PULCEO in Action

API-driven Task Scheduling and Offloading
with PULCEO: An Extension

Sebastian Böhm and Guido Wirtz

University of Bamberg, Germany

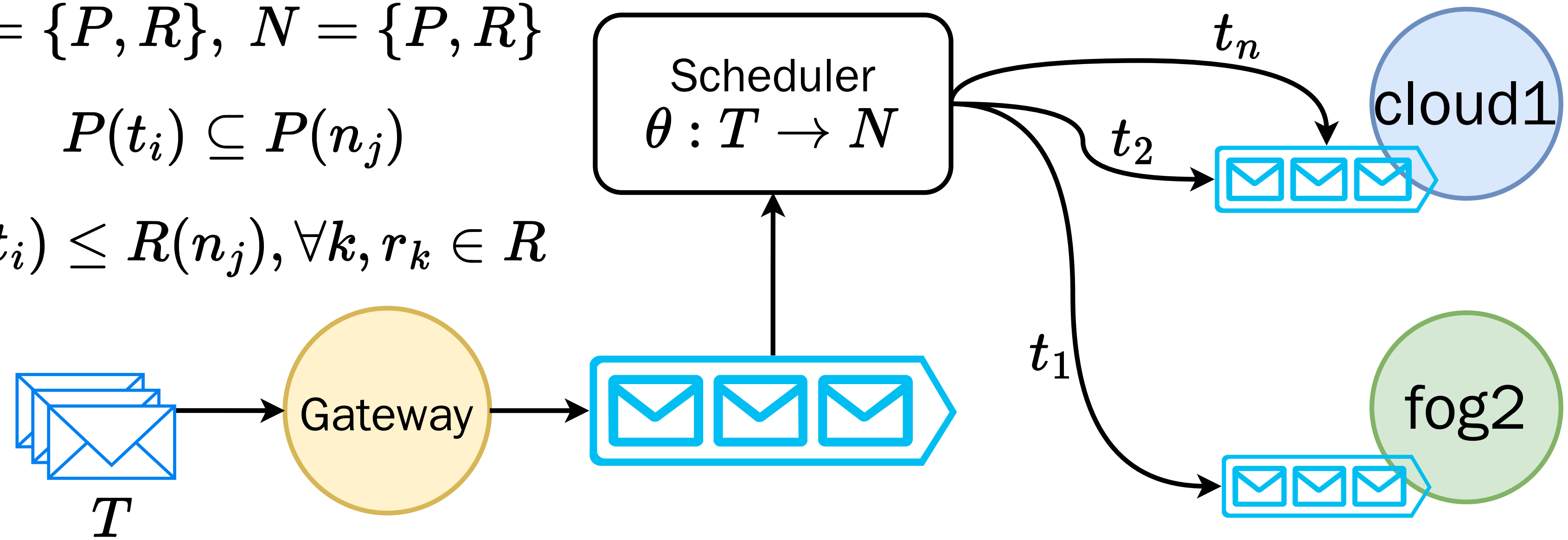
https://github.com/spboehm/pulceo-misc

Task Scheduling / Offloading

$$T = \{P, R\},\ N = \{P, R\}$$

$$P(t_i) \subseteq P(n_j)$$

$$R(t_i) \leq R(n_j), \forall k, r_k \in R$$



- IoT / mobile clients, submitting independent / atomic tasks T
- Heterogeneous cloud, edge and fog nodes N
- Tasks T and Nodes N are described by
  - Properties P: Layer, Location, Costs, …
  - Requirements R: CPU, Memory, Bandwidth, …

Motivation

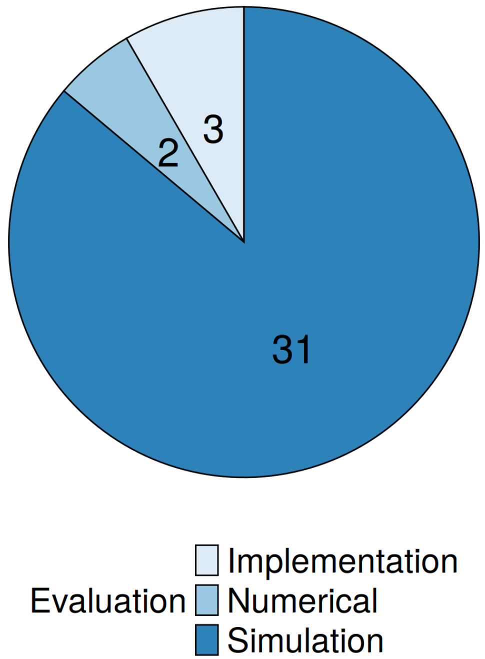
Many solutions exist for **task scheduling / offloading**

But, similar to **Service Placement**:

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

are **limited** because of:

- **simulation-based** evaluations
- **custom** implementations
- missing **real-world experiments**



Only **3** out of **36** task offloading solutions were using real-world evaluations.[1]

1. S. Dong et al., "Task offloading strategies for mobile edge computing: A survey," Computer Networks, vol. 254, p. 110791, Dec. 2024, doi: 10.1016/j.comnet.2024.110791.

Simulations are sufficient?

Selected limitations, addressed by the **authors** of selected solutions:

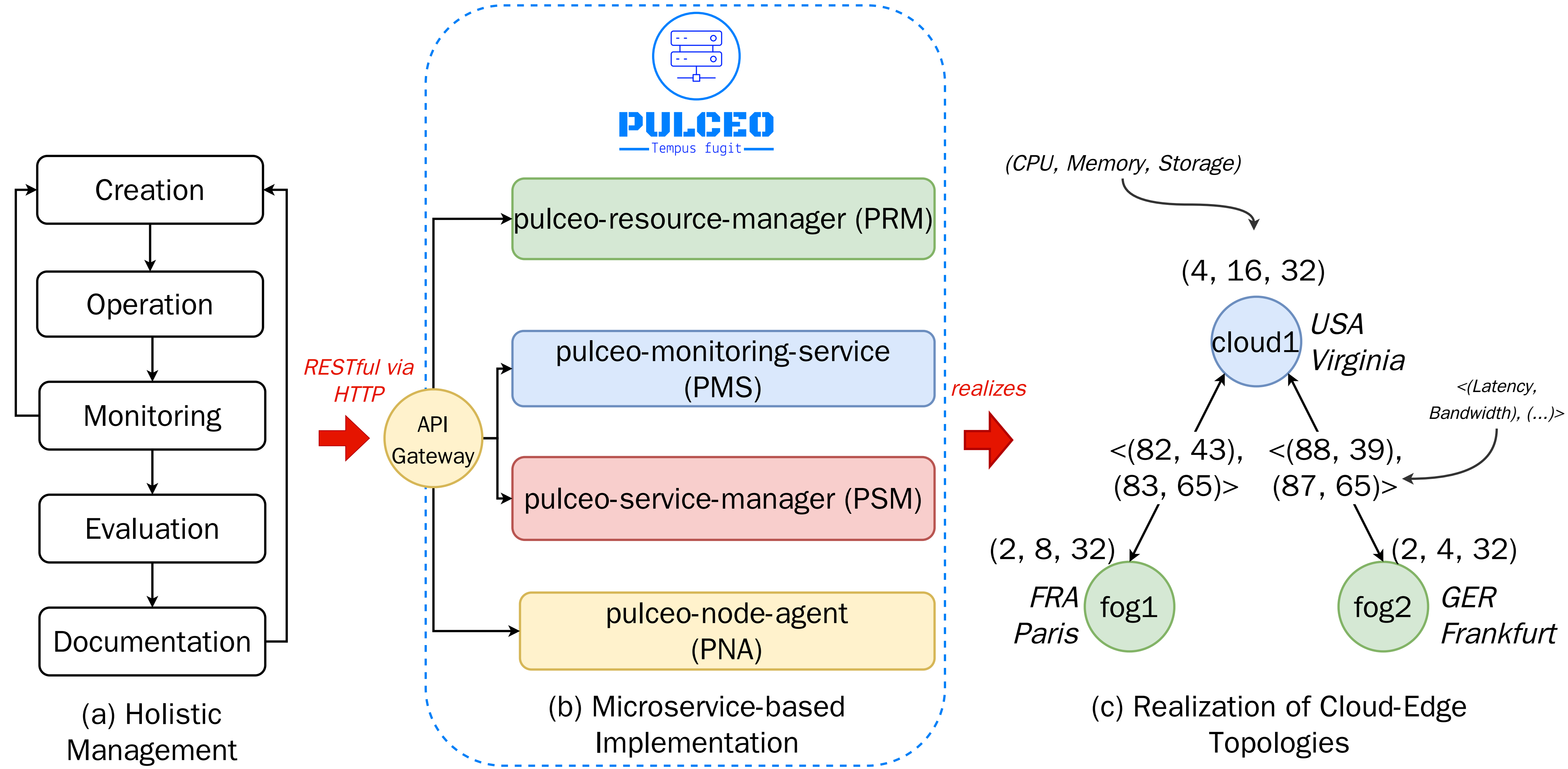
> Where the value of **CPU weight** and **memory weight** is an **empirical issue**, and **multiple experiments are required** to obtain the optimal value such that $\alpha + \beta = 1$. In this paper, all parameters are set to appropriate values.[1]

> Phare [the scheduling solution] requires updated information on the clusters participating in the federation, including **CPU**, **memory**, and **bandwidth usage**. A **real-world implementation** should thus **carefully balance** the resolution of such data and the additional overhead required to process it.[2]

Consequently, there is a need for **real-world experiments** and **evaluations**.

1. Y. Dong, G. Xu, M. Zhang, and X. Meng, “A High-Efficient Joint ’Cloud-Edge’ Aware Strategy for Task Deployment and Load Balancing,” IEEE Access, vol. 9, pp. 12791–12802, 2021, doi: 10.1109/ACCESS.2021.3051672.
2. G. Castellano, S. Galantino, F. Risso, and A. Manzalini, “Scheduling Multi-Component Applications Across Federated Edge Clusters With Phare,” IEEE Open J. Commun. Soc., vol. 5, pp. 1814–1826, 2024, doi: 10.1109/OJCOMS.2024.3377917.

Holistic Cloud-Edge Orchestration



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

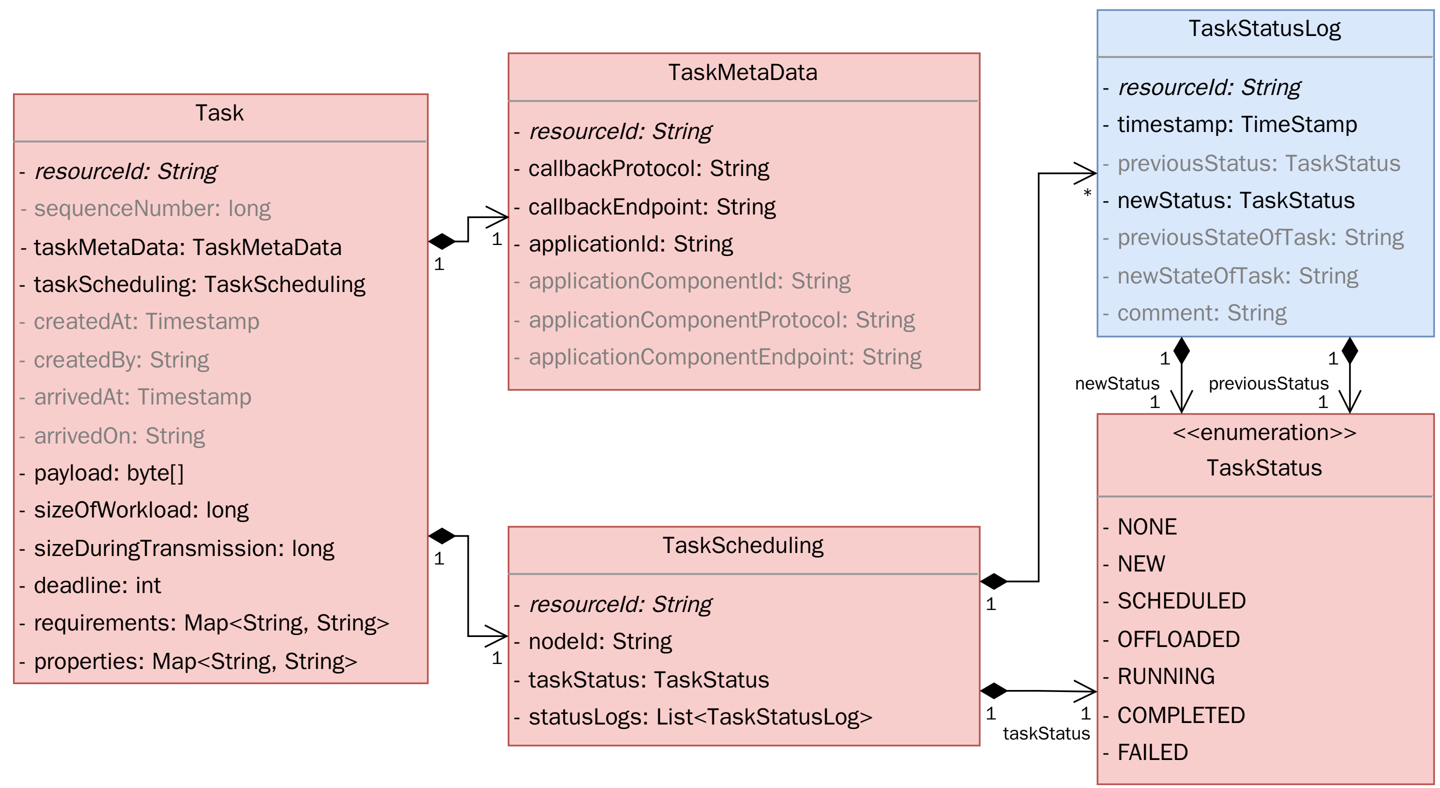
Contributions

1. Task scheduling and offloading in line with holistic cloud-edge orchestration.
2. A general, universal, and flexible model of tasks.
3. Integration of task-processing applications.
4. Automated standard evaluation with metrics for task scheduling and offloading.

Task: Domain Model

Extension of step **Operation**, supported by **p**ulceo-**s**ervice-**m**anager (**PSM**).

- *Task* with fixed / variable
  - requirements R
  - properties T
- *TaskMetaData*: Internal routing information
- *TaskScheduling*: Current state of scheduling
- *TaskStatusLog*: State transitions over time

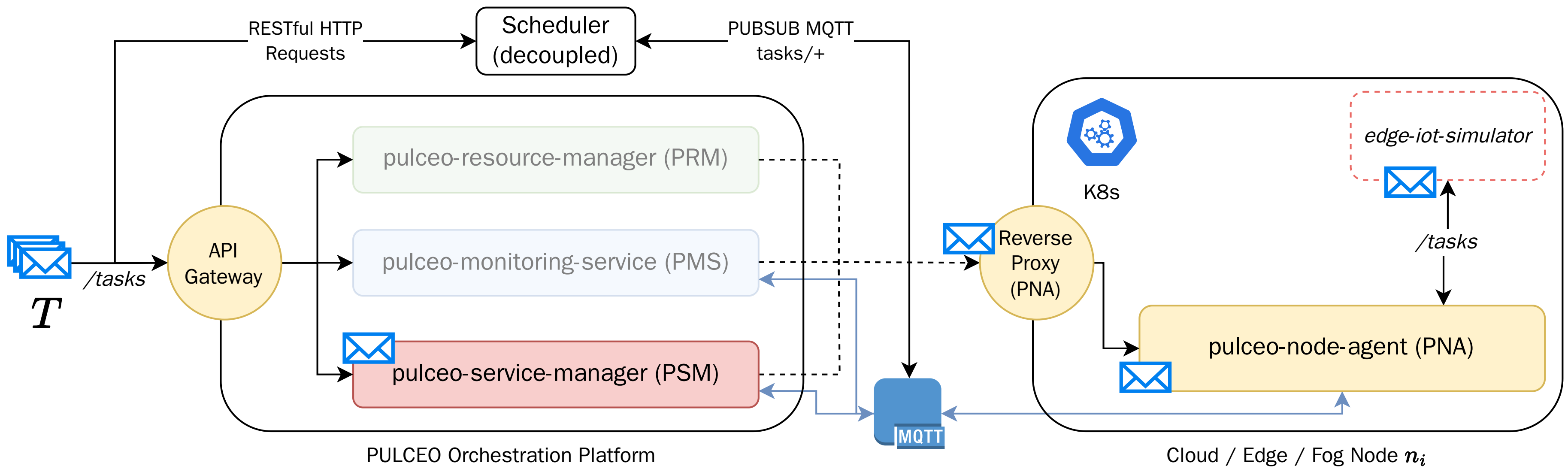


Example: NONE -> NEW -> SCHEDULED -> OFFLOADED -> RUNNING -> COMPLETED

Orchestration Workflow

Experiment · Task Submission, Scheduling, Offloading, Processing, and Completion

Experiment



- Decoupled scheduler uses HTTP (`/tasks`) and MQTT (`tasks/+`)
- $P_{task\_type}(t_i) = S\,(50\%), M\,(30\%), L\,(20\%) \qquad batch\_sizes = 200, \ldots, 400$
- $P_{size}(t_i) = 10, 50, 100 \quad \ldots \qquad R_{cpu\_shares}(t_i) = 250, 500, 1000 \quad \ldots$
- Instance of edge-iot-simulator[1] used for task processing

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

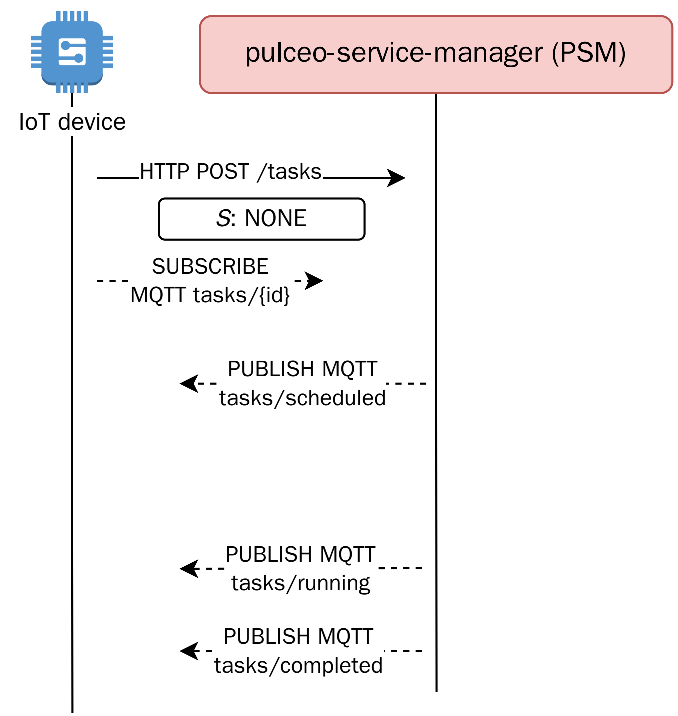
Task Submission

Task states: NONE -> NEW -> SCHEDULED -> OFFLOADED -> RUNNING -> COMPLETED

IoT device submits task:

`HTTP POST /tasks`

```json
{
  "createdBy": "task_emitter.py",
  "sizeOfWorkload": 50,
  "sizeDuringTransmission": 50,
  "deadline": 50,
  "payload_length": 50,
  "payload": "payload",
  "cpu_shares": 500,
  "memory_size": 0.25,
  "properties": {
    "task_type": "small"
  }
 }
```



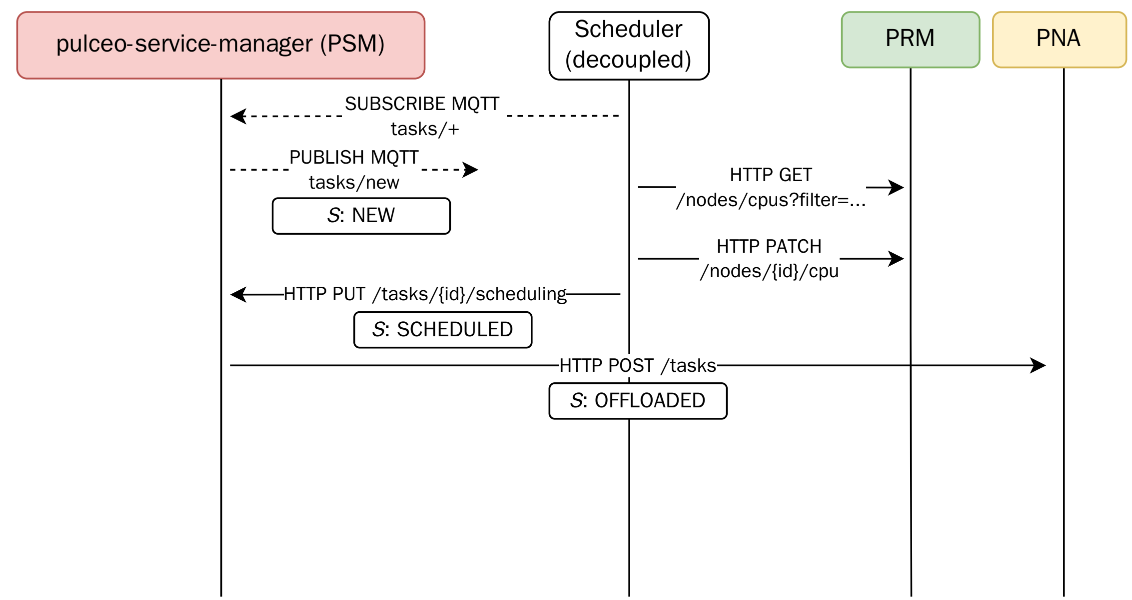
Task Scheduling and Offloading

Task states: NONE -> NEW -> SCHEDULED -> OFFLOADED -> RUNNING -> COMPLETED

Scheduler assigns node:

`HTTP PUT /tasks/{id}/scheduling`

```json
{
  "nodeId": "0b1c6697-...",
  "appId": "74e18419-...",
  "status": "SCHEDULED",
  "properties": {
    "batchSize": "100",
    "layer": "cloud-only",
    "policy": "dong et al."
  }
}
```



PSM offloads task:

`HTTP POST /tasks` (on PNA)

```json
{
  "taskId": "831de05b-...",
  "appId": "74e18419-...",
  "appComponentId": "6a463804-...",
  ...
}
```

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

Task Processing

Task states: NONE -> NEW -> SCHEDULED -> OFFLOADED -> RUNNING -> COMPLETED

Forward task to application:

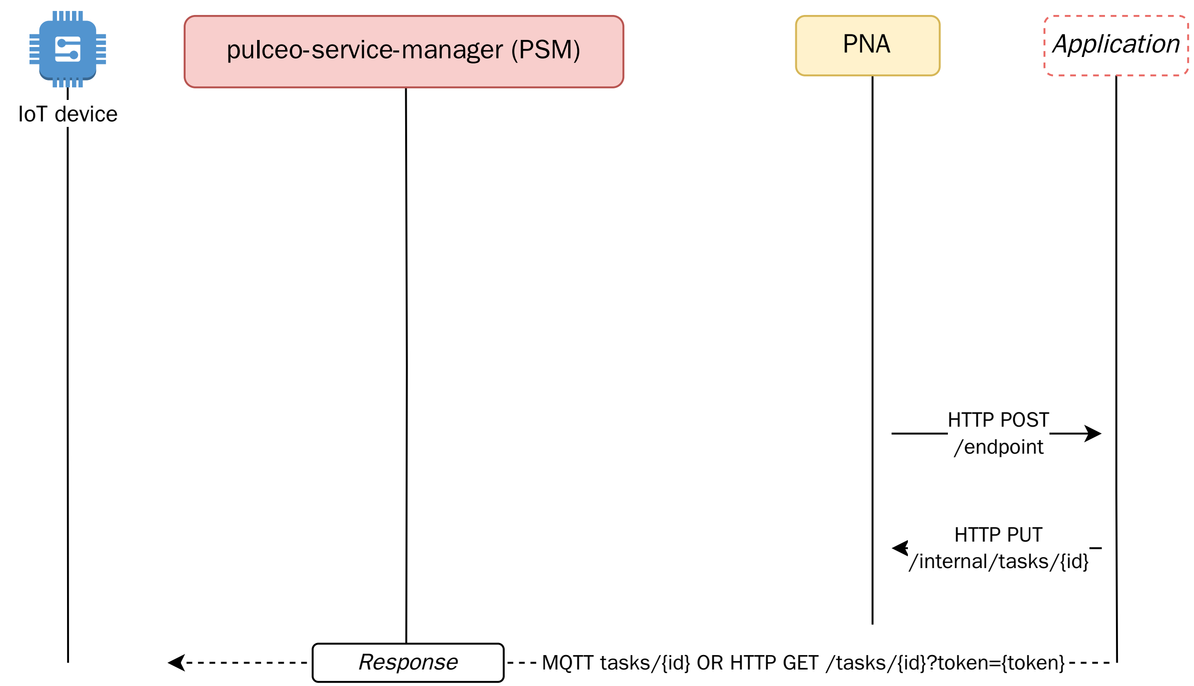
`HTTP POST /endpoint`

```
{
  "taskId": "831de05b-...",
  "callbackProtocol": "0b1c6697-...",
  "callbackEndpoint": "74e18419-...",
  ...
}
```

Update task status internally:

`HTTP PUT /internal/tasks/{id}`

```
{
  "taskId": "831de05b-...",
  "newTaskStatus": "COMPLETED"
}
```

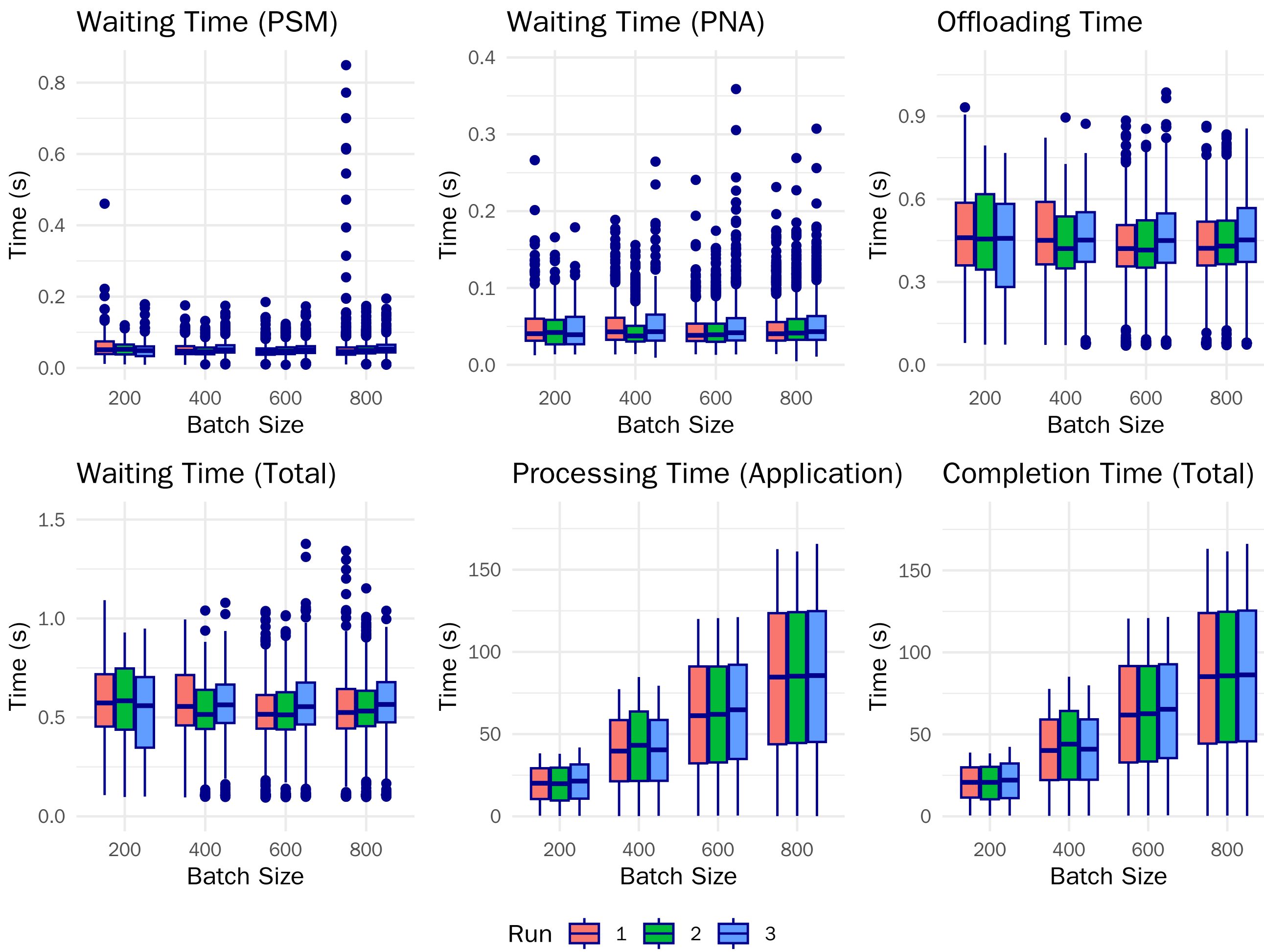


Evaluation

Task Metrics · Performance Metrics

- Provided by the automatically generated orchestration reports
- On top of already implemented metrics, like CPU, memory, and network-related metrics

Task Metrics



Drill-down into overall completion time:

- Scheduling (PSM): 0.05s
- Waiting (PNA): 0.05s
- Offloading: 0.44s
- Waiting (Total): 0.54s

Overhead:

- Communication (HTTP)
- Telemetry
- Proxied application

Performance Metrics

- **Task Response Time:** Time between submitting the task and receiving the response
- **Task Arrival Rate:** Tasks/s arriving at PSM
- **Task Throughput:** Tasks/s completed by the system

Conclusion & Limitations

Contributions

1. 🚀 Task scheduling and offloading in line with holistic cloud-edge orchestration
2. 🌐 A general, universal, and flexible model of tasks
3. ✅ Integration of task-processing applications
4. 🔄 Automated standard evaluation with metrics for task scheduling and offloading

Empirical evaluation with first insights on the overall overhead.

Limitations

- 🚨 Integration of task-processing applications only via HTTP and MQTT
- 🚨 Centralized, cloud-based, and only partly decentralized orchestration
- 🚨 Minimal cloud-edge topology for first evaluation