FogIoT Orchestrator: an Orchestration System for IoT Applications in Fog Environment

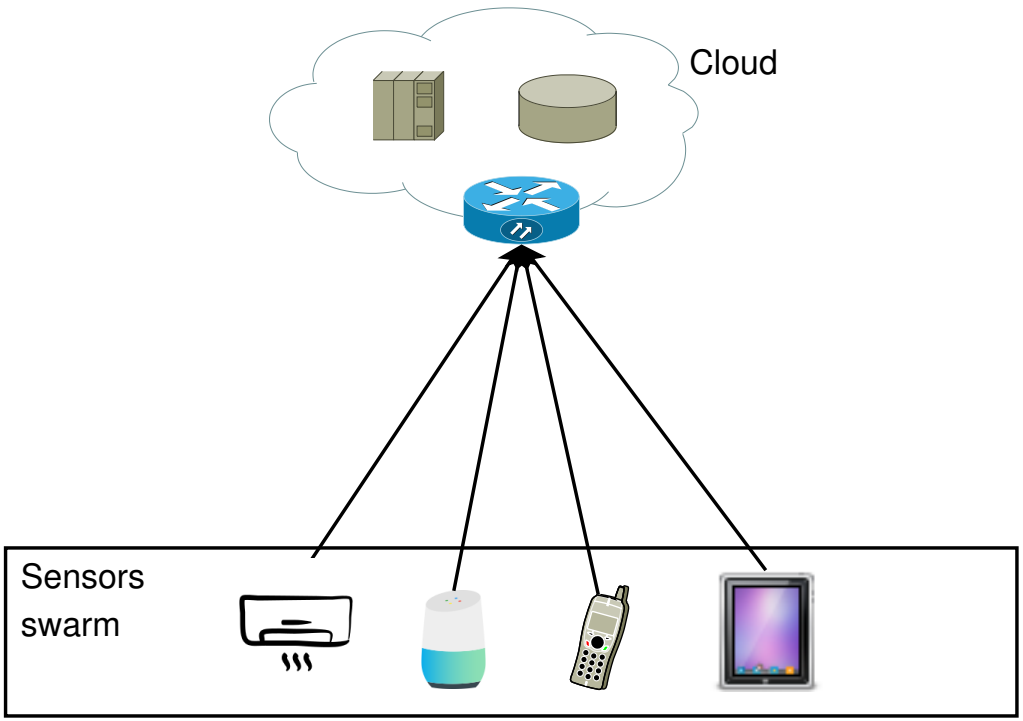
Bruno Donassolo - Orange Labs

Ilhem Fajjari - Orange Labs
Arnaud Legrand - INRIA - LIG
Panayotis Mertikopoulos - INRIA - LIG

April 5, 2018

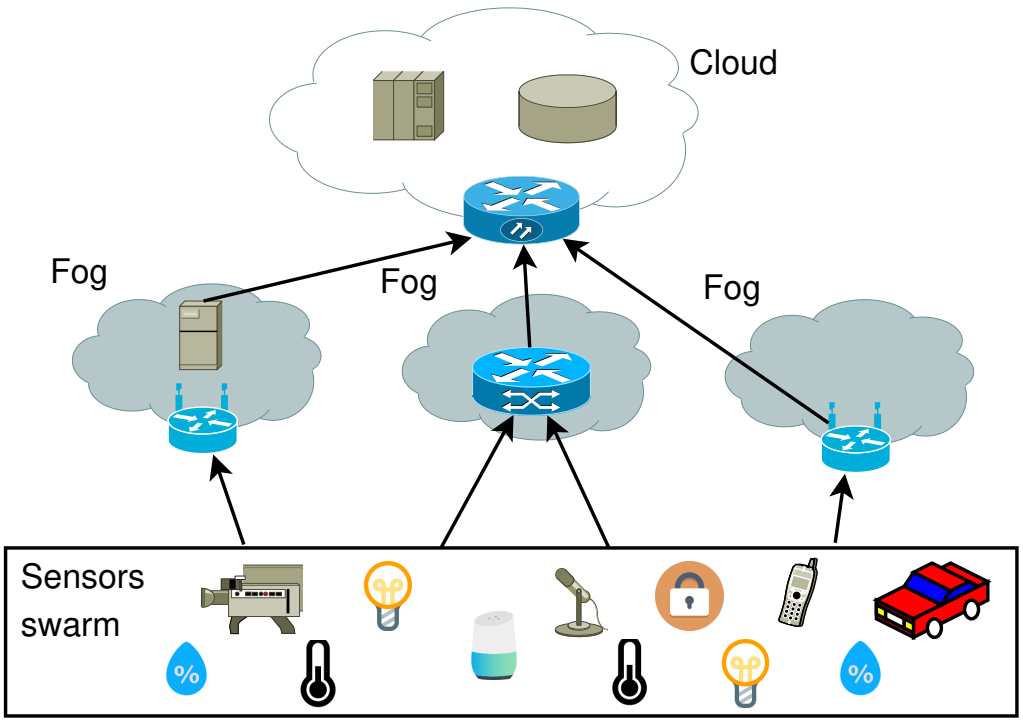
Outline

IoT: Cloud



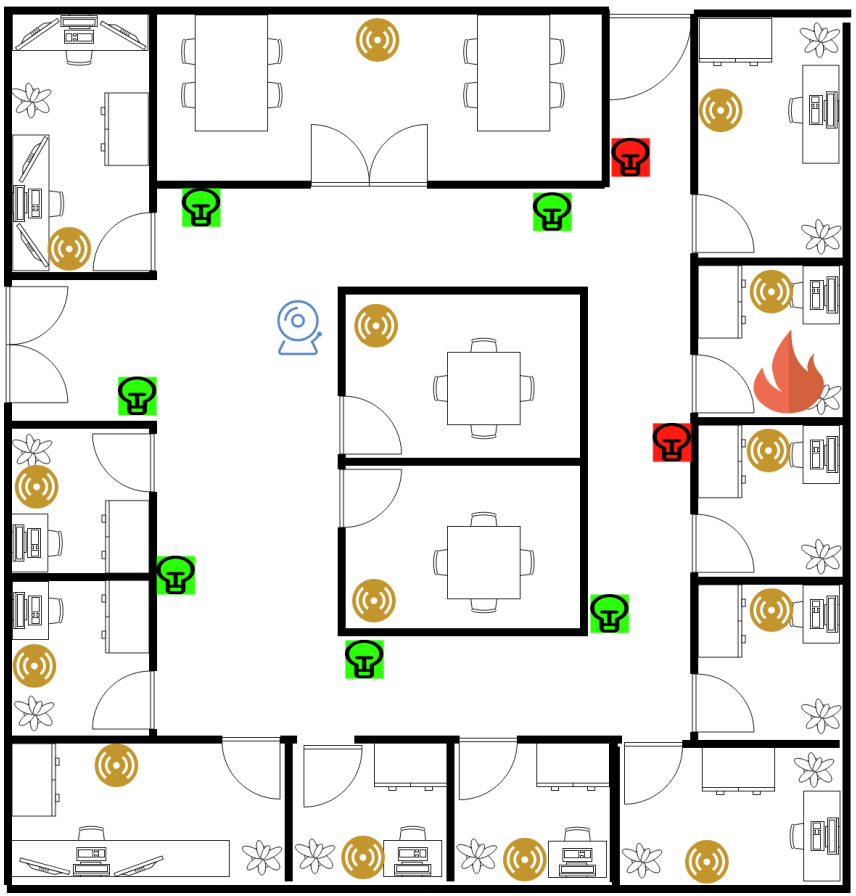
- Apps rely on the cloud for processing.
  - scalable
  - cost-effective
- Problem: latency critical applications.
- E.g.: augmented reality
  - max delay in the order of miliseconds.

IoT: Fog



- Fog Computing
- Complex, heterogeneous, distributed, mobile and dynamic environment.
- Applications that run over it are not simpler...

Application - Fire detection/combat



2-phases:

- detection
- evacuation plan

Requirements:

- Low latency: detection
- Processing: evacuation path

Main challenges in a fog environment

What do we need to create a fog environment?

1. Infrastructure
   - IoT sensors/actuators
   - Cloud
   - Devices in the range edge-cloud

Main challenges in a fog environment

What do we need to create a fog environment?

1. Infrastructure
   - IoT sensors/actuators
   - Cloud
   - Devices in the range edge-cloud
2. Model/write the application

Main challenges in a fog environment

What do we need to create a fog environment?

1. Infrastructure
   - IoT sensors/actuators
   - Cloud
   - Devices in the range edge-cloud
2. Model/write the application
3. Hardware abstraction

Main challenges in a fog environment

What do we need to create a fog environment?

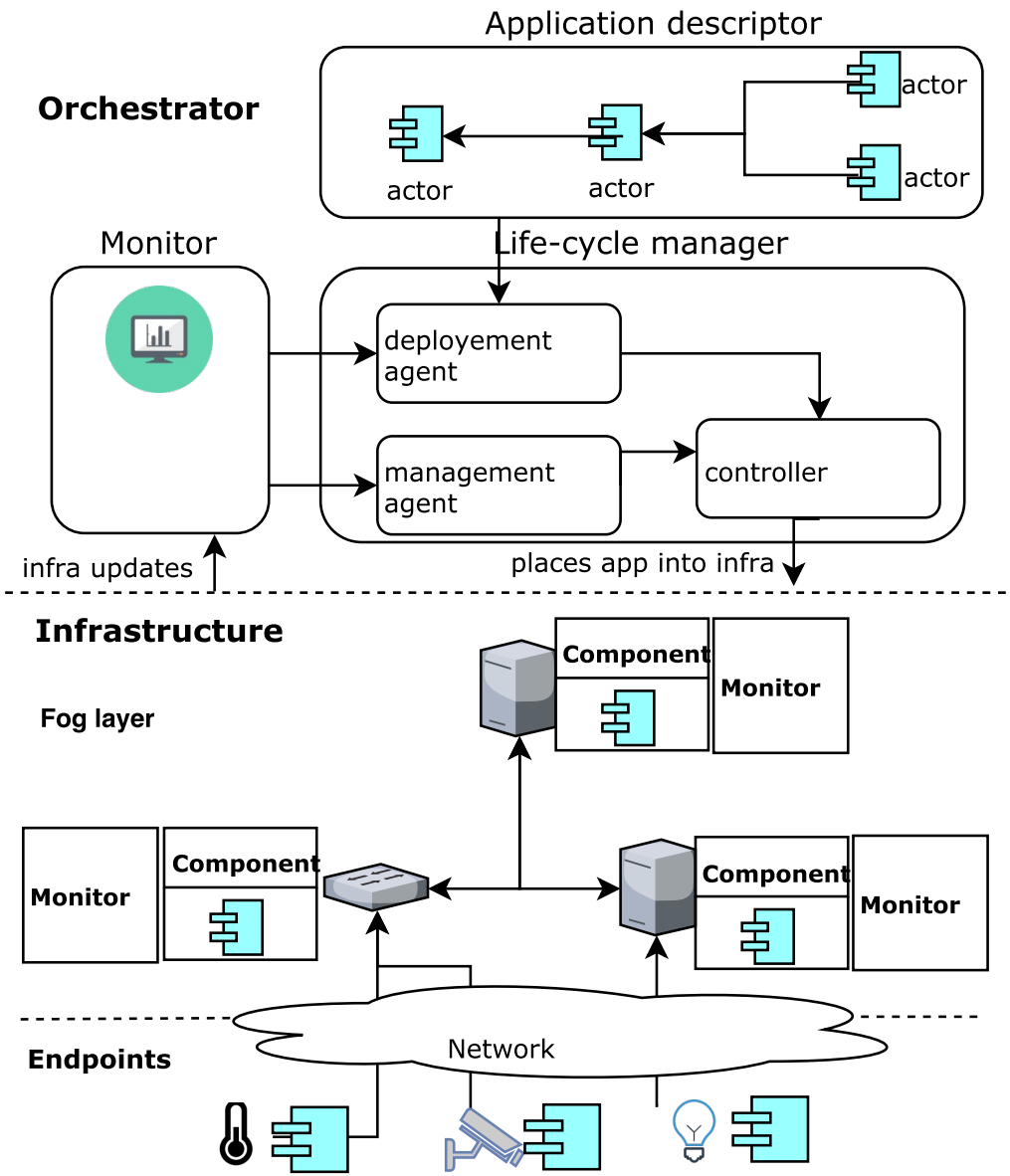
1. Infrastructure
   - IoT sensors/actuators
   - Cloud
   - Devices in the range edge-cloud
2. Model/write the application
3. Hardware abstraction
4. Deploy app. components in the infrastructure

Main challenges in a fog environment

What do we need to create a fog environment?

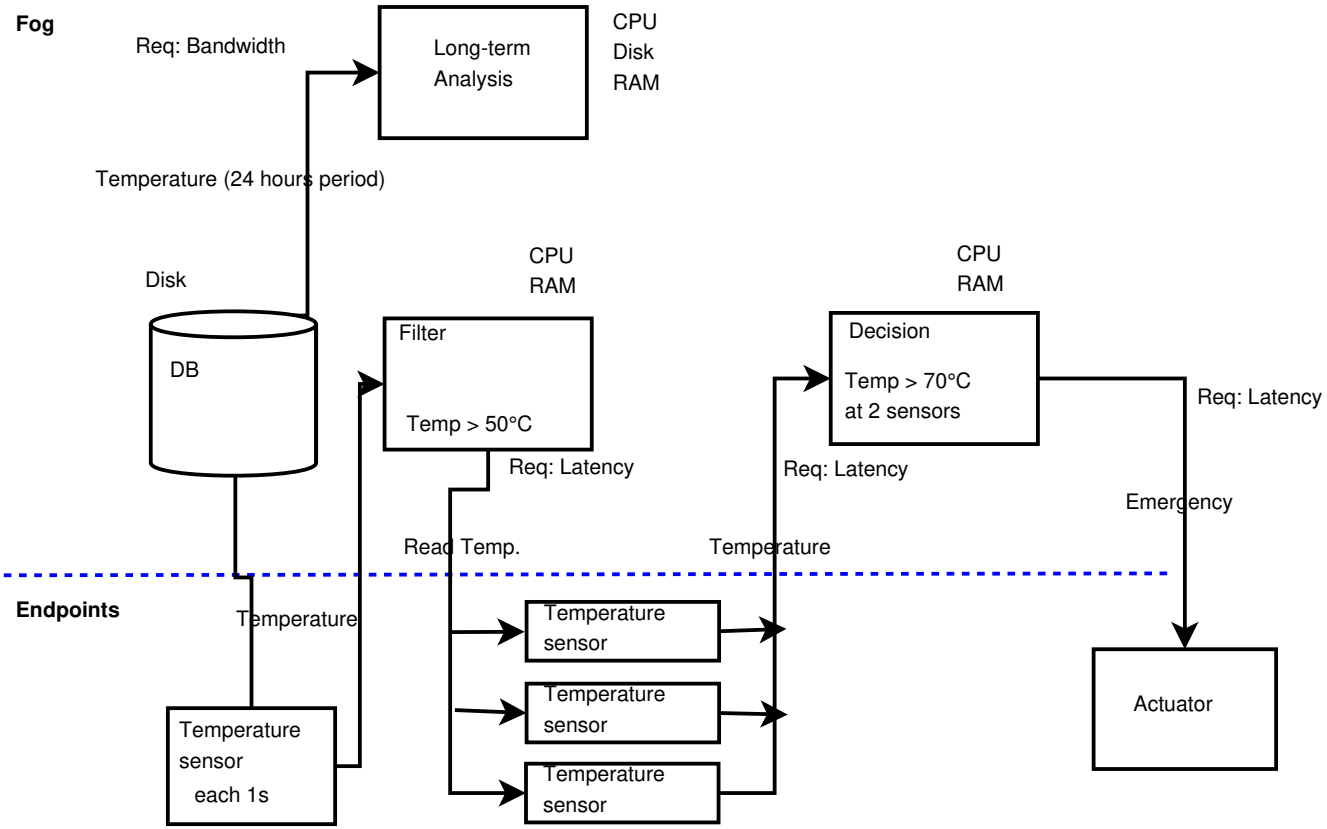
1. Infrastructure
   - IoT sensors/actuators
   - Cloud
   - Devices in the range edge-cloud
2. Model/write the application
3. Hardware abstraction
4. Deploy app. components in the infrastructure
5. Monitoring the infrastructure
   - CPU/RAM
   - Network latency/bandwidth

Architecture



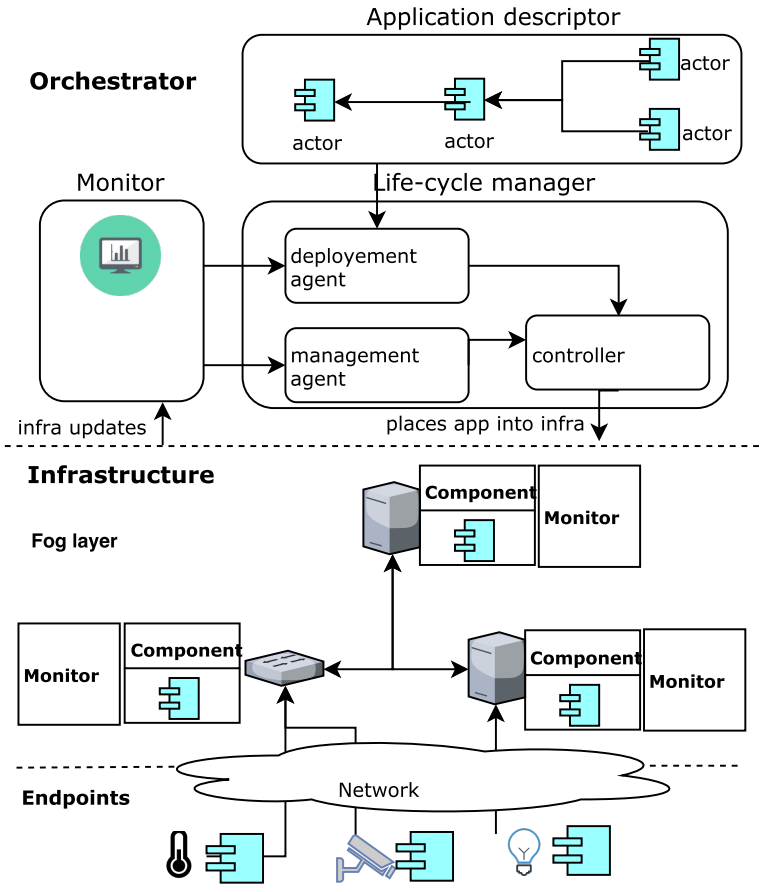
- Application descriptor:
  - actor-based, data-flow programming model
  - agnostic to infrastructure
- Life-cycle manager:
  - actors placement
  - reconfiguration
- Hw abstraction
  - containers

Use case

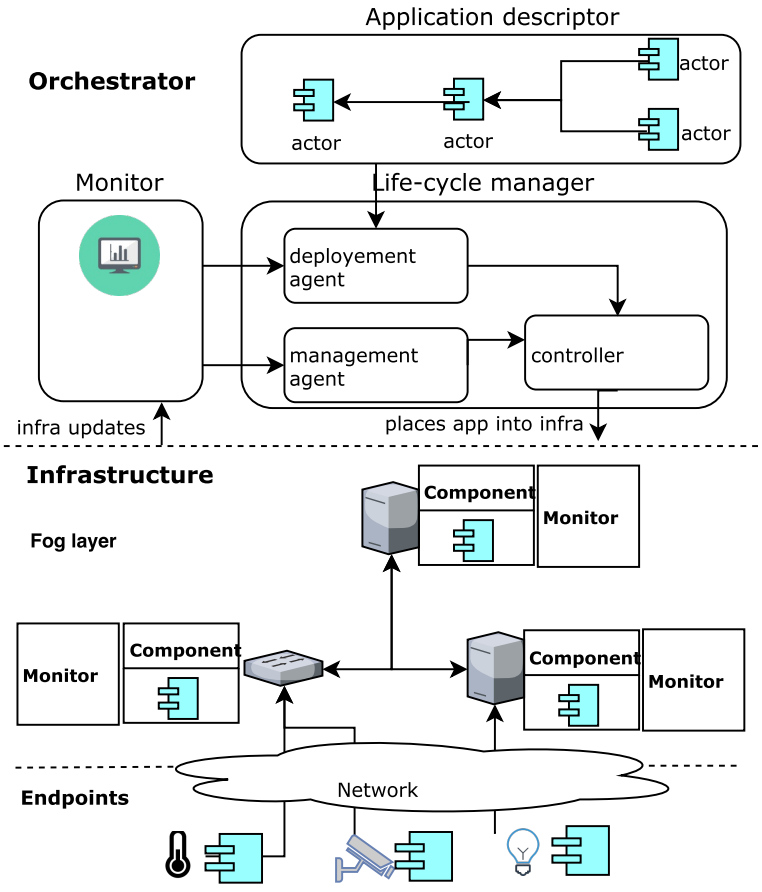


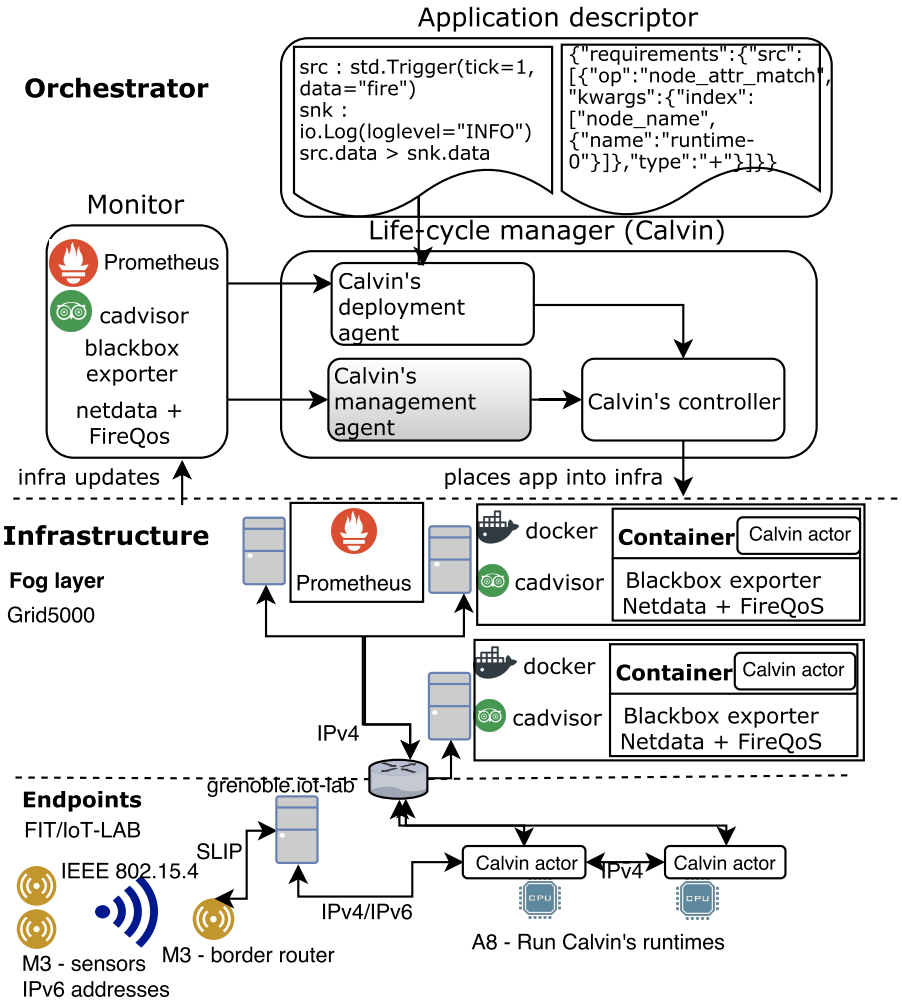
- Simplified fire detection app
- But, it contains the typical requirements associated to a fog application

High-level

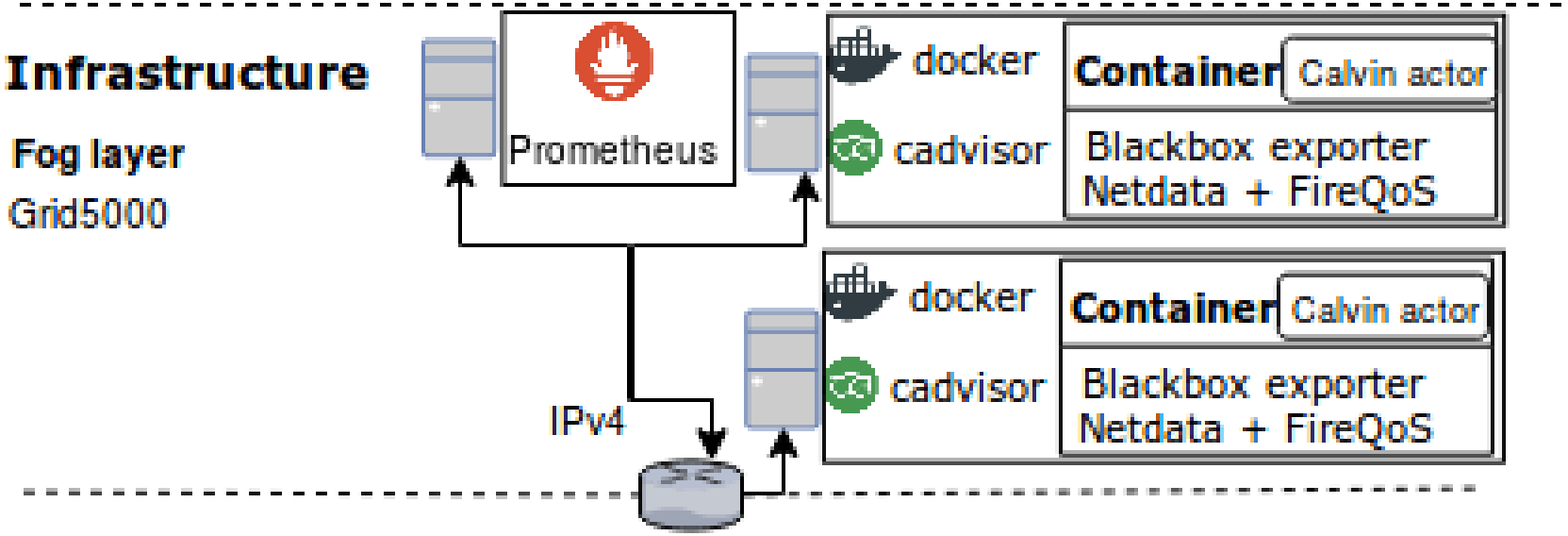

Application descriptor
Orchestrator
actor
actor
actor
actor
Monitor
Life-cycle manager
deployement agent
management agent
controller
infra updates
places app into infra
Infrastructure
Fog layer
Component
Monitor
Monitor
Component
Component
Monitor
Endpoints
Network

High-level


Application descriptor
Orchestrator
actor
actor
actor
actor
Monitor
Life-cycle manager
deployement agent
management agent
controller
infra updates
places app into infra
Infrastructure
Fog layer
Component
Monitor
Monitor
Component
Component
Monitor
Network
Endpoints

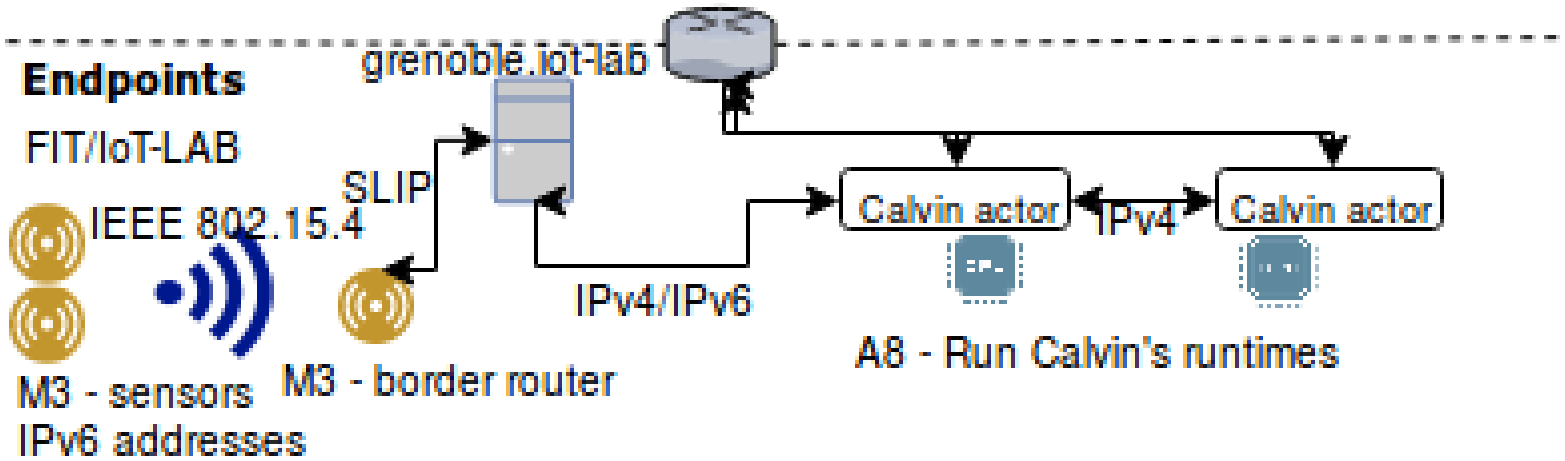

Application descriptor
Orchestrator
src : std.Trigger(tick=1, data="fire")
snk : io.Log(loglevel="INFO")
src.data > snk.data
{"requirements":{"src":[{"op":"node_attr_match","kwargs":{"index":["node_name",{"name":"runtime-0"}]},"type":"+"}]}}
Monitor
Prometheus
cadvisor
blackbox exporter
netdata + FireQos
Life-cycle manager (Calvin)
Calvin's deployment agent
Calvin's management agent
Calvin's controller
infra updates
places app into infra
Infrastructure
Fog layer
Grid5000
Prometheus
docker
cadvisor
Container
Calvin actor
Blackbox exporter
Netdata + FireQoS
docker
cadvisor
Container
Calvin actor
Blackbox exporter
Netdata + FireQoS
IPv4
grenoble.iot-lab
Endpoints
FIT/IoT-LAB
IEEE 802.15.4
SLIP
IPv4/IPv6
Calvin actor
IPv4
Calvin actor
M3 - sensors
IPv6 addresses
M3 - border router
A8 - Run Calvin's runtimes

Grid5000



- Provisioning: Ansible
- Using different sites: global VLAN
  - Forward multicast packets needed by Calvin

FIT/IoT-LAB



- Provisioning:
  - python/bash scripts
  - FIT tools: open-a8-cli, opkg
- A8 nodes: calvin
- M3 nodes (temperature sensor): CoAP protocol, IPv6/SLIP
- M3 nodes and OSs: RIOT vs Contiki

Overview: Grid5000 and FIT/IoT-LAB

- Main problem using both platforms: Connectivity
  - private, independent networks

Overview: Grid5000 and FIT/IoT-LAB

- Main problem using both platforms: Connectivity
  - private, independent networks
- Solution:
  - VPNs
  - Install openvpn in A8 nodes to put them in the Grid5000 network
  - https://www.grid5000.fr/mediawiki/index.php/VPN

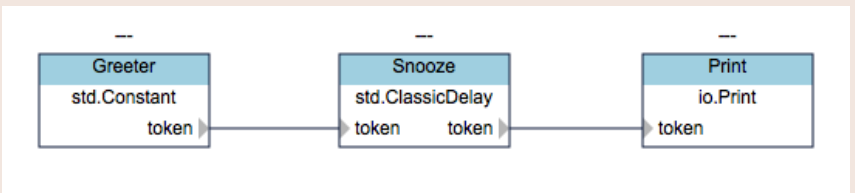
Overview: Grid5000 and FIT/IoT-LAB

- Main problem using both platforms: Connectivity
  - private, independent networks
- Solution:
  - VPNs
  - Install openvpn in A8 nodes to put them in the Grid5000 network
  - https://www.grid5000.fr/mediawiki/index.php/VPN
- Problem:
  - Artificial link
  - Realistic?
  - Latency: $25ms$
  - Bandwidth: $20Mb$

Calvin

- Open source project lead by Ericsson
- https://github.com/EricssonResearch/calvin-base

- Concept:
  - IoT development must be simple
  - Not worry about communication protocols and hardware specifics



- Applications:
  - Actor model: private internal state
  - Flow based computing

Calvin

Application description:

- GUI
- Text: own syntax

Functional

```
src : std.Trigger(tick=1, data="fire")
snk : io.Log(loglevel="INFO")
src.data > snk.data
```

Deployment

```
{"requirements":{"src":[{
"op":"node_attr_match",
"kwargs":{"index":["node_name",
{"name":"runtime-0"}]},"type":"+"}]}}
```

Calvin

Architecture:

- Calvin's runtimes: abstraction to actors
- Requirement: IP connectivity
  - Multicast packets to node discovery

Deployment:

- Automatic select runtime to run actors

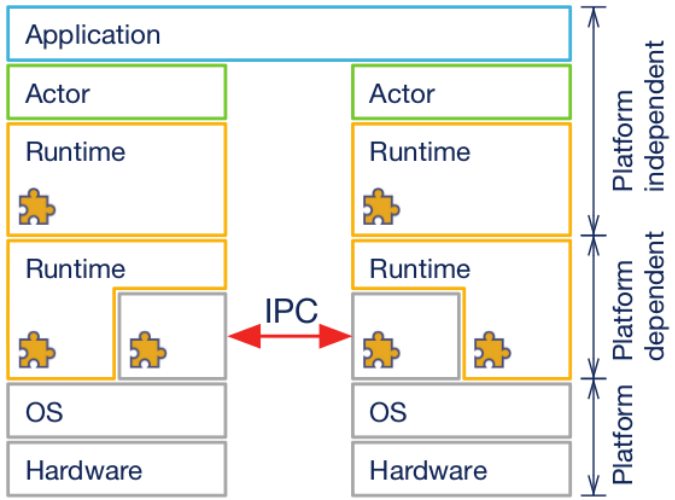


Image from: Calvin – Merging Cloud and IoT

https://doi.org/10.1016/j.procs.2015.05.059

Calvin

Architecture:

- Calvin's runtimes: abstraction to actors
- Requirement: IP connectivity
  - Multicast packets to node discovery

Deployment:

- Automatic select runtime to run actors

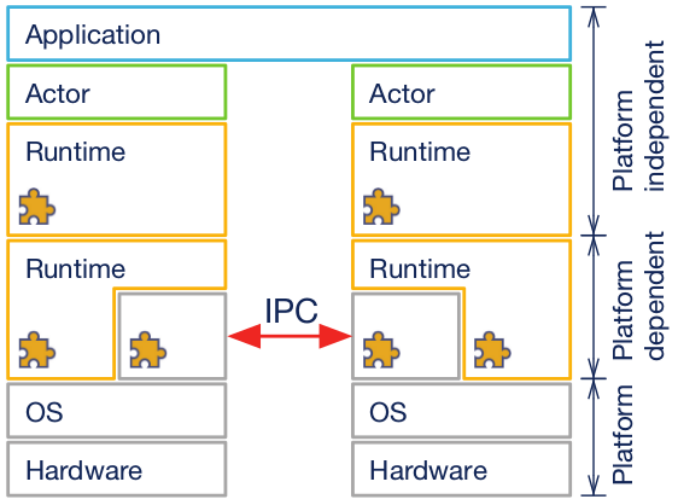


Image from: Calvin – Merging Cloud and IoT

https://doi.org/10.1016/j.procs.2015.05.059

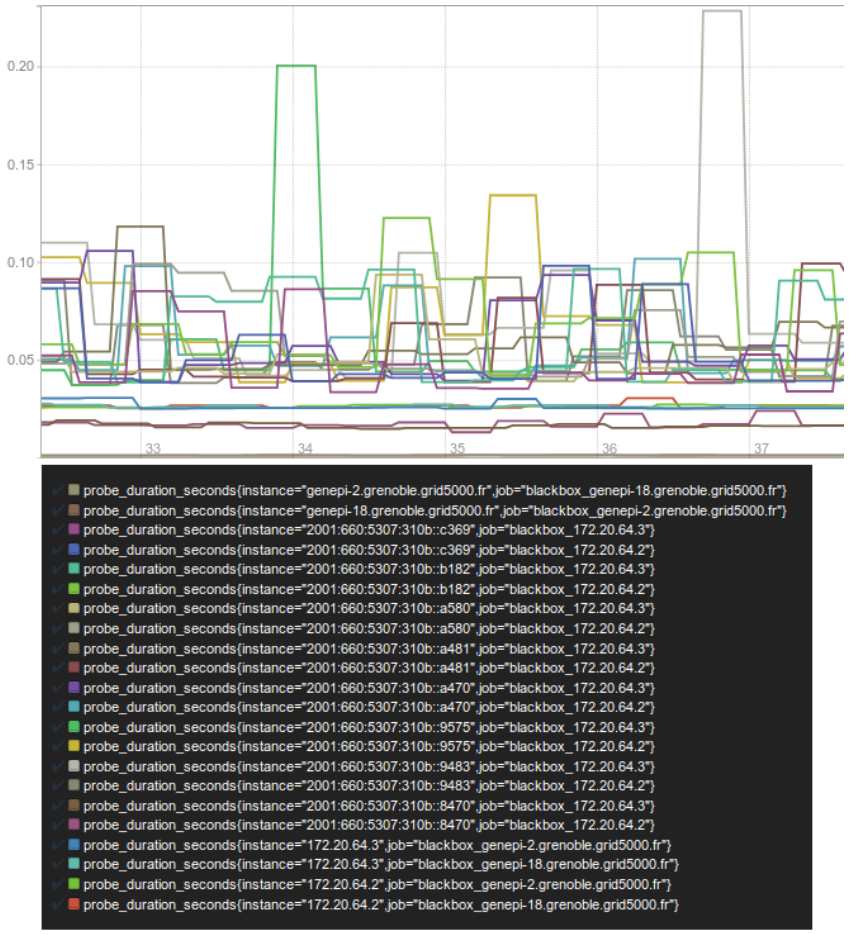
What is missing?

App is running... What about the monitoring?

Monitoring - Prometheus

Prometheus

- https://prometheus.io/
- Time-series database
- Allow post-mortem analysis of tests
- Easy integration with other tools

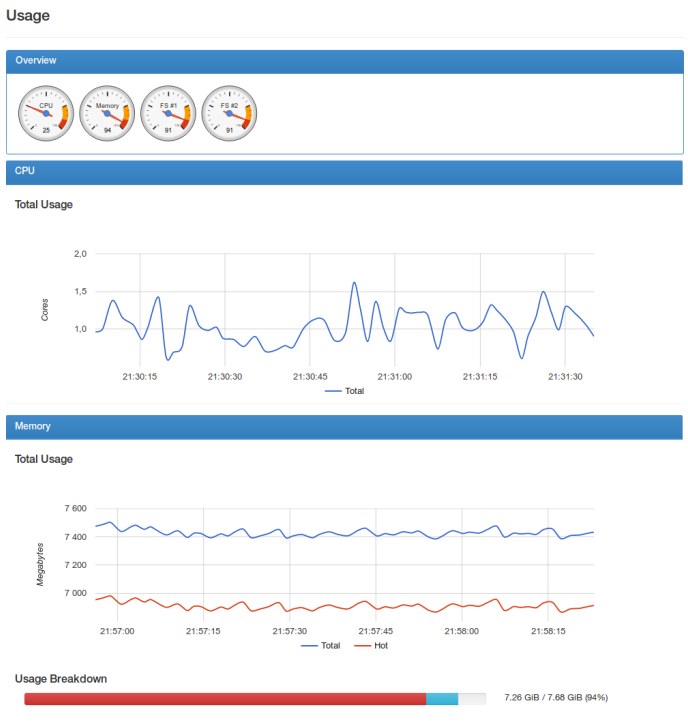



```
scrape_configs:
- job_name: 'prometheus'
  static_configs:
    - targets:['localhost:9090']
```

Monitoring - Cadvisor

Cadvisor

- https://github.com/google/cadvisor
- Monitors performance of docker containers
  - CPU
  - RAM
- Real-time
- Easy to deploy:
  - docker run google/cadvisor:latest
- Exporting/visualizing metrics:
  - Web UI
  - REST
  - Prometheus:
    - localhost:8080/metrics



Monitoring - Blackbox exporter

Blackbox exporter

- https://github.com/prometheus/blackbox_exporter
- Service availability:
  - HTTP, HTTPS, DNS, TCP and ICMP.
- Our use, monitor network latency
- Access:
  - http://localhost:9115/probe?target=google.com&module=icmp

Monitoring - Blackbox exporter

Blackbox exporter

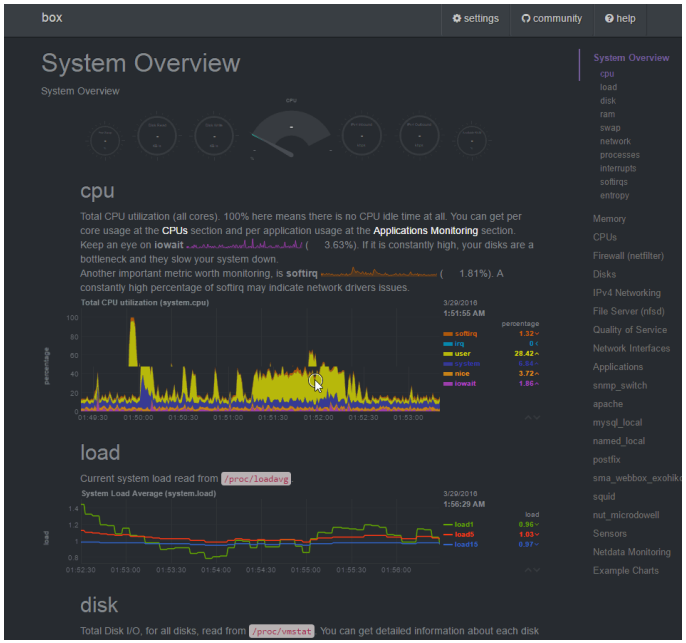
- https://github.com/prometheus/blackbox_exporter
- Service availability:
  - HTTP, HTTPS, DNS, TCP and ICMP.
- Our use, monitor network latency
- Access:
  - http://localhost:9115/probe?target=google.com&module=icmp

```
scrape_configs:
  - job_name: 'blackbox'
      module: [icmp]  # ping request
    static_configs:
      - targets:   # List of target IPs
    relabel_configs:
replacement: 127.0.0.1:9115  # The blackbox exporter's
```

Monitoring - Netdata

Netdata - FireQoS - Traffic Control

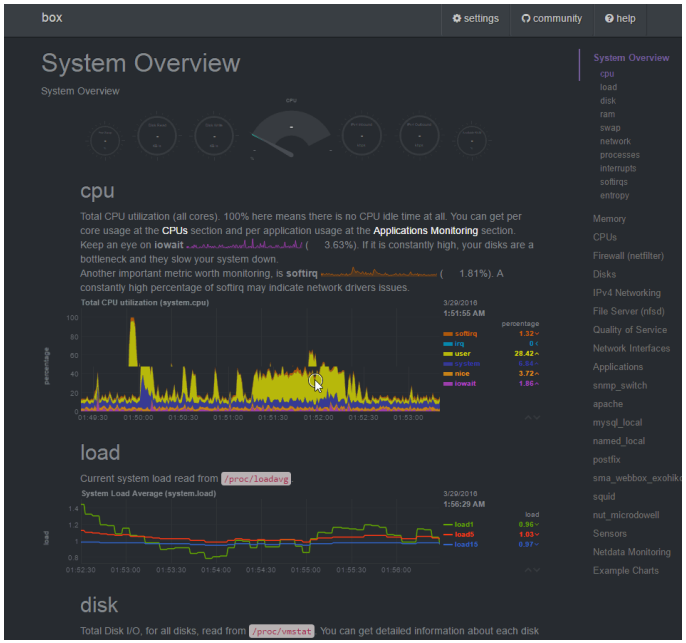
- https://github.com/firehol/netdata
- Last metric to collect:
  - network bandwidth
- Another monitoring tool, but for hosts
  - tons of metrics



Monitoring - Netdata

Netdata - FireQoS - Traffic Control

- https://github.com/firehol/netdata
- Last metric to collect:
  - network bandwidth
- Another monitoring tool, but for hosts
  - tons of metrics



Objective

Measure bandwidth used by calvin between 2 machines

```
interface eth0 world bidirectional ethernet
    class calvin
        match host IP_address
```

Conclusion

- We propose an architecture to orchestrate fog applications
  - Work in progress
  - We show that using both platforms, it is possible to create a fog environment

Conclusion

- We propose an architecture to orchestrate fog applications
  - Work in progress
  - We show that using both platforms, it is possible to create a fog environment
- 2 nice testbeds, complementary capabilities
  - Grid5000: powerful, homogeneous, scalable, focus: cloud, HPC
  - FIT/IoT-LAB: heterogeneous, scalable, focus: IoT

Conclusion

- We propose an architecture to orchestrate fog applications
  - Work in progress
  - We show that using both platforms, it is possible to create a fog environment
- 2 nice testbeds, complementary capabilities
  - Grid5000: powerful, homogeneous, scalable, focus: cloud, HPC
  - FIT/IoT-LAB: heterogeneous, scalable, focus: IoT
- But still, 2 separate platforms
  - 2 queues, usage policies
  - 2 setup process

Conclusion

- We propose an architecture to orchestrate fog applications
  - Work in progress
  - We show that using both platforms, it is possible to create a fog environment
- 2 nice testbeds, complementary capabilities
  - Grid5000: powerful, homogeneous, scalable, focus: cloud, HPC
  - FIT/IoT-LAB: heterogeneous, scalable, focus: IoT
- But still, 2 separate platforms
  - 2 queues, usage policies
  - 2 setup process
- Looking forward for SILECS infrastructure.

That's the End

Thanks