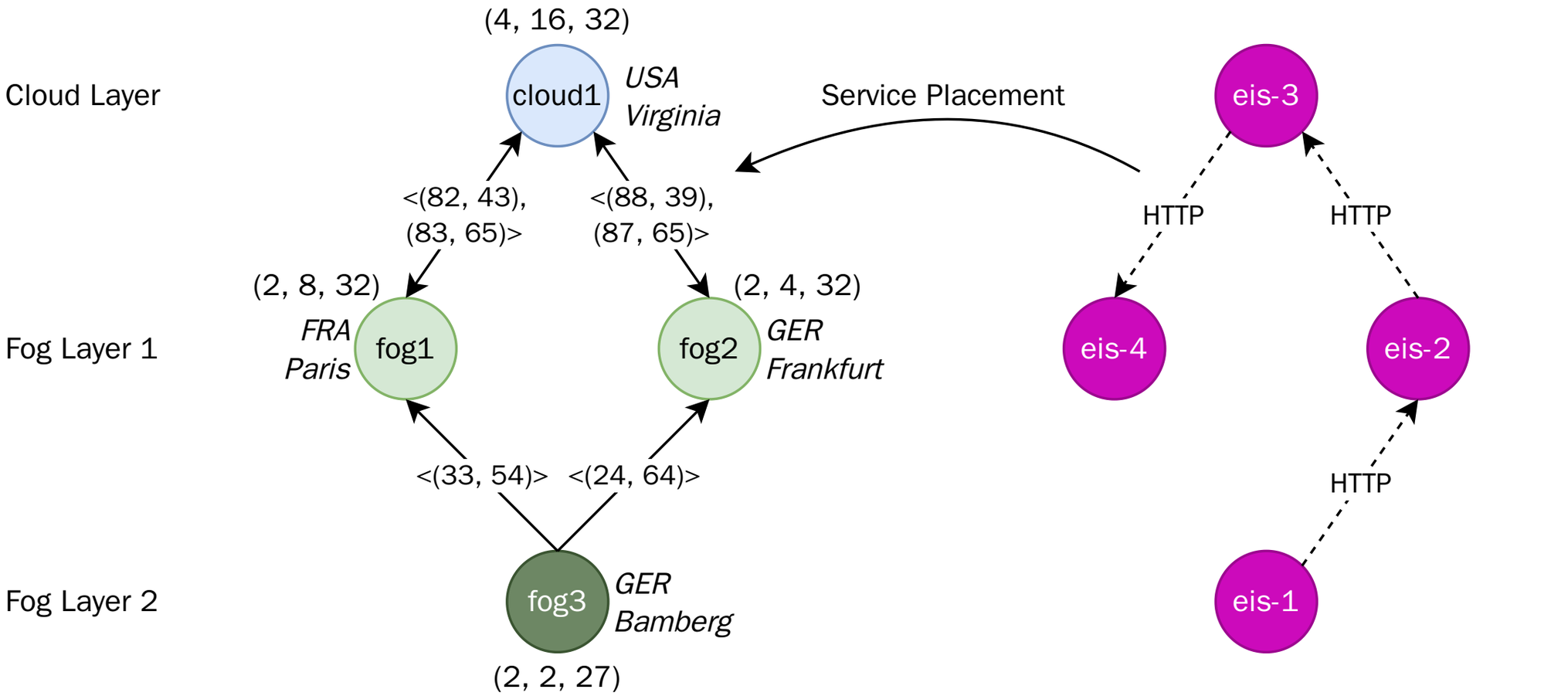
https://github.com/spboehm/pulceo-misc

Case Study

Topology · Orchestration workflow

https://github.com/spboehm/pulceo-misc

Topology



Representational cloud-edge topology with nodes, links, and requests for service placement.

Creation

Providers

Providers supply computational resources, e.g., Compute, Network, Storage, etc.

Two types of providers:

- **On-premises** providers (any virtual machine), built-in
- **Cloud** providers (API availability), e.g., Microsoft Azure

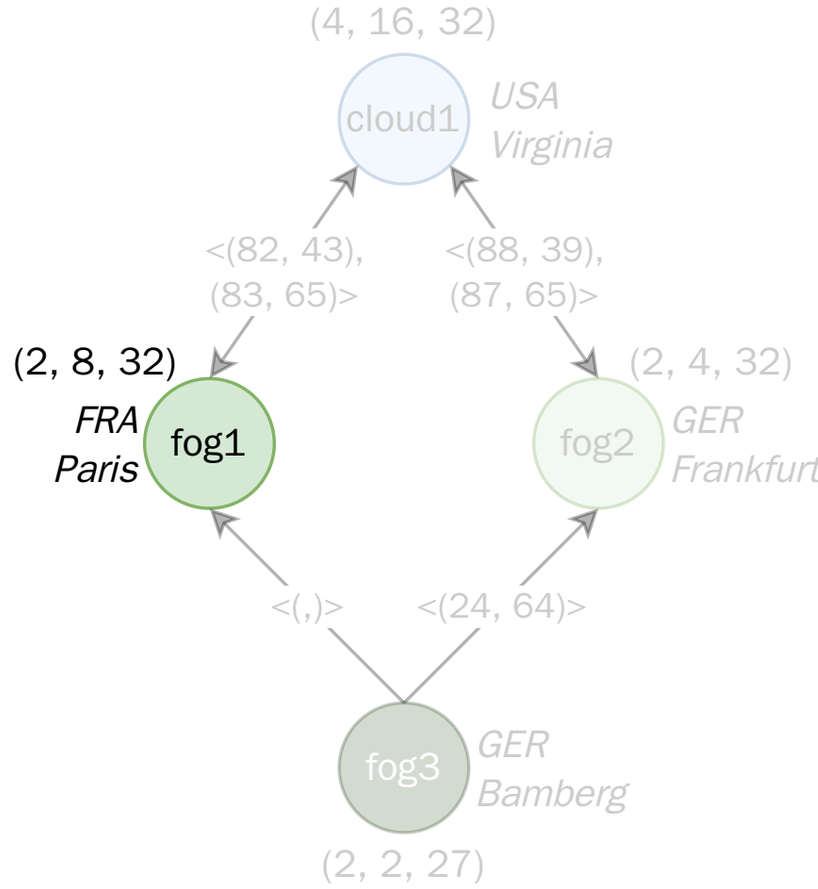
Example: Creation of Microsoft Azure as provider with a service principal

```
curl --request POST \
  --url http://localhost:8081/api/v1/providers \
  --header 'Accept: application/json' \
  --header 'Content-Type: application/json' \
  --data '{
    "providerName": "azure-provider",
    "providerType": "AZURE",
    "clientId": "00000000-00000000-00000000-00000000",
    "clientSecret": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",
    "tenantId": "00000000-00000000-00000000-00000000",
    "subscriptionId": "00000000-00000000-00000000-00000000"
  }'
```


Nodes (fog1)

- Provider: Microsoft Azure (Cloud)
- Capabilities: 2 vCPU, 8 GB memory, 32 GB storage
- Location: France, Paris

```
curl --request POST \
  --url http://localhost:8081/api/v1/nodes \
  --header 'Accept: application/json' \
  --header 'Authorization: XXXXX' \
  --header 'Content-Type: application/json' \
  --data '{
    "nodeType":"AZURE",
    "providerName":"azure-provider",
    "name":"fog1",
    "type":"fog",
    "cpu":"2",
    "memory":"8",
    "region":"francecentral",
    "tags": []
}'
```

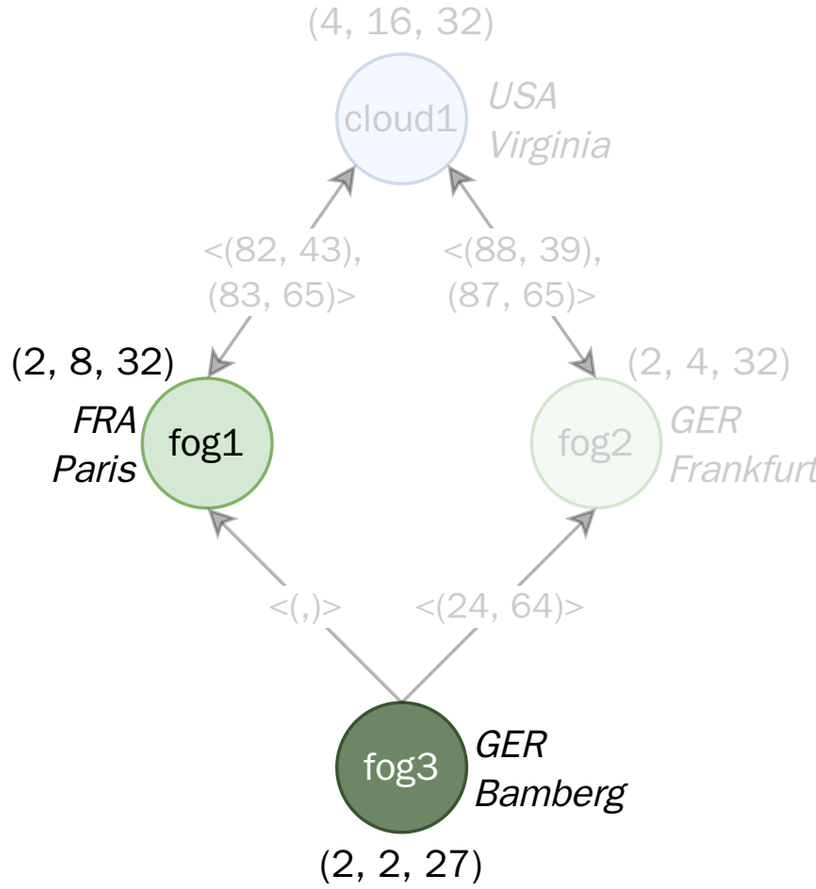


https://github.com/spboehm/pulceo-misc

Nodes (fog3)

- Provider: Local data center (On-premises)
- Capabilities: 2 vCPU, 2 GB memory, 27 GB storage
- Location: Bamberg, Germany

```
curl --request POST \
  --url http://localhost:8081/api/v1/nodes \
  --header 'Accept: application/json' \
  --header 'Authorization: XXXXX' \
  --header 'Content-Type: application/json' \
  --data '{
    "nodeType":"ONPREM",
    "type": "fog",
    "name": "fog3",
    "providerName":"default",
    "hostname":"h5138.pi.uni-bamberg.de",
    "pnaInitToken":"XXXXX",
    "country": "Germany",
    "state": "Bavaria",
    "city": "Bamberg",
    "latitude": 49.9036,
    "longitude": 10.8700,
    "tags": []
}'
```

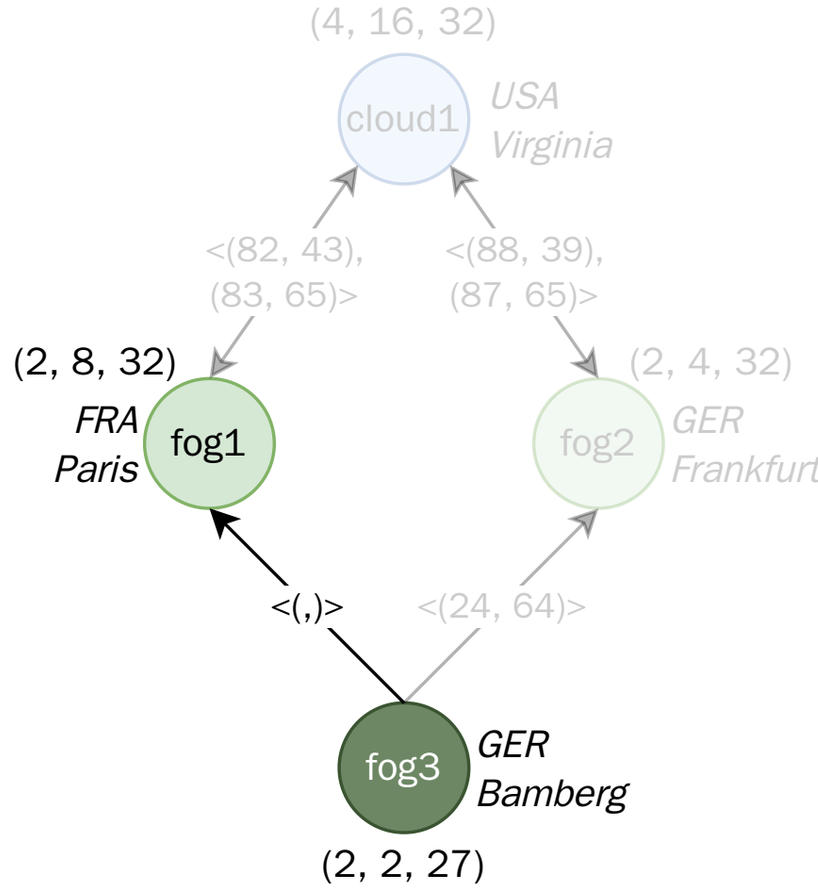


https://github.com/spboehm/pulceo-misc

Links (Example fog3-fog1)

- Link between fog3 and fog1
- Represents a logical connection
- Later used to obtain round-trip time and bandwidth between nodes
- <(,)>: <(*latency*, *bandwidth*), …>

```
curl --request POST \
  --url http://localhost:8081/api/v1/links \
  --header 'Accept: application/json' \
  --header 'Authorization: XXXXX' \
  --header 'Content-Type: application/json' \
  --data '{
    "linkType": "NODE_LINK",
    "name": "fog3-fog1",
    "srcNodeId": "fog3",
    "destNodeId": "fog1"
}'
```



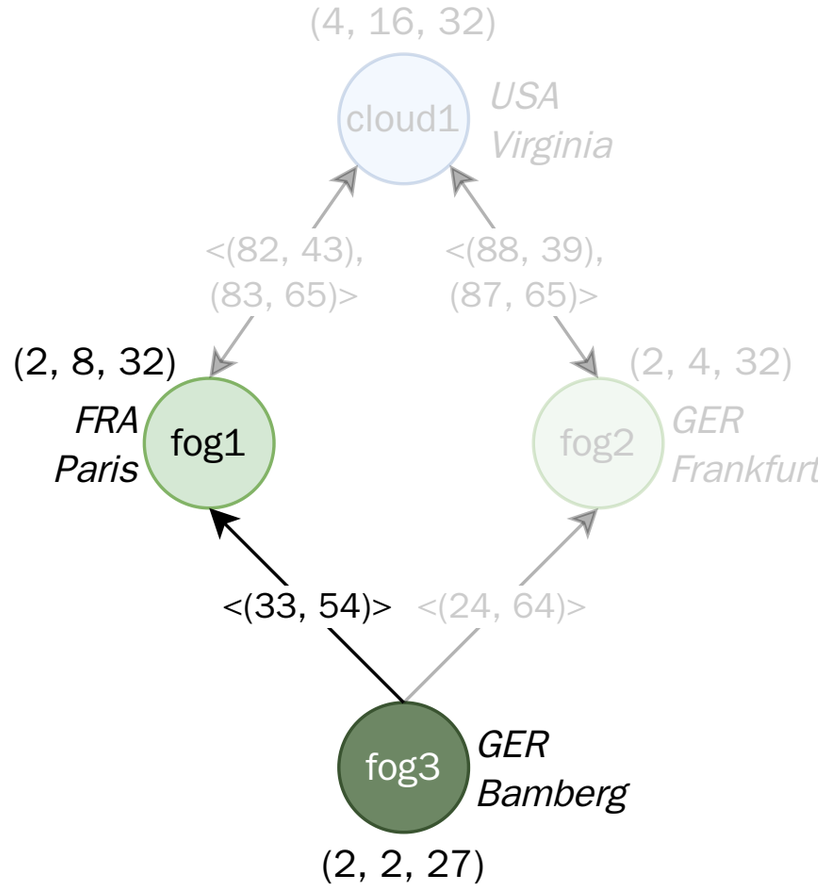
Monitoring

Nodes · Links

Metric Requests

- Collection of monitoring data
  - CPU, memory, storage, and network utilization for nodes and applications
  - ICMP round-trip time, TCP & UDP bandwidth for links
- Individual and batch (*) assignments
- **Example:** Latency all for links, once per hour (recurrence 3600s = 1h)

```
curl --request POST \
  --url http://localhost:8081/api/v1/metric-reques
  --header 'Accept: application/json' \
  --header 'Authorization: XXXX' \
  --header 'Content-Type: application/json' \
  --data '{
    "type":"icmp-rtt",
    "linkId":"*",
    "recurrence":"3600"
}'
```



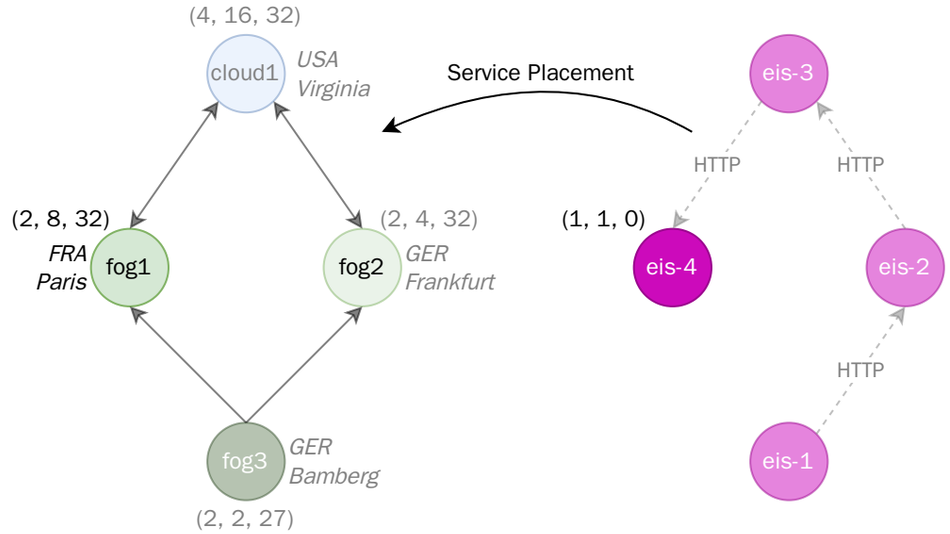
https://github.com/spboehm/pulceo-misc

Operation

Workload · Resource Management · Service Placement

Workload

- Placement of edge-iot-simulator (eis)[1] on node fog1
- Mimics a typical application:
  - Simulates a temperature sensor (sends temperature readings at regular intervals)
  - Allows to perform HTTP requests to simulate a microservice application
- Example resource assignments:
  - 1 vCPU (1000 shares)
  - 1 GB memory (1000 MB)



1. https://github.com/spboehm/edge-iot-simulator

Resource Management (Example CPU)

Read CPU resources of fog1

```
curl --request GET \
  --url http://localhost:8081/api/v1/nodes/fog1/cp
  --header 'Accept: application/json' \
  --header 'Authorization: XXXXX' \
```


```
{
    "uuid": "8aeae447-a552-4ea2-86a3-2bd1f79d6117"
    "nodeUUID": "e1076174-380a-47e4-a468-b9fd1b0ea
    "nodeName": "fog1",
    "cpuCapacity": {...},
    "cpuAllocatable": {
        "modelName": "Intel(R) Xeon(R) Platinum 82
        "cores": 2,
        "threads": 2,
        "bogoMIPS": 5187.81,
        "minimalFrequency": 2593.906,
        "averageFrequency": 2593.906,
        "maximalFrequency": 2593.906,
        "shares": 2000,
        "slots": 0.0,
        "mips": 5187.81,
        "gflop": 0.0
    }
}
```


Update CPU resources of fog1

```
curl --request PATCH \
  --url http://localhost:8081/api/v1/nodes/fog1/cp
  --data '{
    "key": "shares",
    "value": 1000
}'
```


```
{
    "uuid": "8aeae447-a552-4ea2-86a3-2bd1f79d6117"
    "nodeUUID": "e1076174-380a-47e4-a468-b9fd1b0ea
    "nodeName": "fog1",
    "cpuCapacity": {...},
    "cpuAllocatable": {
        "cores": 2,
        "threads": 2,
        "bogoMIPS": 5187.81,
        "averageFrequency": 2593.906,
        "shares": 1000,
        "slots": 0.0,
        "mips": 5187.81,
        "gflop": 0.0,
    ...
    }
}
```


https://github.com/spboehm/pulceo-misc

Applications (Service Placement)

```
curl --request POST \
  --url http://localhost:8081/api/v1/applications \
  --header 'Accept: application/json' \
  --header 'Authorization: XXXXX' \
  --data '{
    "nodeId": "fog1",
    "name": "edge-iot-simulator",
    "applicationComponents": [
        {
            "name": "component-eis",
            "image": "ghcr.io/spboehm/edge-iot-simulator:v1.1.0",
            "port": 80,
            "protocol": "HTTPS",
            "applicationComponentType": "PUBLIC",Resource utilization with
            "environmentVariables": {
                "MQTT_SERVER_NAME":"XXXXX.s1.eu.hivemq.cloud",
                "MQTT_PORT":"8883",
                "MQTT_TLS":"True",
                "MQTT_USERNAME": "XXXXX",
                "MQTT_PASSWORD": "XXXXX",
                "MQTT_CLIENT_ID":"fog1-edge-iot-simulator",
                "WEB_PORT": 80
            }
        }
    ]
}'
```

Resource utilization with *If needed, further Metrics Requests to monitor the placed Applications can be issued.*

https://github.com/spboehm/pulceo-misc

Evaluation

API Requests · Idle Resource Utilization · Link Quality Metrics · Application Resource Utilization · Application Response Time

https://github.com/spboehm/pulceo-misc

API Requests

41 requests required to perform the entire orchestration workflow

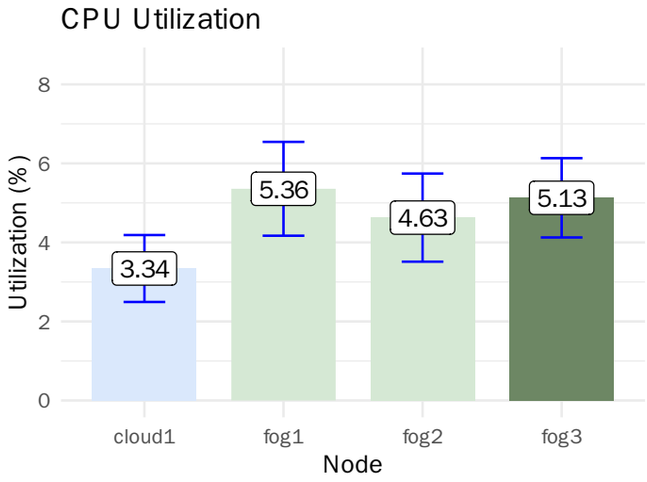
Table 2. Overview of API requests.

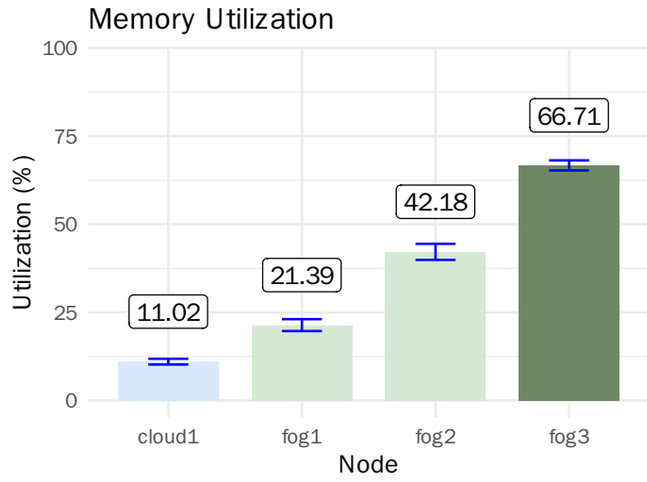
#	Service	Endpoint	Method	Count	Explanation
1	PRM	/providers	POST	1	Register Azure as *Provider*
2	PRM	/nodes	POST	4	Create *Nodes*
3	PRM	/nodes/fog3	GET	1	Read properties of *Node fog3*
4	PRM	/nodes/fog3	PATCH	1	Update layer of *fog3* to 2
5	PRM	/links	POST	6	Create *Links* between *Nodes*
6	PMS	/metric-requests	POST	7	Create requests for *Nodes* and *Links*
7	PSM	/applications	POST	4	Create edge-iot-simulator
8	PRM	/nodes/{id}/cpu	GET	4	Read allocatable shares
9	PRM	/nodes/{id}/cpu	PATCH	4	Update allocatable shares
10	PRM	/nodes/{id}/memory	GET	4	Read allocatable capacity
11	PRM	/nodes/{id}/memory	PATCH	4	Update allocatable capacity
12	PMS	/metric-requests	POST	4	Create requests for *Applications*

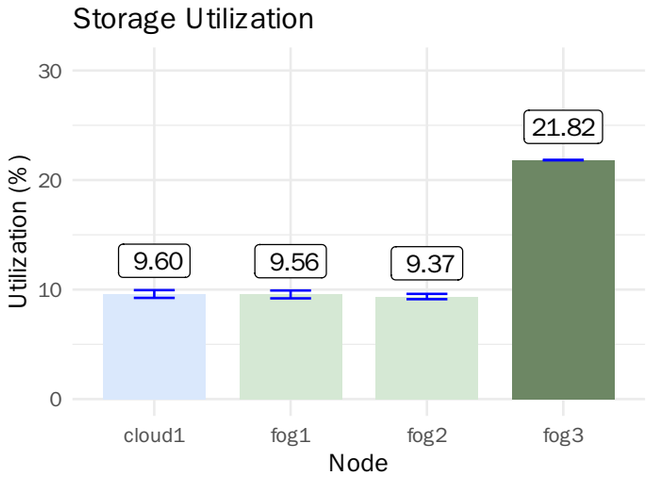
https://github.com/spboehm/pulceo-misc

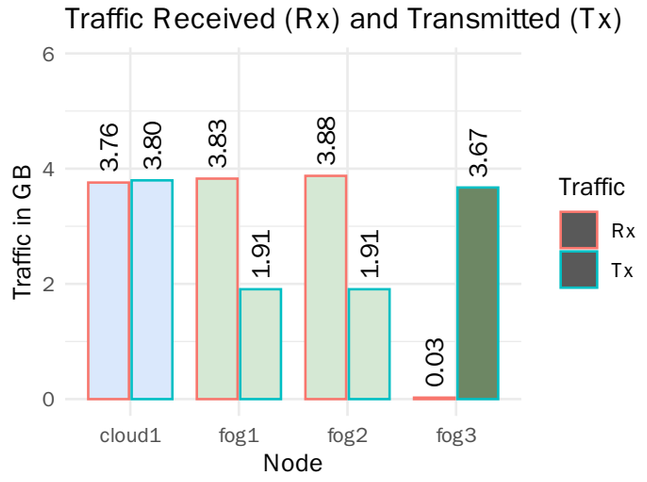
Idle Resource Utilization by Nodes

Including **pulceo-node-agent,** fully configured **monitoring,** and **Kubernetes**









https://github.com/spboehm/pulceo-misc

Link Quality Metrics

Using a **high-performance** and **stable** network for on-premises and cloud

Table 3. ICMP round-trip time (ms) between nodes.

v_1	v_2	Min	Mean	Max	Med	SD
cloud1	fog1	80.795	81.142	82.924	81.098	0.402
cloud1	fog2	86.709	88.896	91.149	89.038	1.217
fog1	cloud1	80.779	81.107	82.327	81.024	0.304
fog2	cloud1	86.460	87.802	88.819	87.960	0.548
fog3	fog1	25.558	26.139	33.058	25.802	1.487
fog3	fog2	13.077	15.402	24.150	14.709	2.472

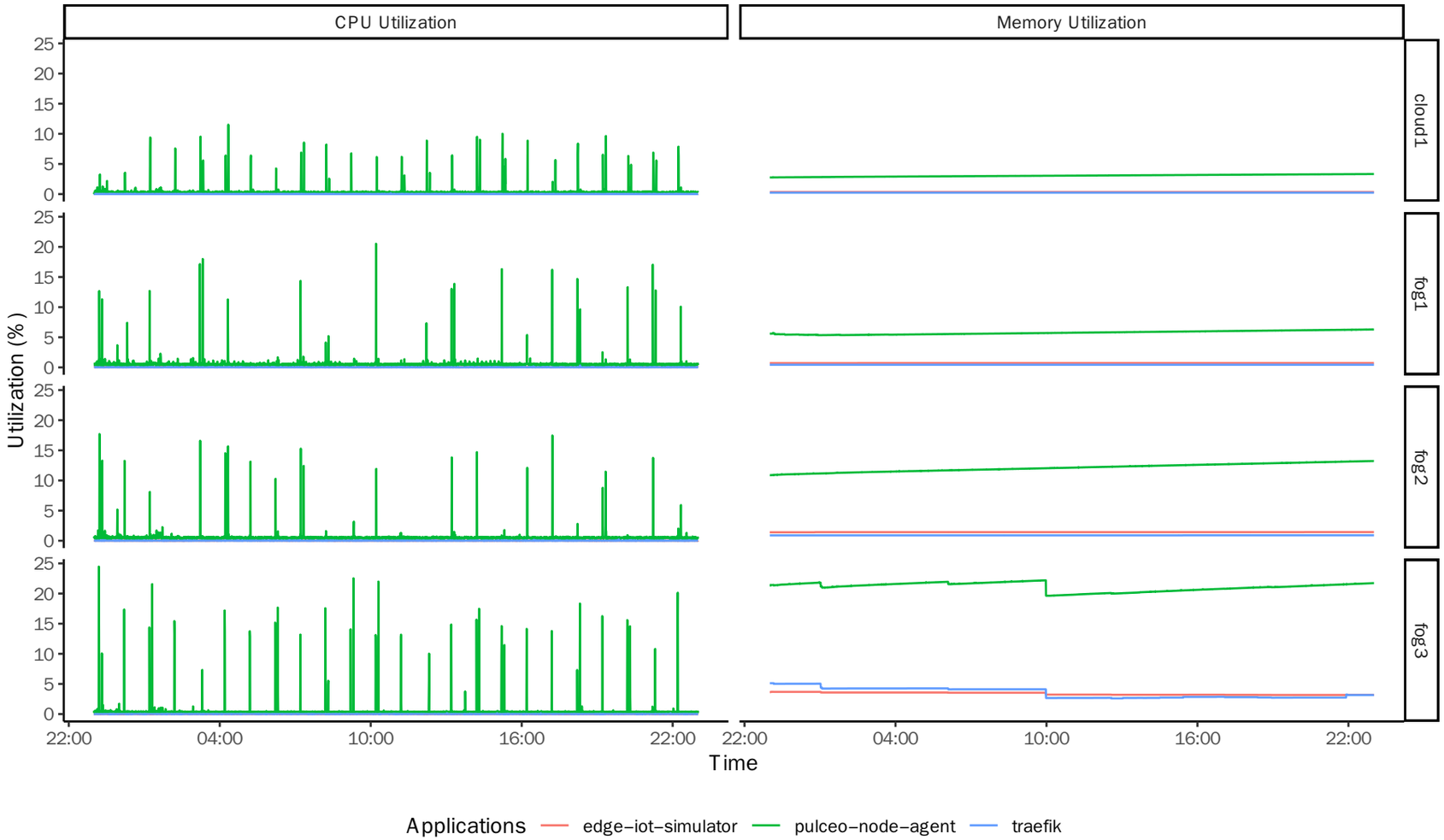
Table 4. TCP and UDP bandwidth (Mbps) between nodes.

		TCP					UDP				
v_1	v_2	Min	Mean	Max	Med	SD	Min	Mean	Max	Med	SD
cloud1	fog1	65.000	65.000	65.000	65.000	0.000	63.800	63.942	64.000	63.900	0.058
cloud1	fog2	65.000	65.000	65.000	65.000	0.000	63.800	63.875	63.900	63.900	0.044
fog1	cloud1	43.400	64.061	65.000	65.000	4.504	56.200	63.354	64.000	64.000	2.084
fog2	cloud1	39.400	63.887	65.000	65.000	5.338	55.300	63.517	63.900	63.900	1.751
fog3	fog1	53.500	64.392	65.000	65.000	2.385	64.400	64.443	64.600	64.400	0.066
fog3	fog2	64.400	64.950	65.000	65.000	0.169	64.700	64.762	64.800	64.800	0.049

https://github.com/spboehm/pulceo-misc

Application Resource Utilization

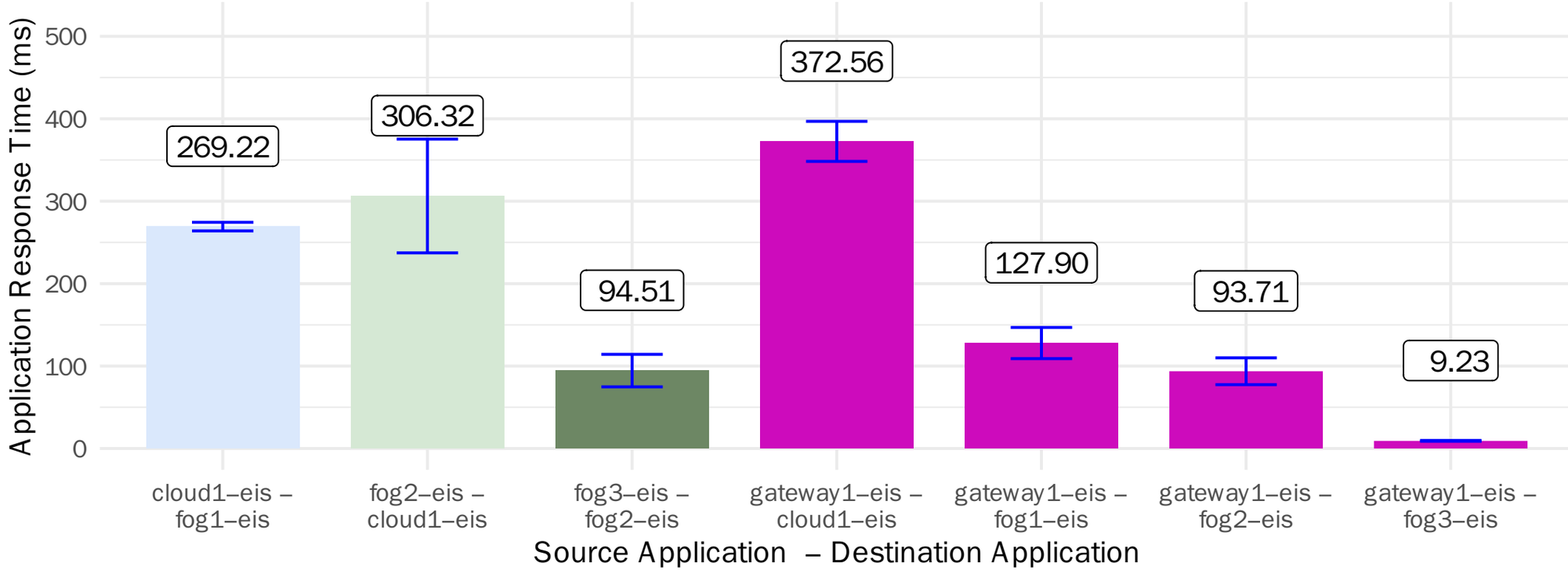
With deployed edge-iot-simulators (eis)



https://github.com/spboehm/pulceo-misc

Application Response Time

- Measured by edge-iot-simulators (eis)
- Values have been submitted in a standardized JSON format via MQTT



https://github.com/spboehm/pulceo-misc

Documentation

Orchestration Data

https://github.com/spboehm/pulceo-misc

Orchestration Data

JSON export of all entities of the domain model: *Providers*, *Nodes*, *Links*, *Metric-Requests*, *Applications*, *Resources*, *CPUs*, *Memory*, *Events*

Example for nodes:

```
curl --request GET \
  --url http://localhost:8081/api/v1/nodes \
  --header 'Accept: application/json' \
  --header 'Authorization: XXXXX' \
```

▶ JSON output for nodes

Reports for the two phases:

SummerSOC2024-prod-idle

SummerSOC2024-prod-load

Related Solutions

EU Projects (Horizon Research), like CODECO, FogAtlas, SODALITE@RT, ENACT, etc.

- Latency and bandwidth measurement not fully implemented
- Scheduling for service placement often pre-implemented
- Focus not on scientific experiments

Conceptual and **prototypical research efforts**, like Sophos, Fluidity, ACOA, etc.

- No holistic cloud-edge orchestration (see above)
- Lack of documentation

Out of scope: **Cloud** and **commercial** solutions

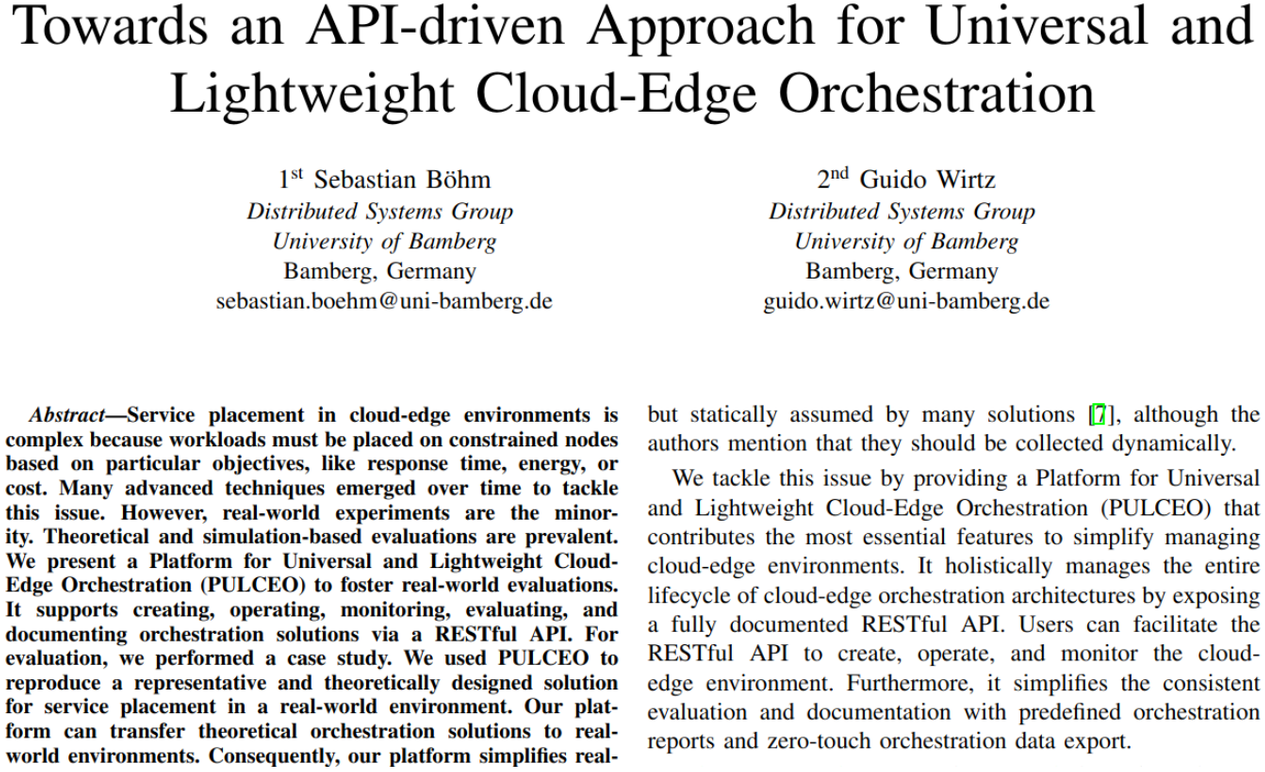
Contributions

- Fully documented **RESTful HTTP API** for universal orchestration (OpenAPI specification)
- **Decoupled** and **holistic** cloud-edge orchestration with evaluation and documentation
- Strong **scientific** and **industrial** foundation
  - Platform architecture based on a scientific meta-study[1]
  - Feature engineering based on scientific, peer-reviewed service placement publications[2]
  - Implementation following an industry standard (OpenFog RA)[3]

Limitations: Only one representational architecture implemented with stable network conditions has been reproduced yet

1. B. Costa, J. Bachiega, L. R. de Carvalho, and A. P. F. Araujo, “Orchestration in Fog Computing: A Comprehensive Survey,” ACM Comput. Surv., vol. 55, no. 2, pp. 1–34, Feb. 2023, doi: 10.1145/3486221.
2. https://spboehm.github.io/pulceo-misc/
3. S. Bohm and G. Wirtz, “PULCEO - A Novel Architecture for Universal and Lightweight Cloud-Edge Orchestration,” in 2023 IEEE International Conference on Service-Oriented System Engineering (SOSE), Athens, Greece: IEEE, Jul. 2023, pp. 37–47.

Future Work



Towards an API-driven Approach for Universal and Lightweight Cloud-Edge Orchestration

1st Sebastian Böhm
Distributed Systems Group
University of Bamberg
Bamberg, Germany
sebastian.boehm@uni-bamberg.de

2nd Guido Wirtz
Distributed Systems Group
University of Bamberg
Bamberg, Germany
guido.wirtz@uni-bamberg.de

***Abstract*—Service placement in cloud-edge environments is complex because workloads must be placed on constrained nodes based on particular objectives, like response time, energy, or cost. Many advanced techniques emerged over time to tackle this issue. However, real-world experiments are the minority. Theoretical and simulation-based evaluations are prevalent. We present a Platform for Universal and Lightweight Cloud-Edge Orchestration (PULCEO) to foster real-world evaluations. It supports creating, operating, monitoring, evaluating, and documenting orchestration solutions via a RESTful API. For evaluation, we performed a case study. We used PULCEO to reproduce a representative and theoretically designed solution for service placement in a real-world environment. Our platform can transfer theoretical orchestration solutions to real-world environments. Consequently, our platform simplifies real-**

but statically assumed by many solutions [7], although the authors mention that they should be collected dynamically.

We tackle this issue by providing a Platform for Universal and Lightweight Cloud-Edge Orchestration (PULCEO) that contributes the most essential features to simplify managing cloud-edge environments. It holistically manages the entire lifecycle of cloud-edge orchestration architectures by exposing a fully documented RESTful API. Users can facilitate the RESTful API to create, operate, and monitor the cloud-edge environment. Furthermore, it simplifies the consistent evaluation and documentation with predefined orchestration reports and zero-touch orchestration data export.

S. Bohm and G. Wirtz, “Towards an API-driven Approach for Universal and Lightweight Cloud-Edge Orchestration,” in 2024 IEEE International Conference on Service-Oriented System Engineering (SOSE), accepted, to be published.

Further tests with **other service placement strategies**, as done for **IEEE SOSE 2024**: Full real-world realization of *Qos-aware Deployment of IoT Application Through the Fog*, by *Brogi and Forti (2017)*

Source Code, Container Images, OpenAPI Specifications, and Documentation

https://github.com/spboehm/pulceo-misc