TECHNISCHE UNIVERSITÄT MÜNCHEN

Oakestra

A Hierarchical Orchestrator for Edge Computing

Nitinder Mohan

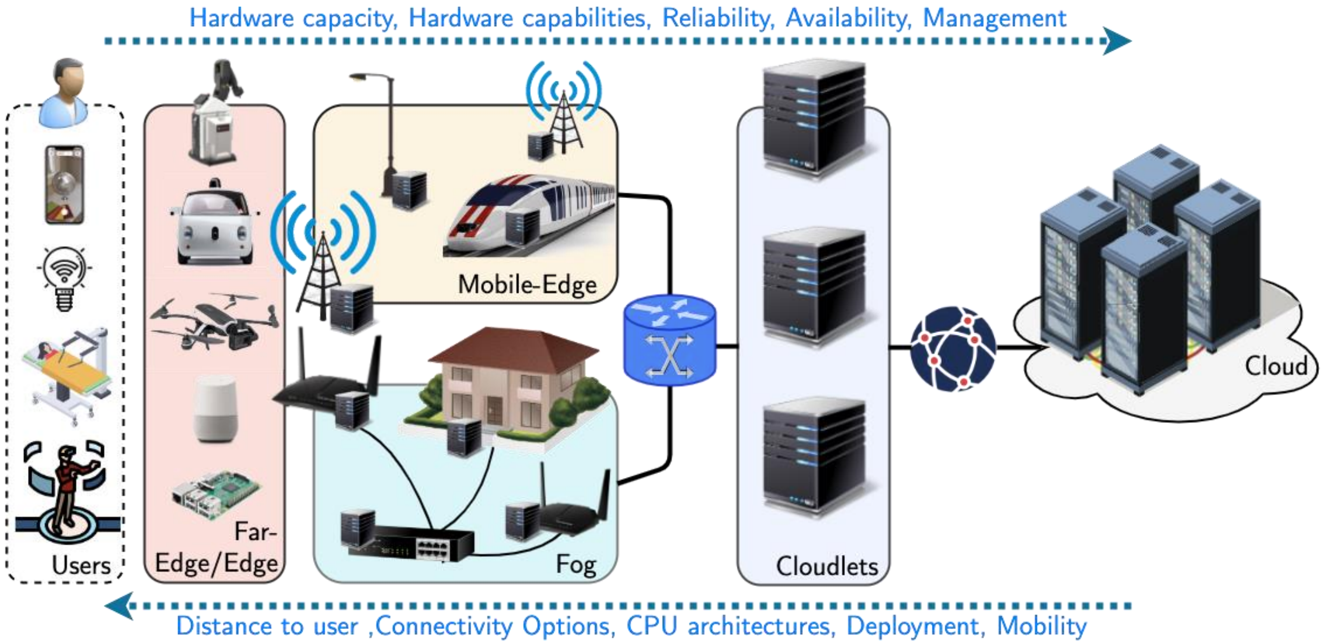
mohan@in.tum.de

nitinder_mohan

Problems with Edge Orchestration

Edge infrastructure can be widely different than cloud

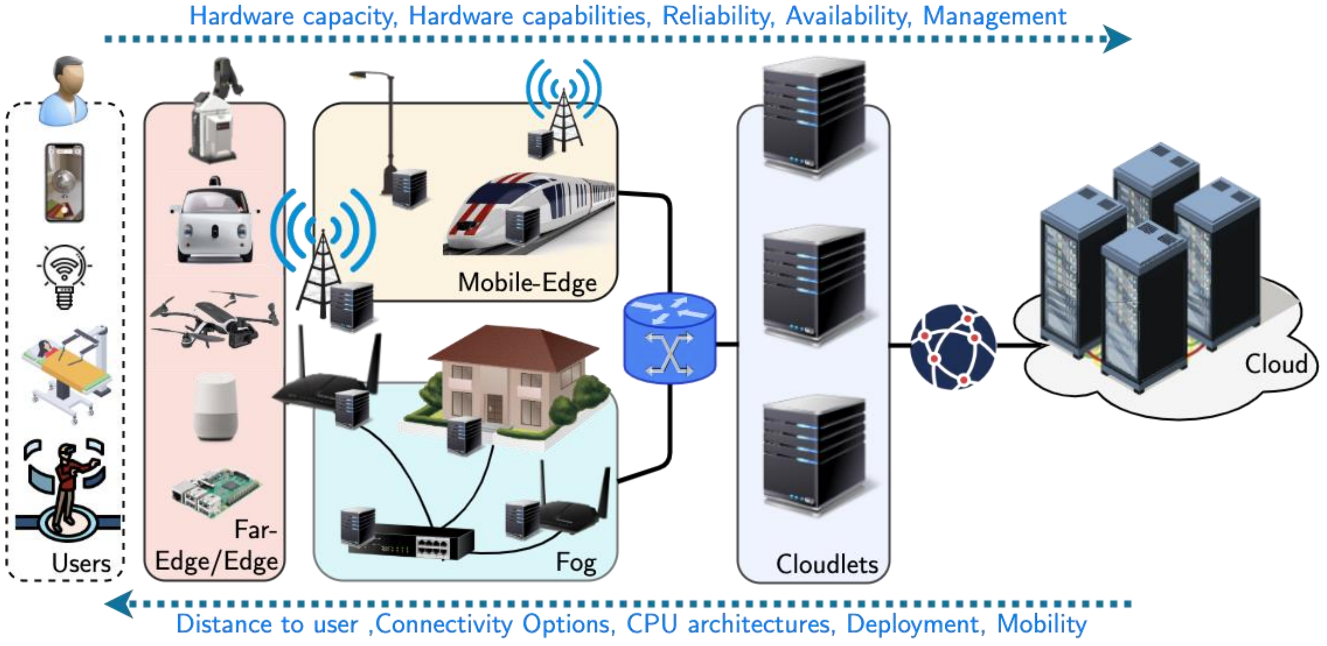
- Hardware can be specialized, heterogeneous and constrained

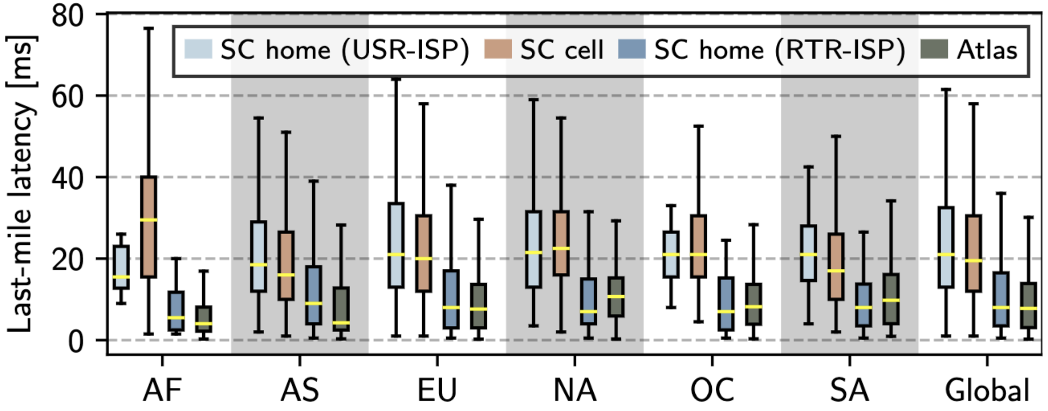

Hardware capacity, Hardware capabilities, Reliability, Availability, Management
Users
Far-Edge/Edge
Mobile-Edge
Fog
Cloudlets
Cloud
Distance to user ,Connectivity Options, CPU architectures, Deployment, Mobility

Problems with Edge Orchestration

Edge infrastructure can be widely different than cloud

- Hardware can be specialized, heterogeneous and constrained
- Network can be limited with fluctuations


Hardware capacity, Hardware capabilities, Reliability, Availability, Management
Users
Far-Edge/Edge
Mobile-Edge
Fog
Cloudlets
Cloud
Distance to user ,Connectivity Options, CPU architectures, Deployment, Mobility

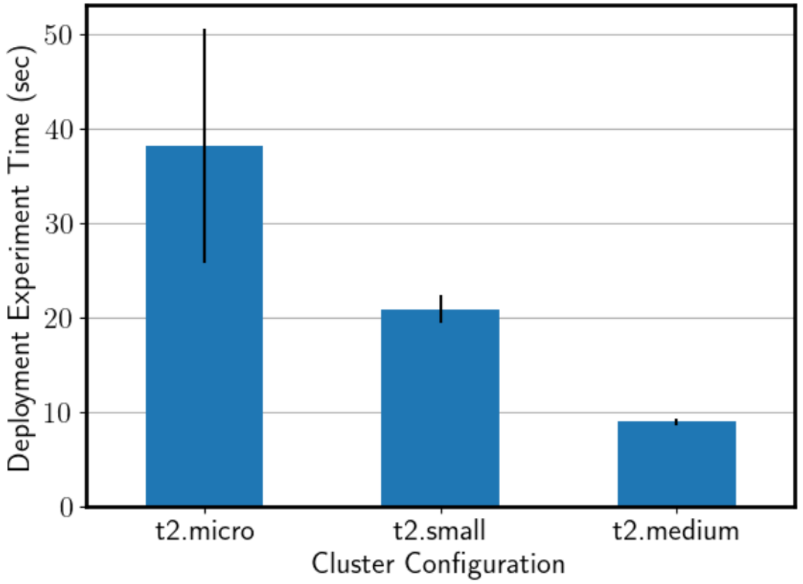


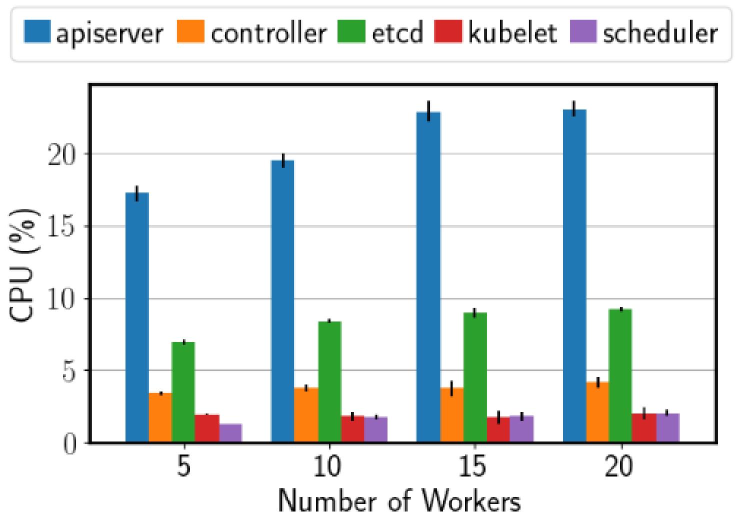
The Khang Dang, Nitinder Mohan, Lorenzo Corneo, Aleksandr Zavodovski, Jörg Ott, and Jussi Kangasharju. 2021. Cloudy with a chance of short RTTs: analyzing cloud connectivity in the internet. IMC 2021

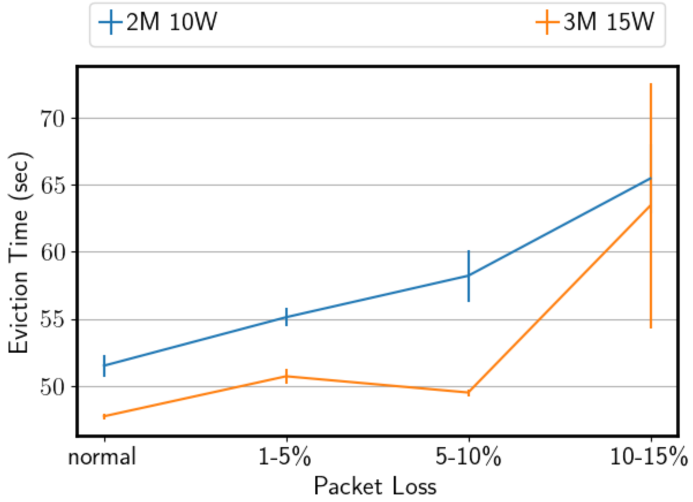
Problems with Edge Orchestration

Many orchestration already exist …

But most are not designed for edge







Sonia Klärmann. *Evaluating the Suitability of Kubernetes for Edge Computing Infrastructure.,* M.Sc. Thesis, TUM

Andrew Jeffery, Heidi Howard, and Richard Mortier. 2021. Rearchitecting Kubernetes for the Edge. EdgeSys '21.

Oakestra

Hierarchical decentralized management

Scalable delegated service scheduling and placement

Modular and extensible design

Lightweight edge device specific implementation

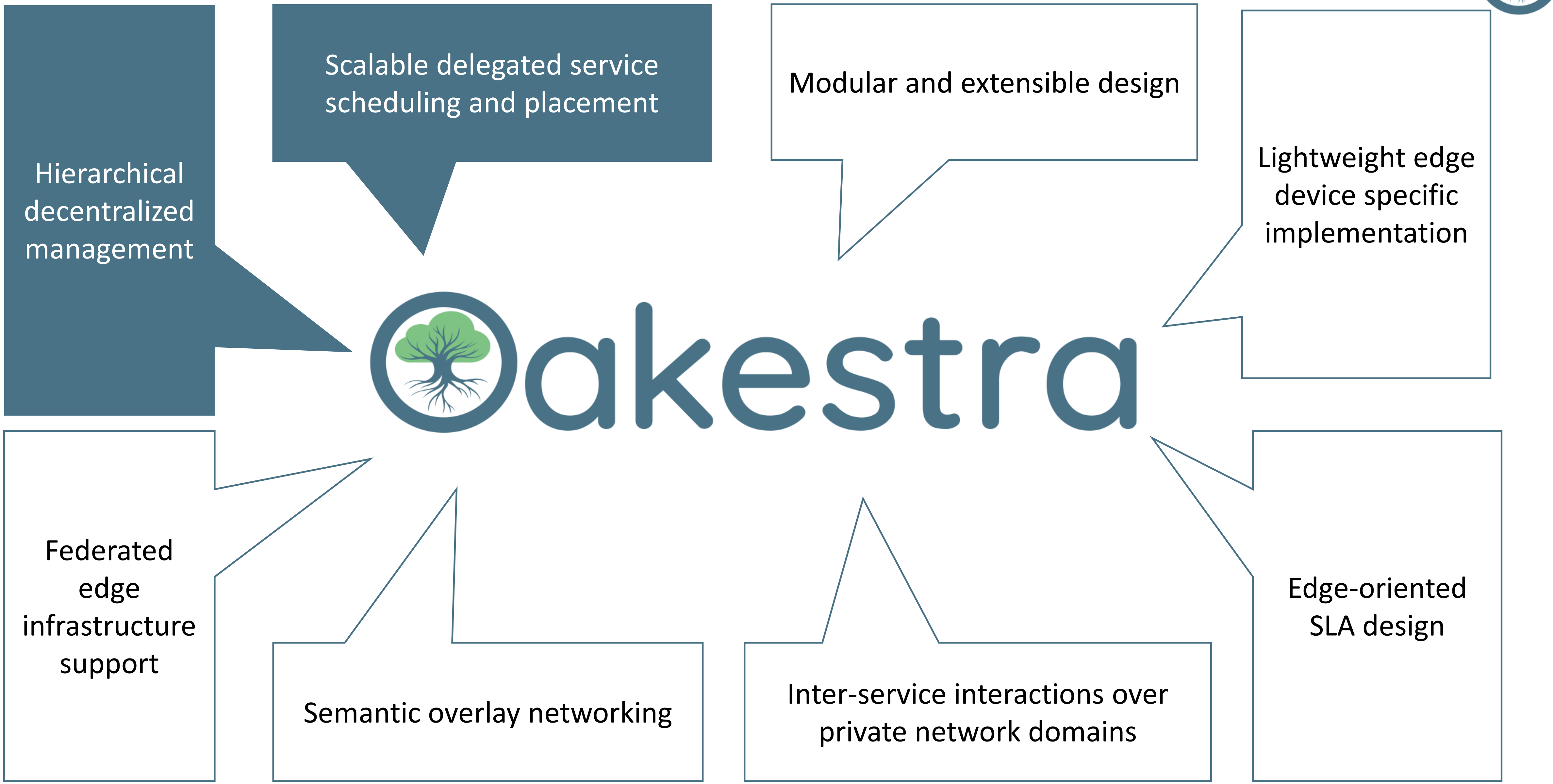
Federated edge infrastructure support

Semantic overlay networking

Inter-service interactions over private network domains

Edge-oriented SLA design

… and more



Oakestra

Hierarchical decentralized management

Scalable delegated service scheduling and placement

Modular and extensible design

Lightweight edge device specific implementation

Federated edge infrastructure support

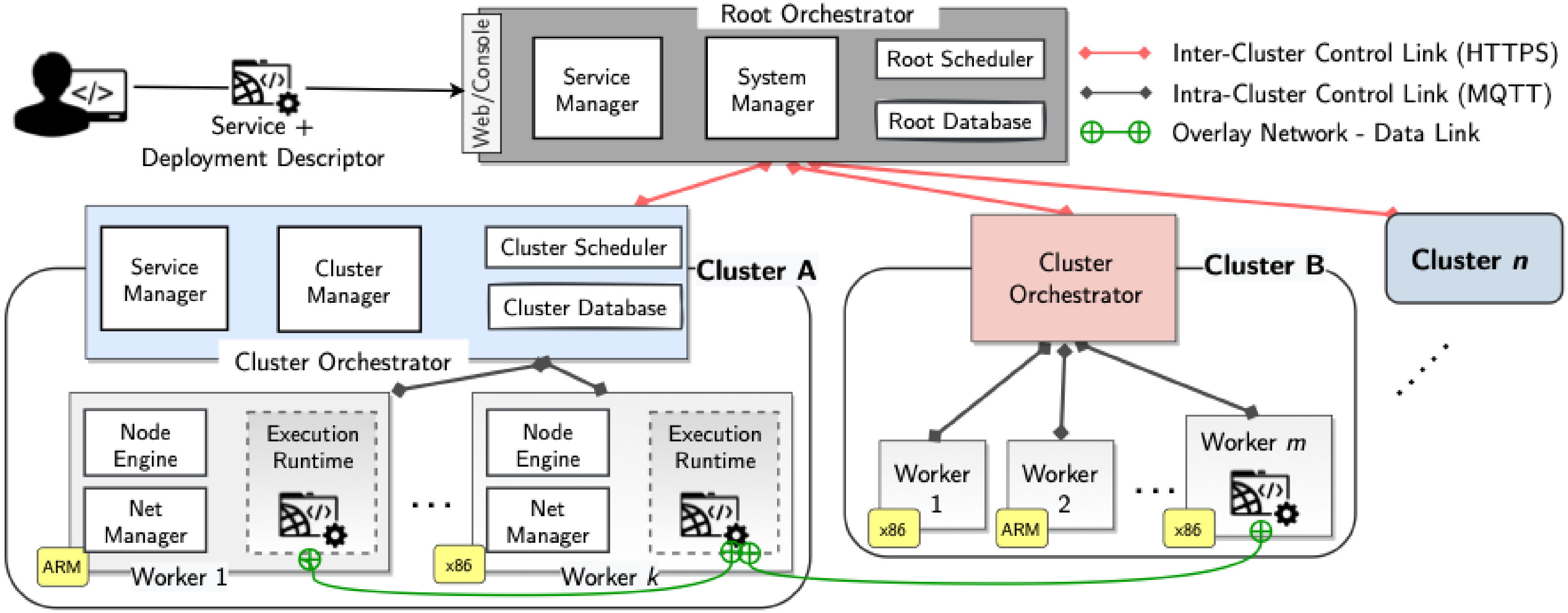
Semantic overlay networking

Inter-service interactions over private network domains

Edge-oriented SLA design

… and more

Oakestra: Overview



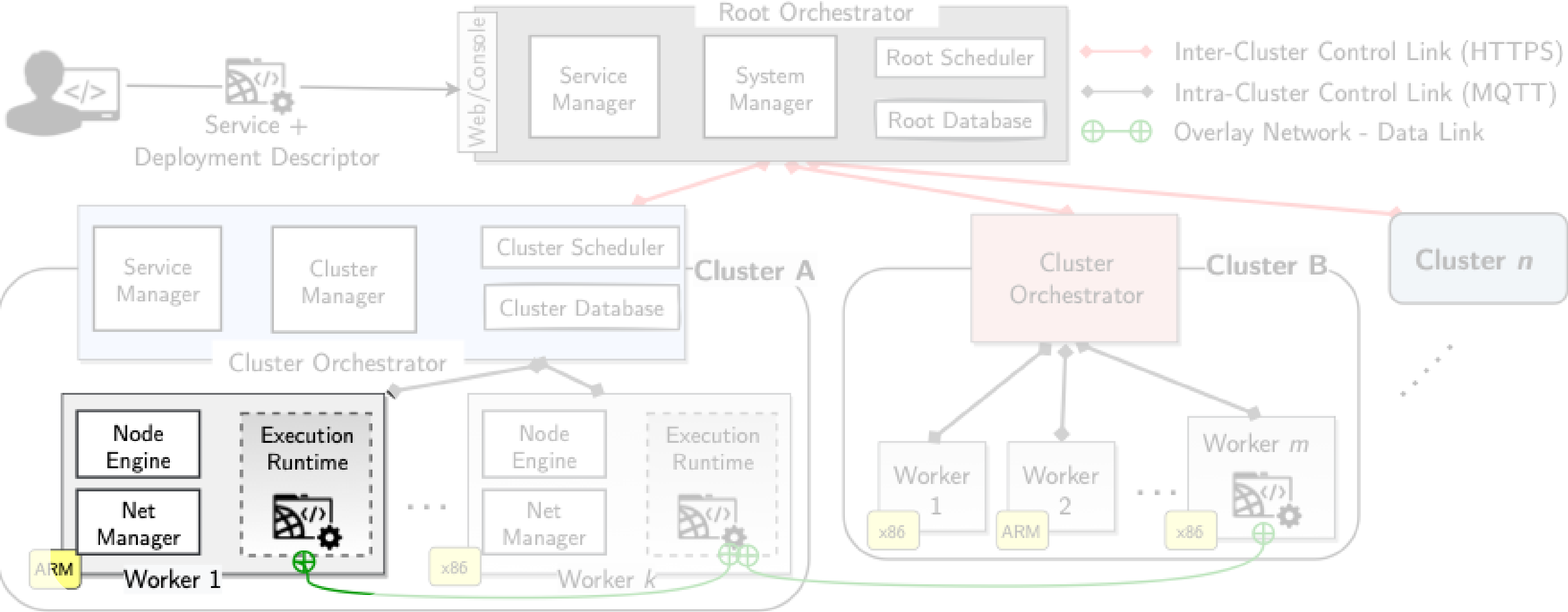
Edge servers organized as different clusters

- Different operators can set up different clusters for a **federated** infrastructure
- Each cluster has its own orchestrator
- Clusters can be private networks

Root Orchestrator Coarse-Grained management over clusters

Cluster Orchestrator Fine-Grained management over compute resources

Oakestra: Composers



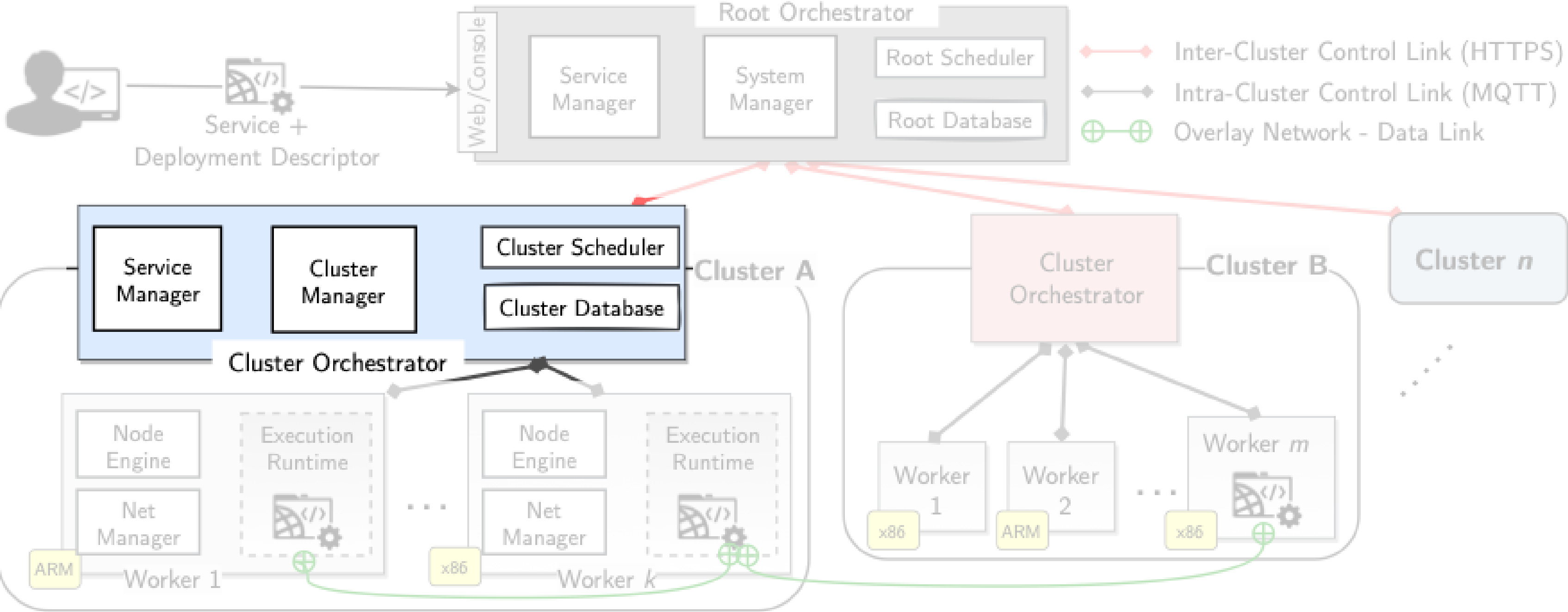
- Each worker has distinct capability. e.g. CPU, GPU, MEM, etc.
- Can have different architectures, x86, ARM

Node Engine Deploying, monitoring, managing services + reporting to orch.

NetManager Inter-service communication across different network domains

Sends frequent resource and service usage information to cluster orchestrator

Oakestra: Composers



- Manages all cluster workers using MQTT communication channel

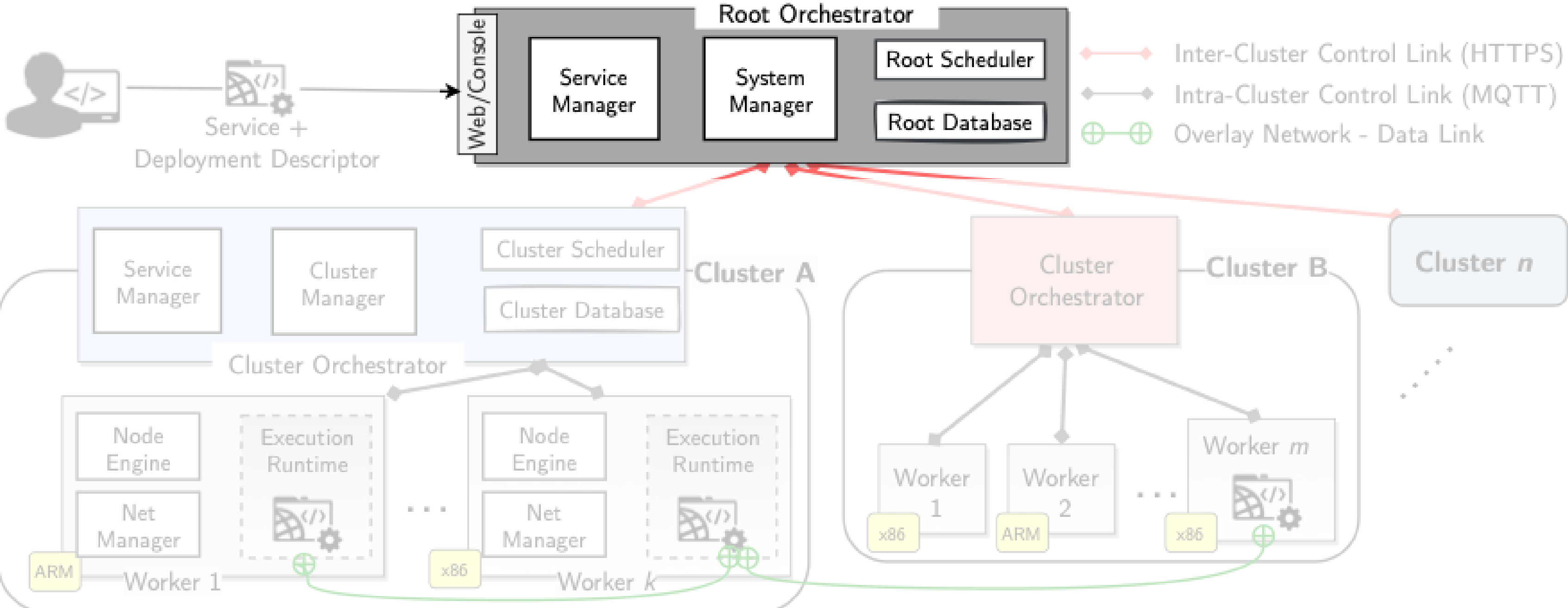
Cluster Manager Monitors and orchestrates workers (incl. failures)

Service Manager Monitors and orchestrates services (deployment, termination, failure)

ClusterScheduler Finds optimal worker for deploying services. Algorithms are plugins

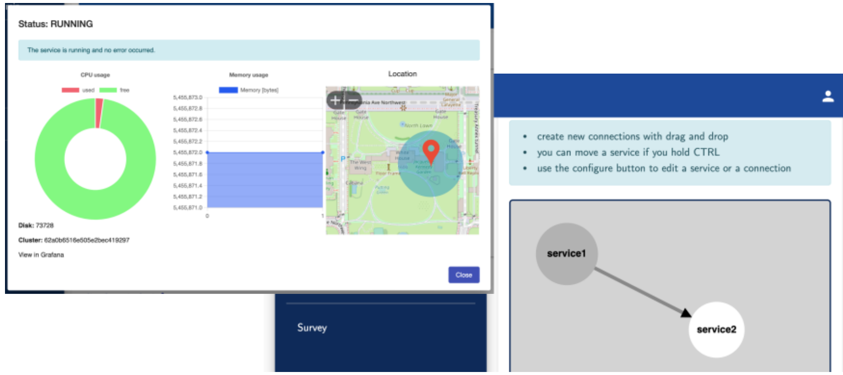
Aggregates resource availability of the cluster and sends it to root at lower frequency

Oakestra: Composers



- “Centralized” control plane that operates at cluster-level [think “master-of-masters”]
- Similar functional components as Cluster Orchestrator

Users can interact via WebUI or CLI



Status: RUNNING

The service is running and no error occurred.

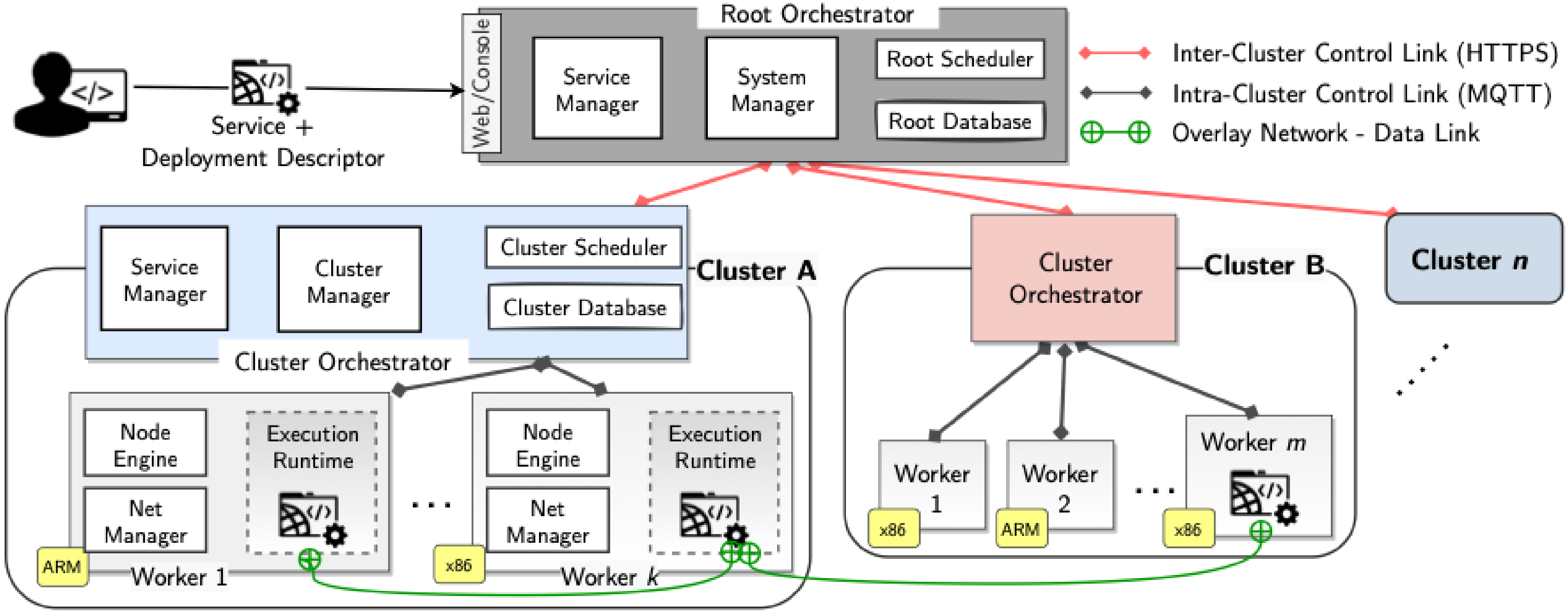
Disk: 73728

Cluster: 62a0b6516e505e2bec419297

View in Grafana

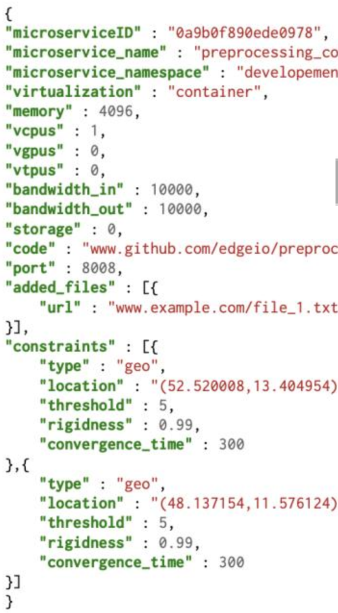
- create new connections with drag and drop
- you can move a service if you hold CTRL
- use the configure button to edit a service or a connection

Oakestra: Scheduling

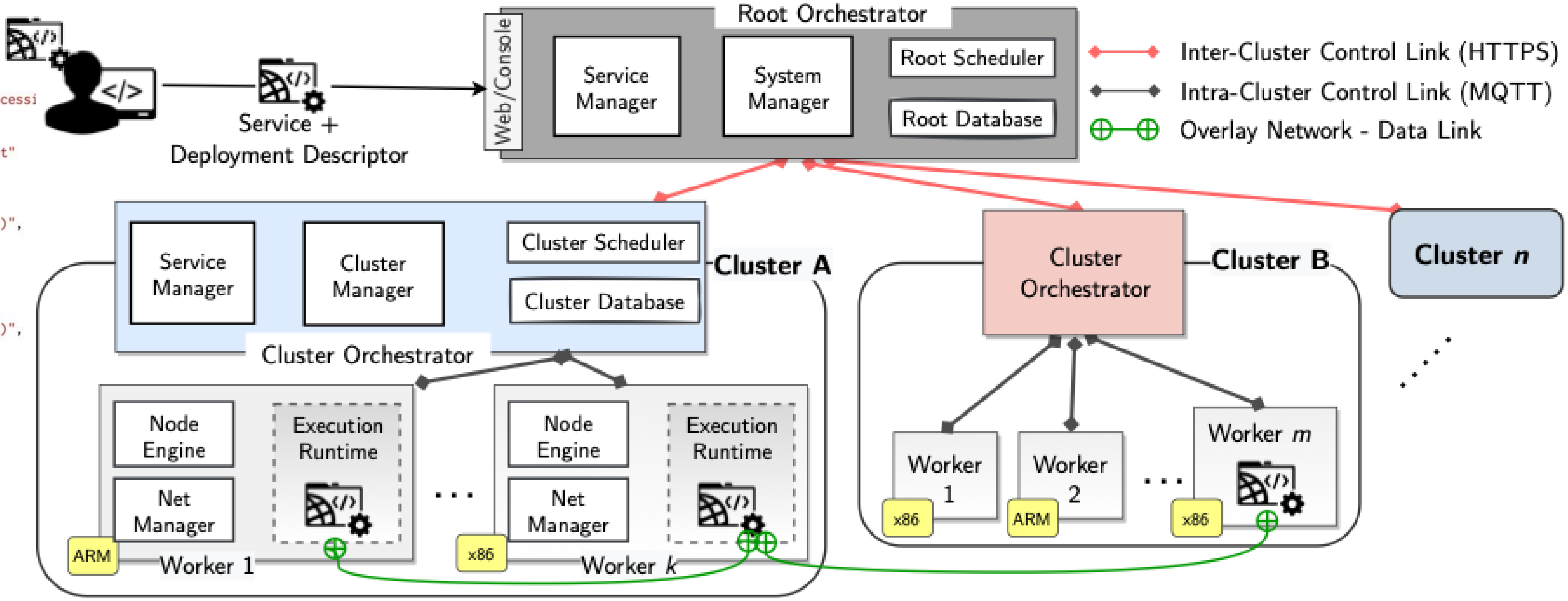


Follows a *delegated* scheduling process

Oakestra: Scheduling



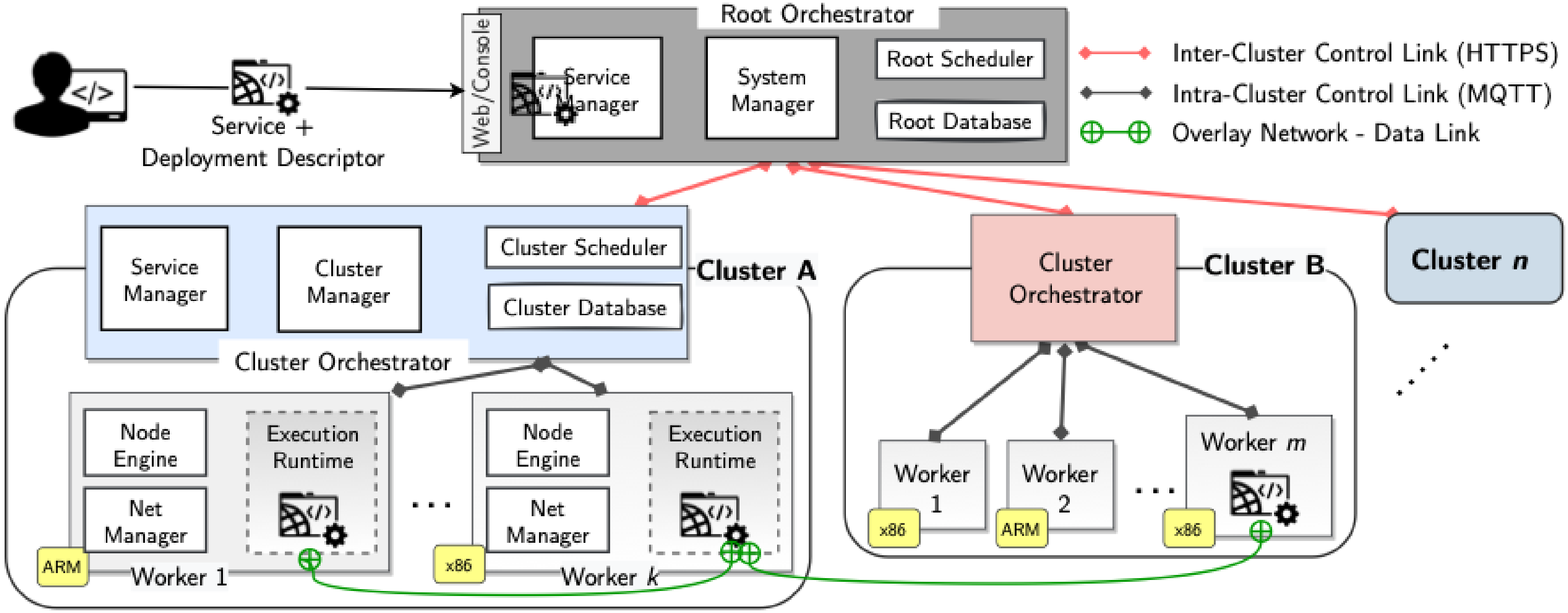
```
{
"microserviceID" : "0a9b0f890ede0978",
"microservice_name" : "preprocessing_collection",
"microservice_namespace" : "developement",
"virtualization" : "container",
"memory" : 4096,
"vcpus" : 1,
"vgpus" : 0,
"vtpus" : 0,
"bandwidth_in" : 10000,
"bandwidth_out" : 10000,
"storage" : 0,
"code" : "www.github.com/edgeio/preprocessi
"port" : 8008,
"added_files" : [{
    "url" : "www.example.com/file_1.txt"
}],
"constraints" : [{
    "type" : "geo",
    "location" : "(52.520008,13.404954)",
    "threshold" : 5,
    "rigidness" : 0.99,
    "convergence_time" : 300
},{
    "type" : "geo",
    "location" : "(48.137154,11.576124)",
    "threshold" : 5,
    "rigidness" : 0.99,
    "convergence_time" : 300
}]
}
```



Follows a *delegated* scheduling process

Step 1: Developer submits application and SLA constraints via web/cli

Oakestra: Scheduling

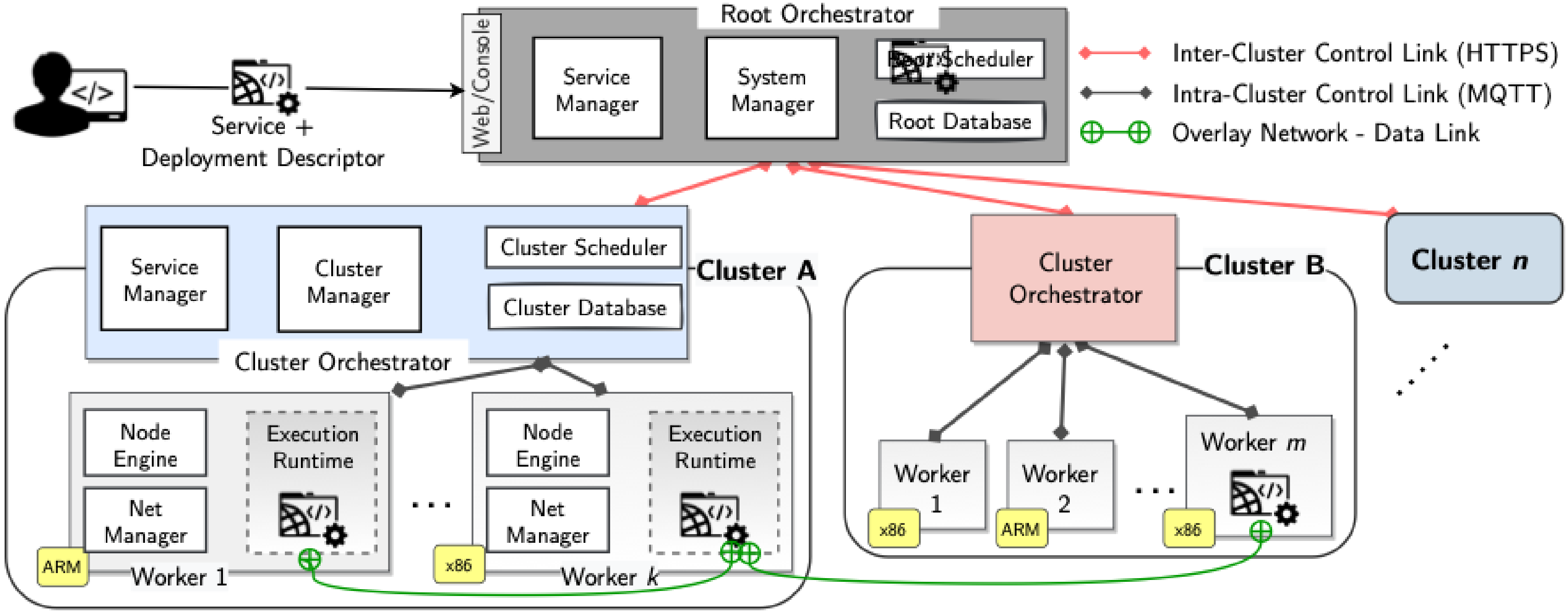


Follows a *delegated* scheduling process

Step 1: Developer submits application and SLA constraints via web/cli

Step 2: Root scheduler calculates “fitting” clusters based on aggregated information

Oakestra: Scheduling



Follows a *delegated* scheduling process

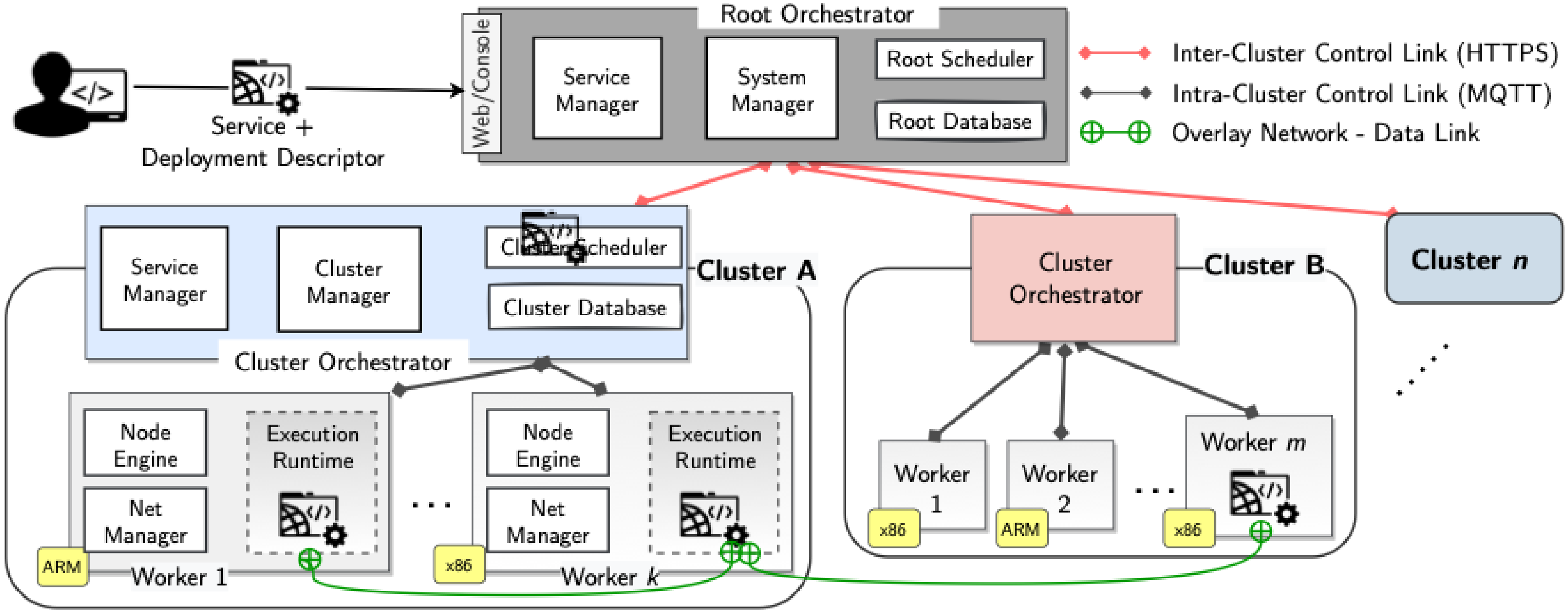
Step 1: Developer submits application and SLA constraints via web/cli

Step 2: Root scheduler calculates “fitting” clusters based on aggregated information

Step 3: Cluster scheduler finds the “optimal” placement for the service within the cluster resources

> Oakestra supports two scheduler plugins: 1) best-fit and 2) latency and geolocation-based

Oakestra: Scheduling



Follows a *delegated* scheduling process

Step 1: Developer submits application and SLA constraints via web/cli

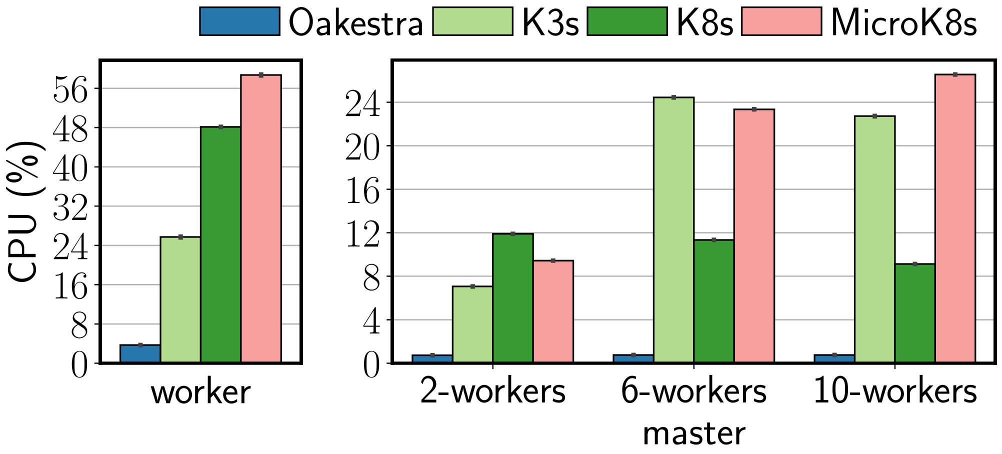
Step 2: Root scheduler calculates “fitting” clusters based on aggregated information

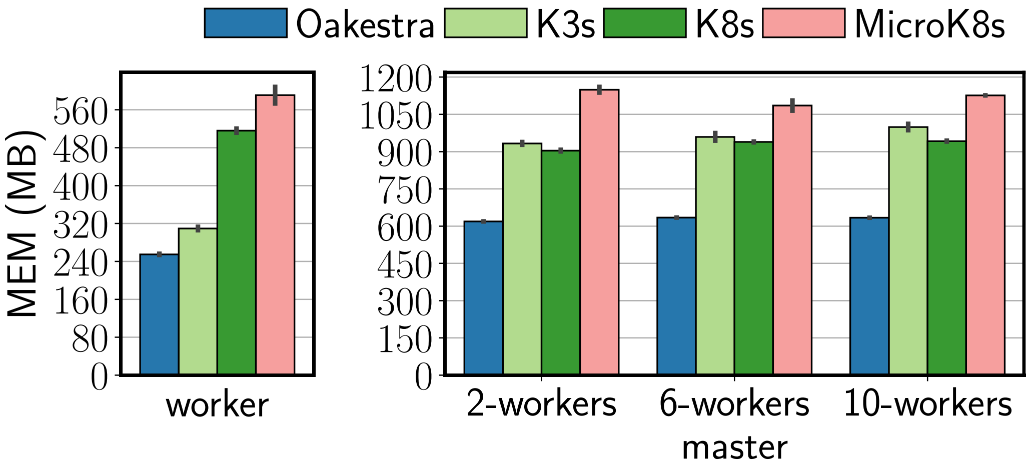
Step 3: Cluster scheduler finds the “optimal” placement for the service within the cluster resources

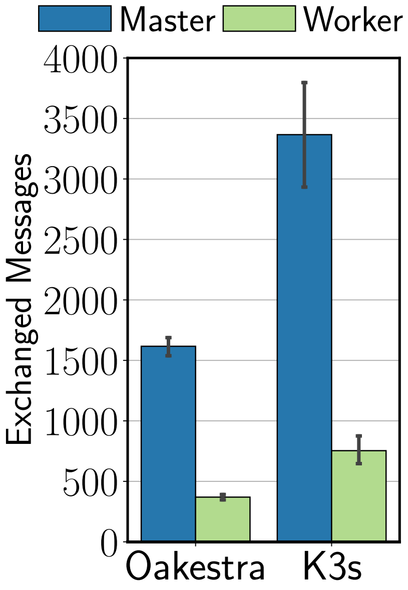
Step 4: Worker nodes accepts/rejects scheduling request and deploys the service

Oakestra in Action

Constrained System Load



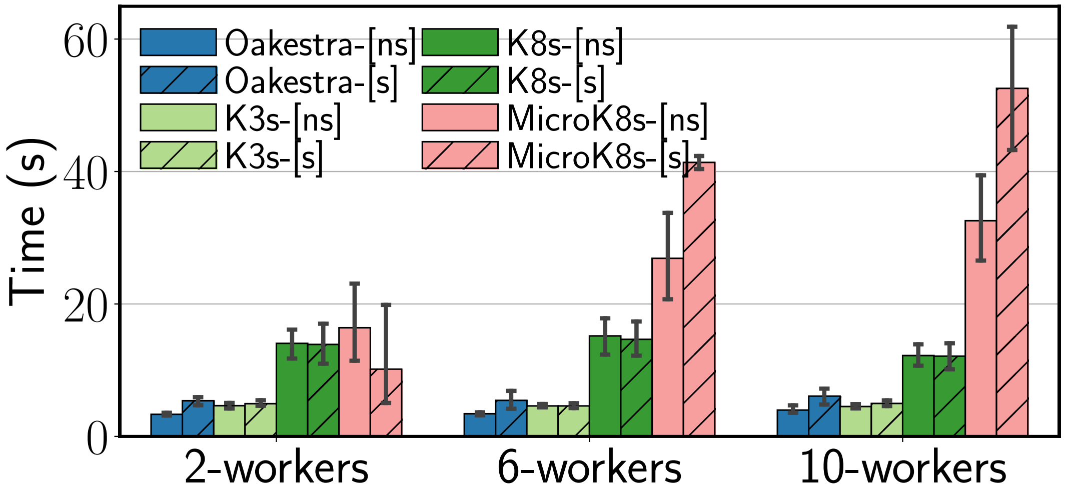


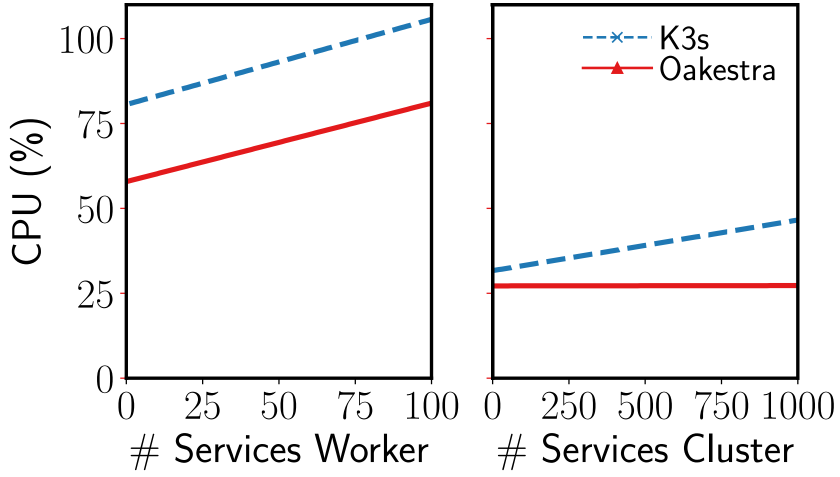


- 6x reduction in Master and 11x reduction in worker CPU usage
- 18% improvement in Master and 33% in worker memory
- 2x reduction in control traffic compared to K3s

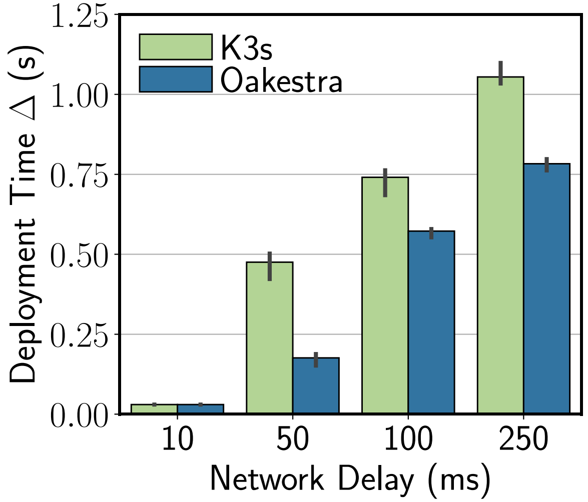
Oakestra in Action

Deployment Time & Scaling





- 10x better than microk8s with scaling deployment
- 20% improvement in scalability than closest competitor: K3s
- Performs significantly better in high delay and lossy networks



Oakestra

Team Credits:

- Giovanni Bartolomeo (PhD)
- Mehdi Yosofie (MSc)
- Oliver Haluszczynsci (MSc)
- Simon Bäurle (MSc)
- Maximilian Eder (MSc)
- Patrick Sabanic (MSc)
- Sonia Klärmann (MSc)
- Ralf Baun (MSc)
- Daniel Mair (BSc)
- Maria Vienalas (BSc)

and Prof. Jörg Ott

Code

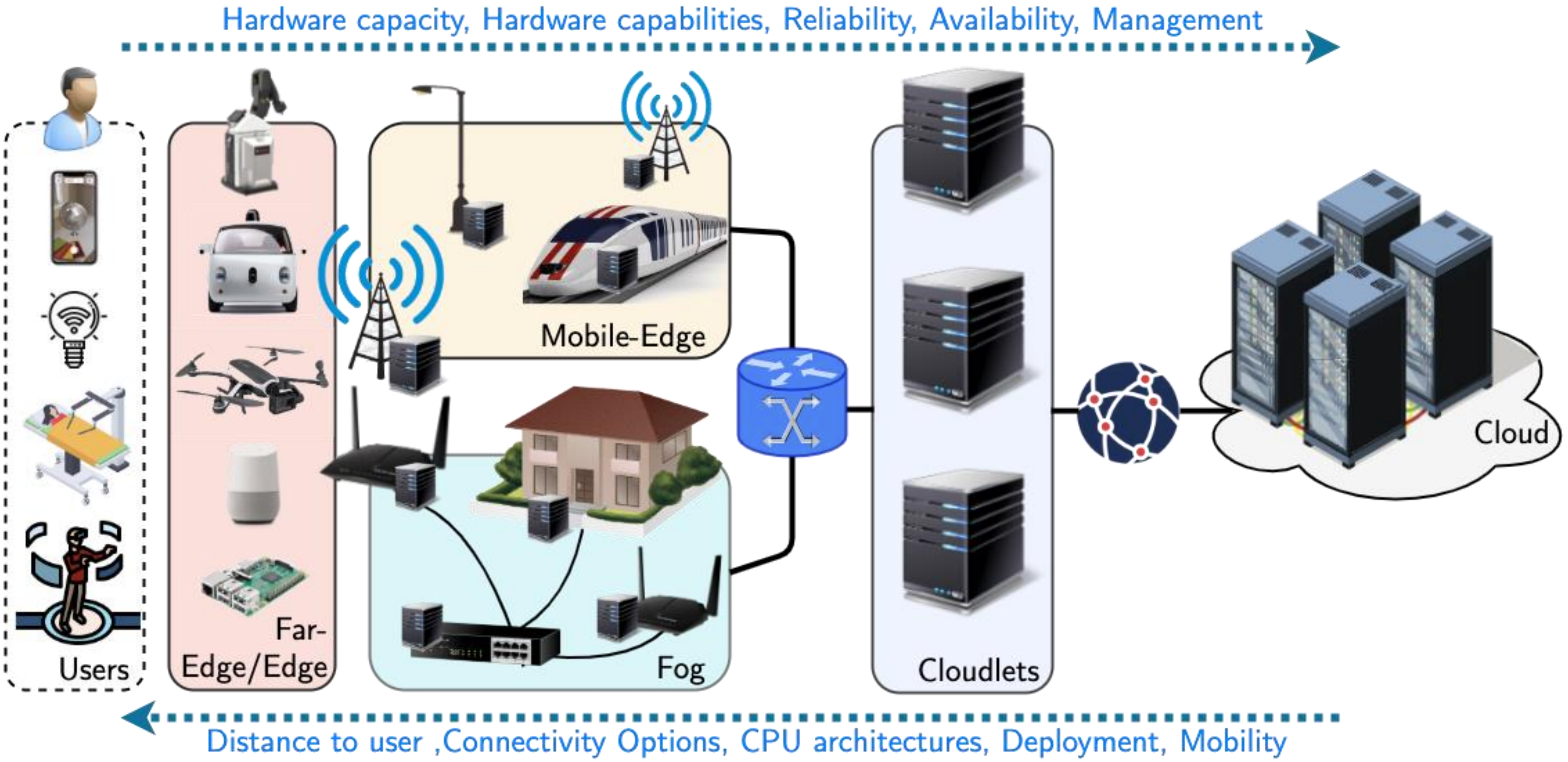
White paper

If you are attending SIGCOMM, check our live demo!

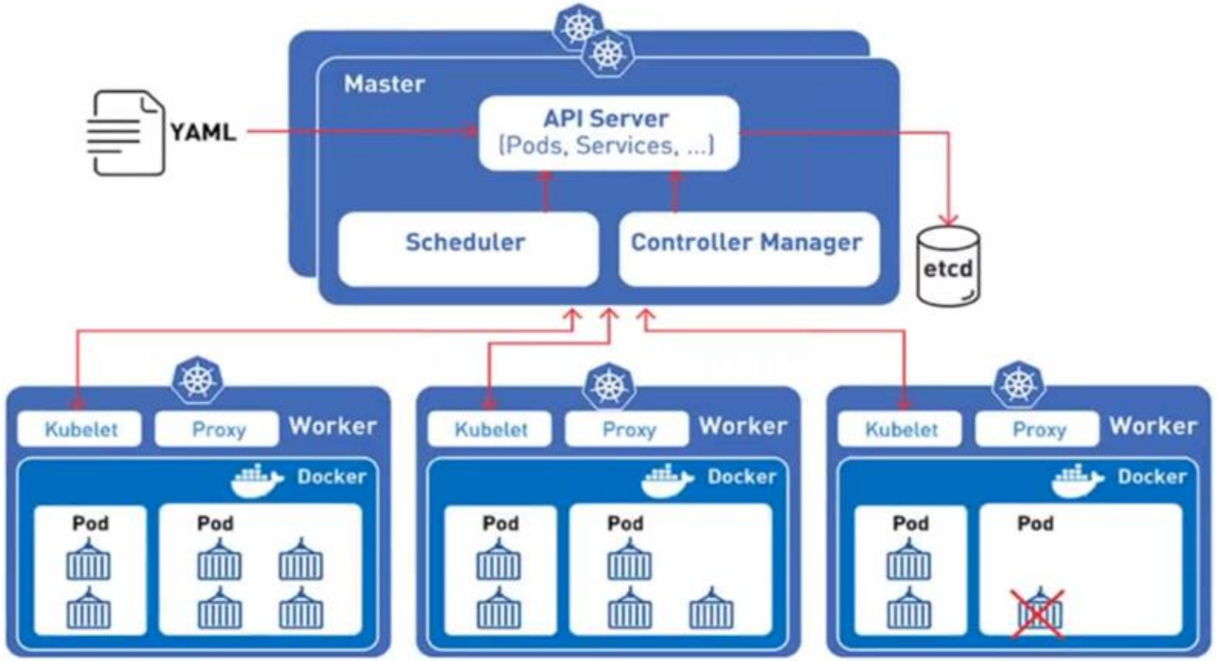
`mohan@in.tum.de`

Backup

A Spectrum of Edge Computing


Hardware capacity, Hardware capabilities, Reliability, Availability, Management
Users
Far-Edge/Edge
Mobile-Edge
Fog
Cloudlets
Cloud
Distance to user ,Connectivity Options, CPU architectures, Deployment, Mobility

Kubernetes at the edge?



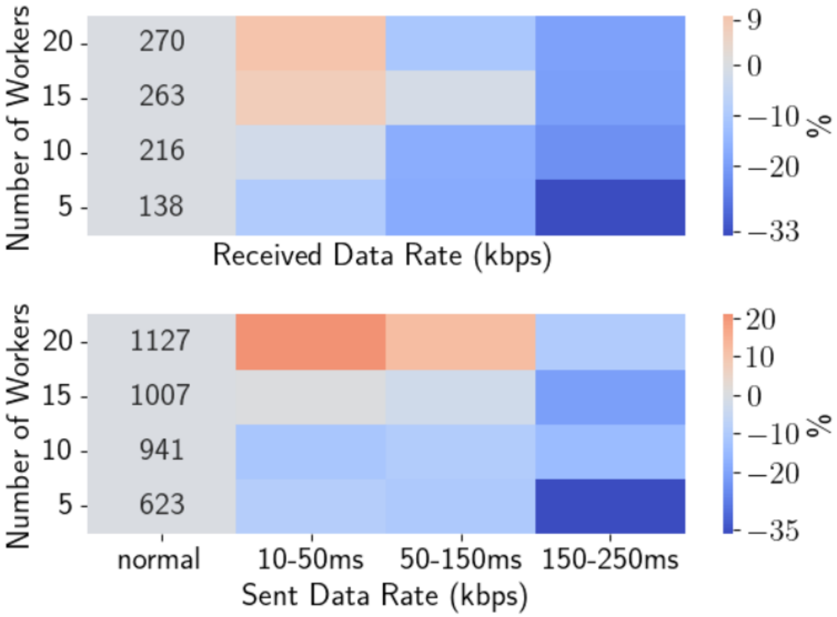
A relatively flat orchestration architecture

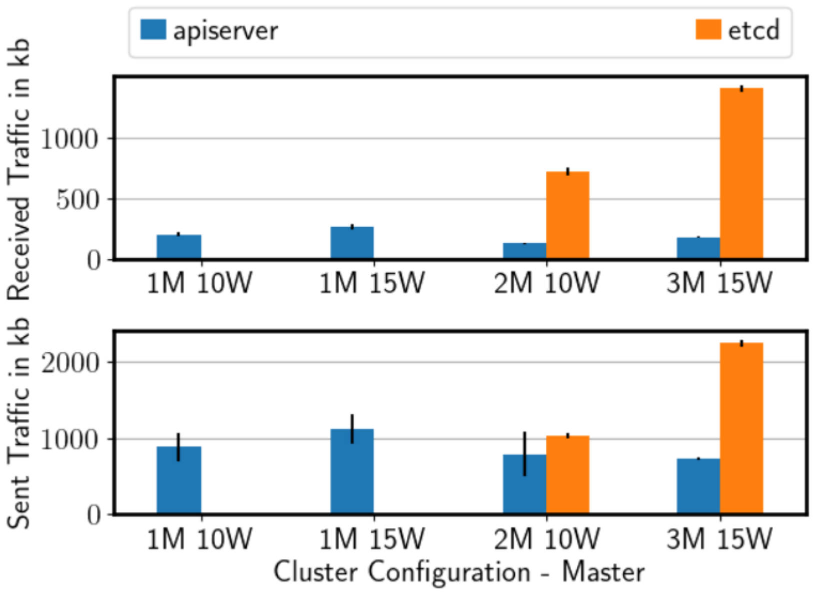
- Compute resources are “Workers”
- Each worker can host multiple “pods” (group of containers)
- Workers are managed by control plane or ”master
- Requires network liveness and strong consistency guarantees

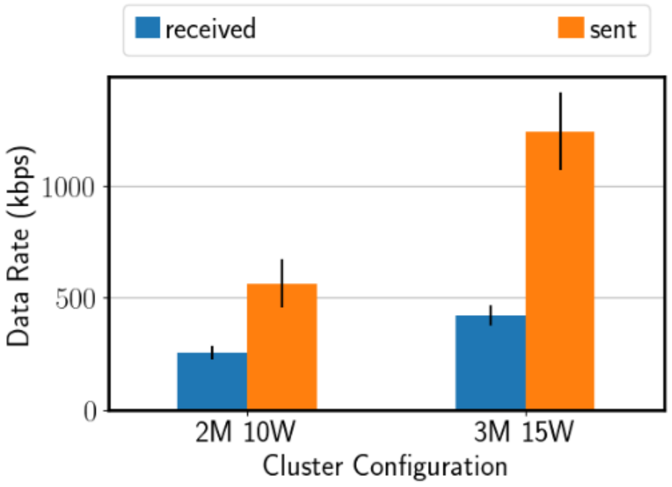
https://kubernetes.io/docs/concepts/overview/components/

What’s the Problem with Orchestration?

Kubernetes Performance Issues

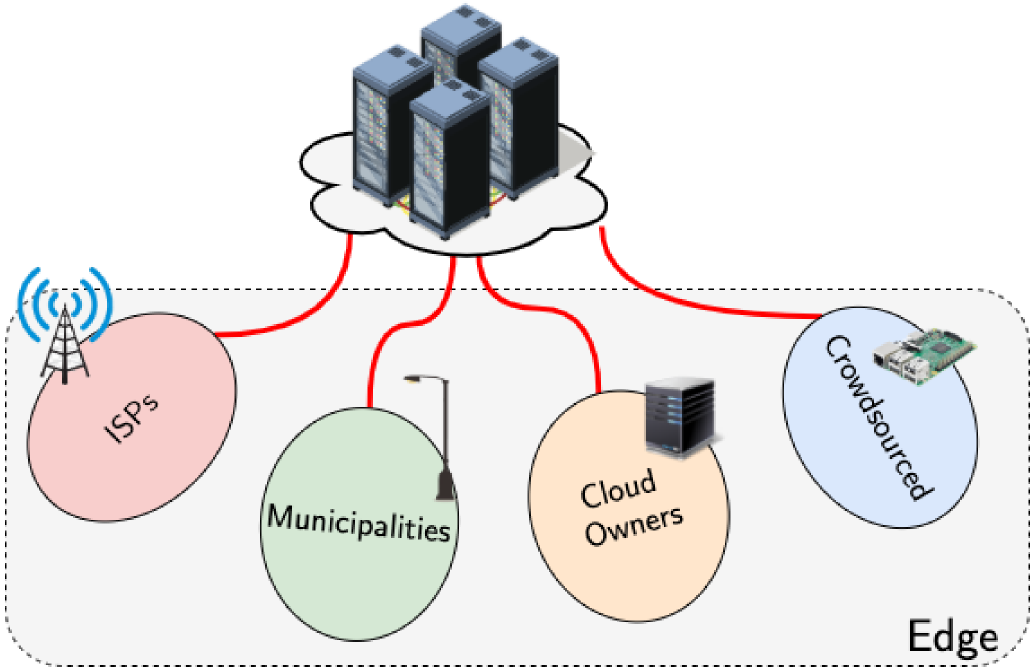






Does not perform well in networks with long and invariable delays!

Oakestra: Tenets



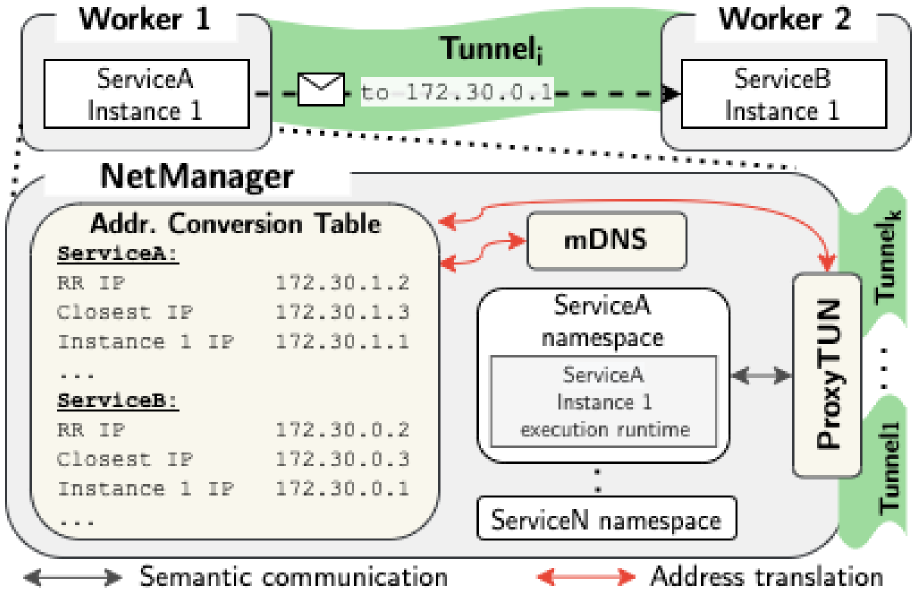
1. Support for Heterogeneity
   - in capabilities, e.g. CPU, GPU, TPU, …
   - in architectures, e.g. ARM, x86, …
   - in access, e.g. WiFi, ethernet, cellular, …
   - in virtualization support, e.g. containers, microVM, unikernels, …

2. Scalable and flexible execution

3. Support **federated** infrastructures involving multiple operators at different bands of the edge spectrum

Oakestra: Data Plane Networking

Designed to support the heterogeneous network environment at the edge

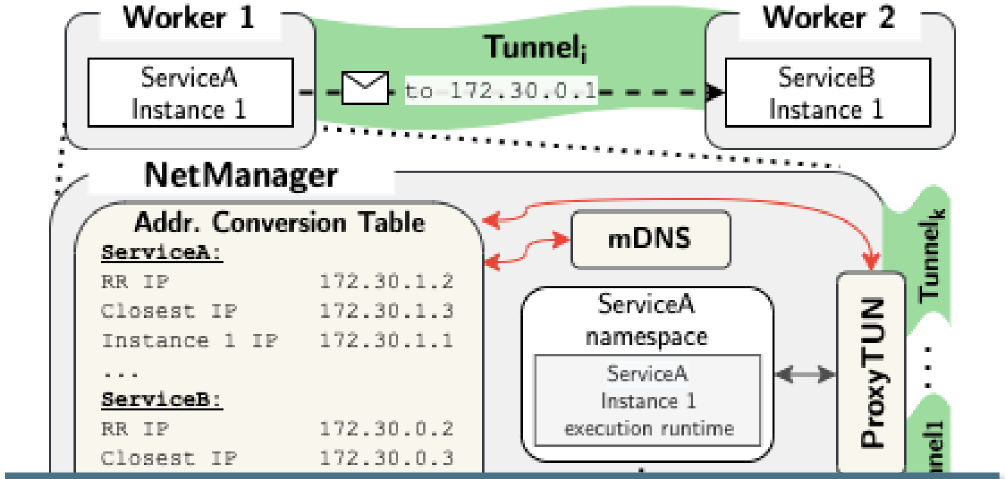


Addr. Conversion Table

```
ServiceA:
RR IP            172.30.1.2
Closest IP       172.30.1.3
Instance 1 IP    172.30.1.1
...
ServiceB:
RR IP            172.30.0.2
Closest IP       172.30.0.3
Instance 1 IP    172.30.0.1
...
```

Oakestra: Data Plane Networking

Designed to support the heterogeneous network environment at the edge



Addr. Conversion Table

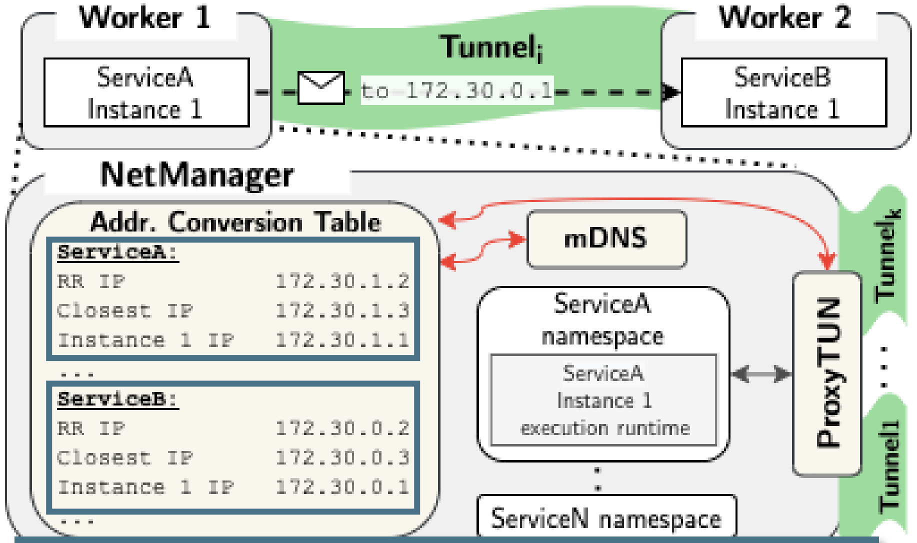
```
ServiceA:
RR IP            172.30.1.2
Closest IP       172.30.1.3
Instance 1 IP    172.30.1.1
...
ServiceB:
RR IP            172.30.0.2
Closest IP       172.30.0.3
```

```
http://appname.appns.servicename.
servicens.instancenumber.
routing_policy.local:port/api
```

1. Semantic Service Addressing keeps track of multiple service instances deployed on different resource

Oakestra: Data Plane Networking

Designed to support the heterogeneous network environment at the edge



Addr. Conversion Table

```
ServiceA:
RR IP               172.30.1.2
Closest IP          172.30.1.3
Instance 1 IP       172.30.1.1
...
ServiceB:
RR IP               172.30.0.2
Closest IP          172.30.0.3
Instance 1 IP       172.30.0.1
...
```

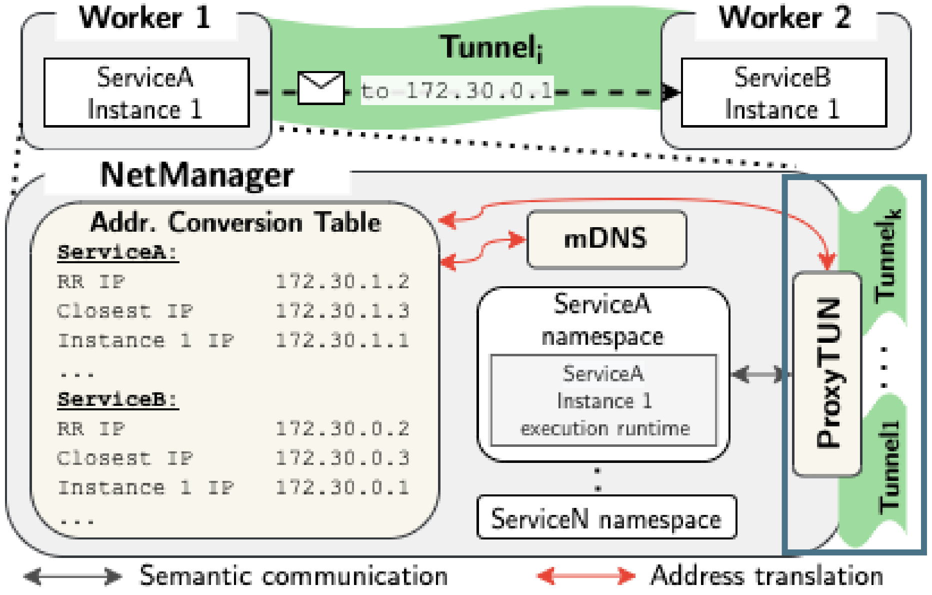
```
1. Round Robin
2. Closest instance deployed
3. Specific instances
```

1. Semantic Service Addressing keeps track of multiple service instances deployed on different resource

2. Dynamic routing policies support load balancing at the edge

Oakestra: Data Plane Networking

Designed to support the heterogeneous network environment at the edge



Addr. Conversion Table

```
ServiceA:
RR IP            172.30.1.2
Closest IP       172.30.1.3
Instance 1 IP    172.30.1.1
...
ServiceB:
RR IP            172.30.0.2
Closest IP       172.30.0.3
Instance 1 IP    172.30.0.1
...
```

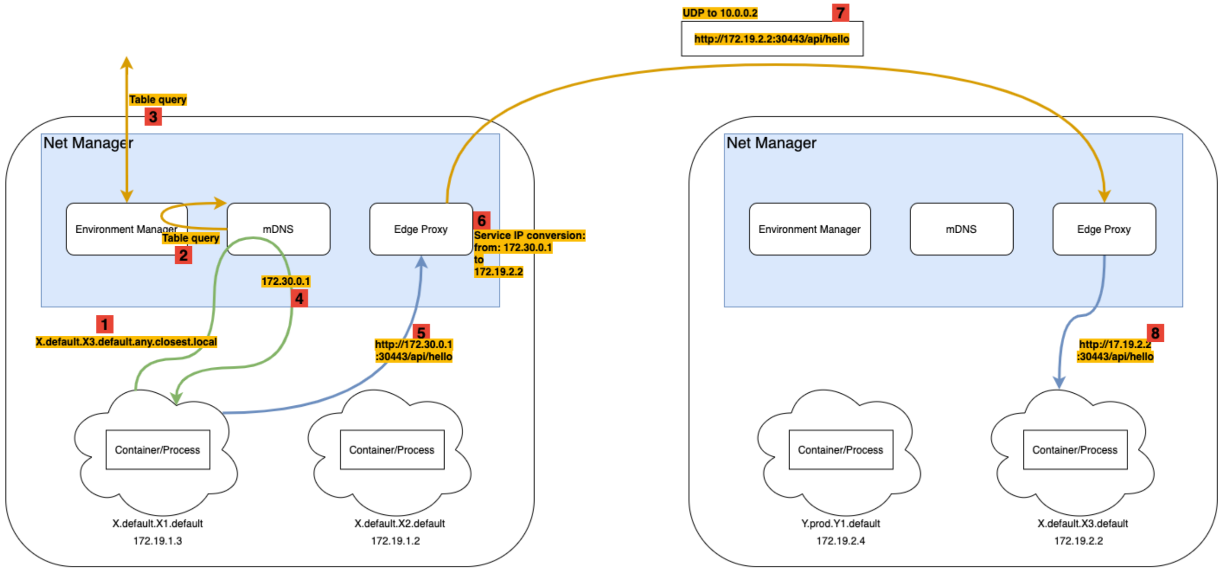
1. Semantic Service Addressing keeps track of multiple service instances deployed on different resource

2. Dynamic routing policies support load balancing at the edge

3. Worker-supported L4 tunneling allows services to interact across network domains (and cluster boundaries)

EdgeIO Features

Flexible Networking over Operational Boundaries



http://appname.appns.servicename.servicens.
instancenumber.policy.local:port/api

- Run distributed applications on nodes behind different organization networks
- Compute nodes need not be in same (or public) network to participate
- Flexible networking that supports application migration, replication, termination and failures
- Load-balancing between multiple application instances

Oakestra: Implementation

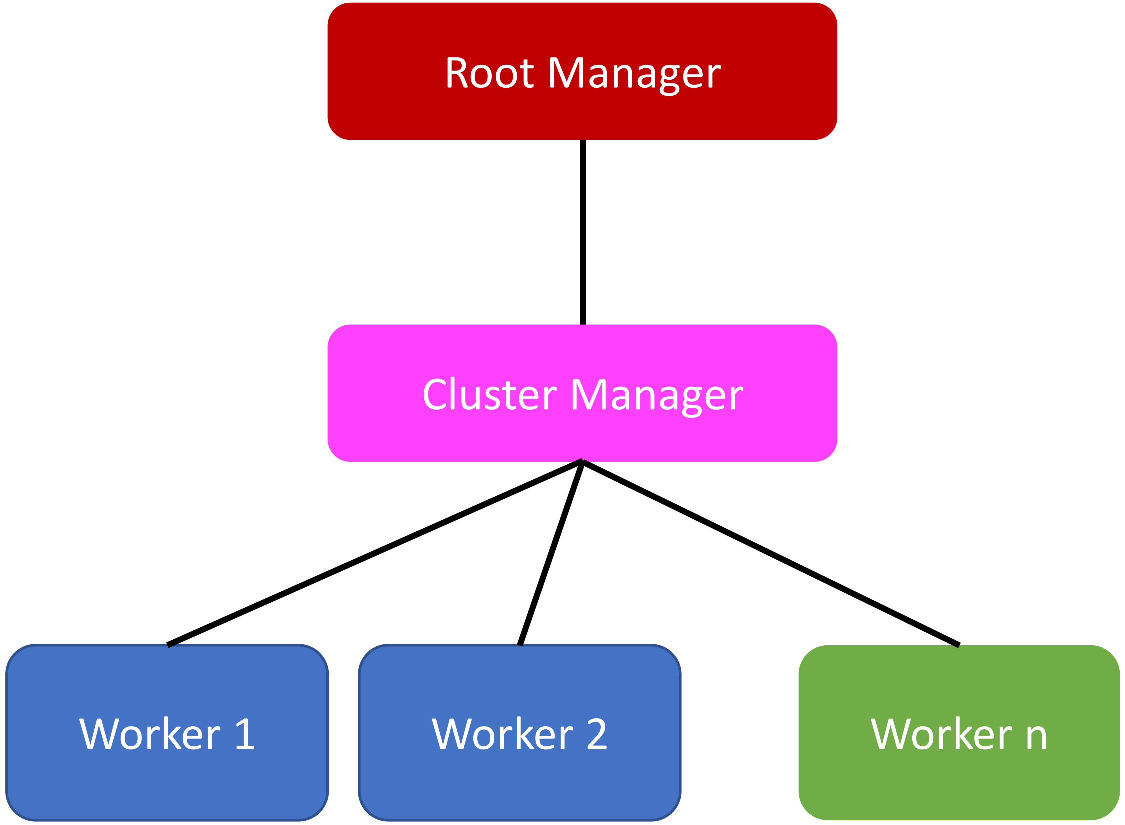
- Implemented in approximately 11000 LOC
- Main implementation in Python and networking component in GoLang
- Easily deployable as Linux containers for both x86 and ARM achitectures
- Currently supports Linux and Docker container-ized services (support for Unikraft-based unikernels in progress, more virtualization support to be added in future)

Preliminary Evaluation

Emulation of diversity and connectivity of edge infrastructures was most important to us.

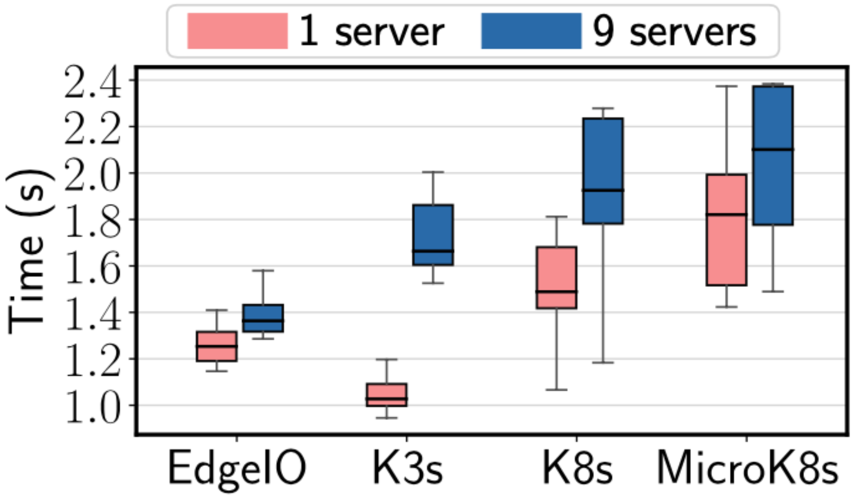
HPI Infrastructure Setup:

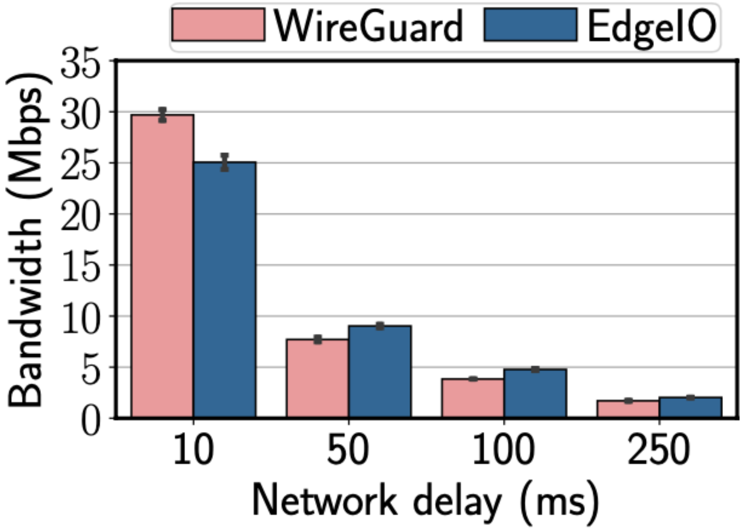
1. 17 VMs of size S
2. 17 VMs of size M
3. 3 VMs of size L
4. 4 VMs of size XL



Oakestra in Action

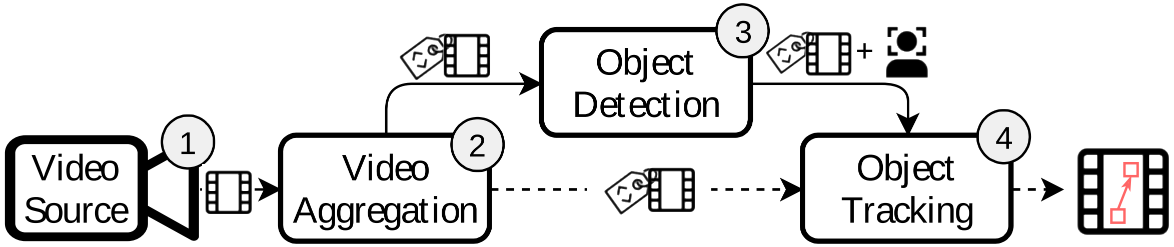
Data-Plane Communication



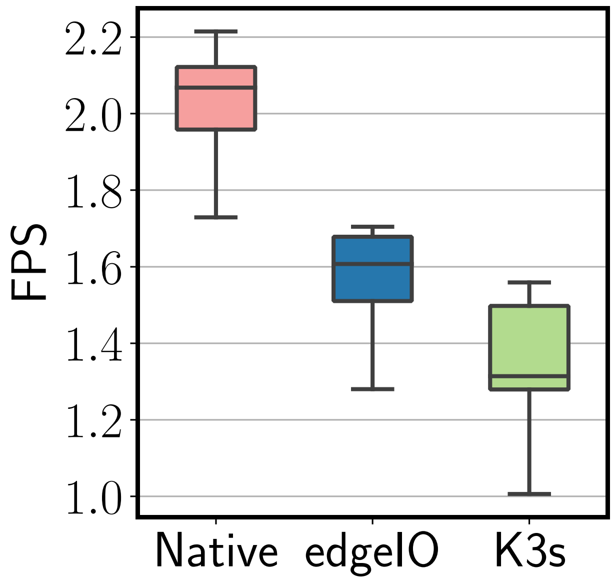


- 10 – 50% improvement over the state-of-the-art
- Similar bandwidth usage while tunneling traffic

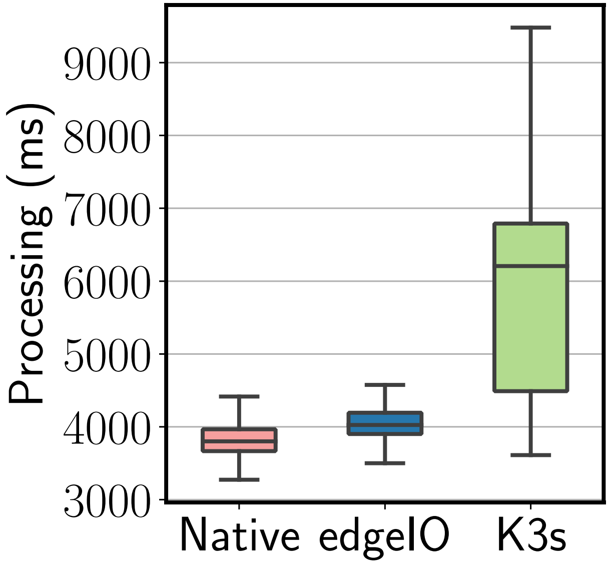
Oakestra: Live Video Analytics



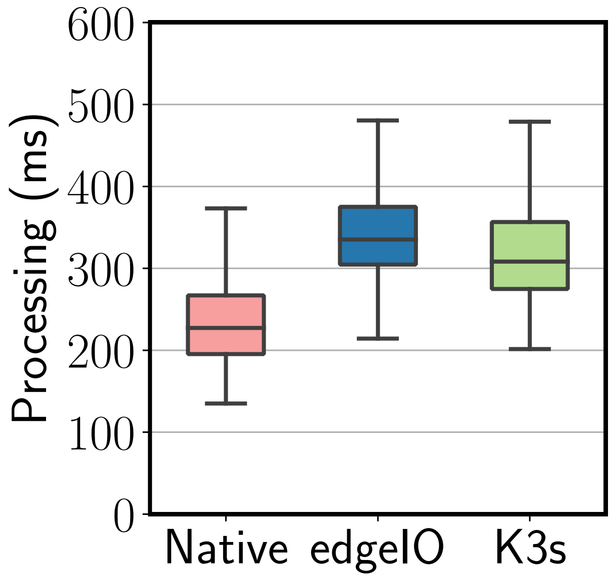
FPS



Detection



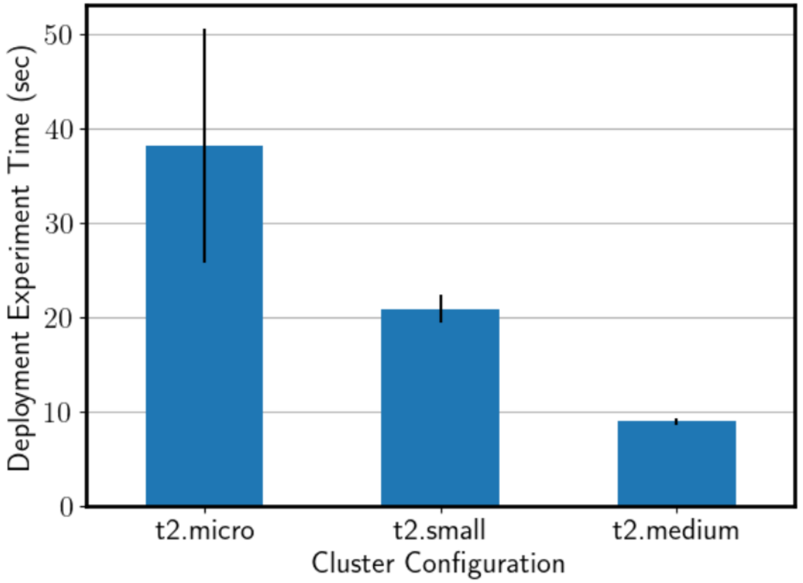
Tracking

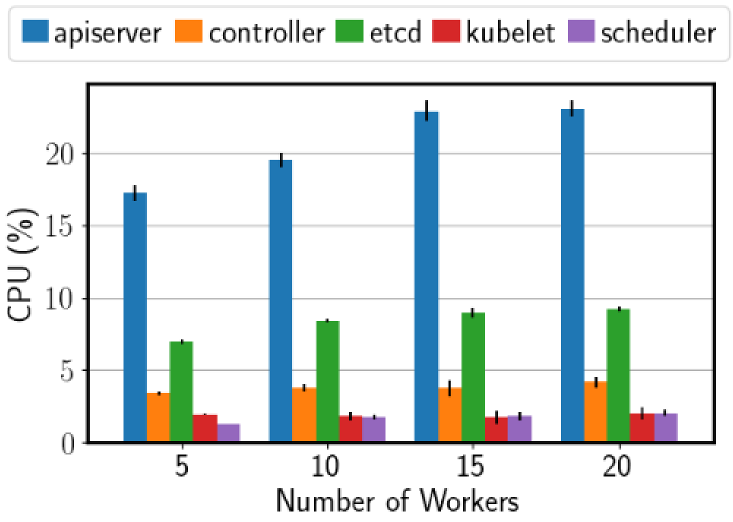


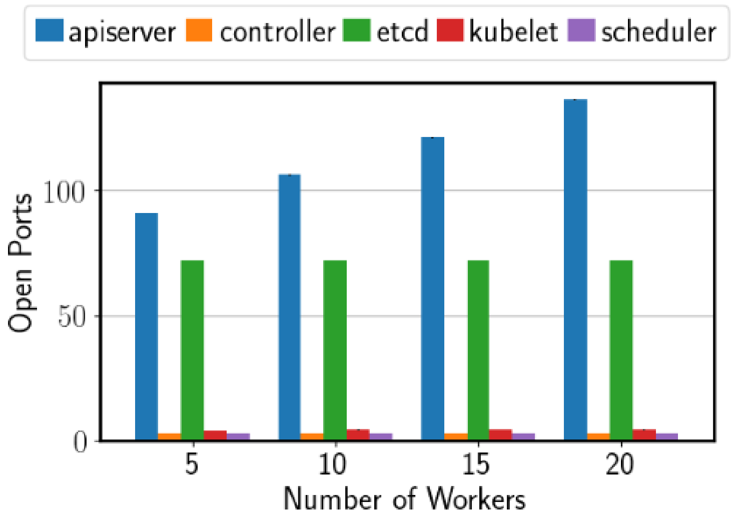
Upto 10% improvement in application performance

Simon Bäurle and Nitinder Mohan. 2022. ComB: a flexible, application-oriented benchmark for edge computing. In 5th International Workshop on Edge Systems, Analytics and Networking (EdgeSys '22).

Kubernetes at the Edge?

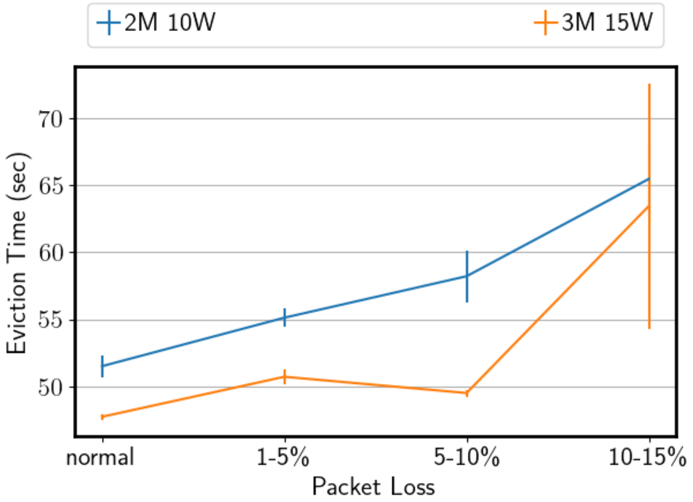






Too much overhead for constrained nodes!

Which worsens with worsening



Sonia Klärmann. *Evaluating the Suitability of Kubernetes for Edge Computing Infrastructure.*, M.Sc. Thesis, TUM

(Inter and Intra Cluster) Communication

Where do we go from here?

- Modular service scheduler for EdgeIO