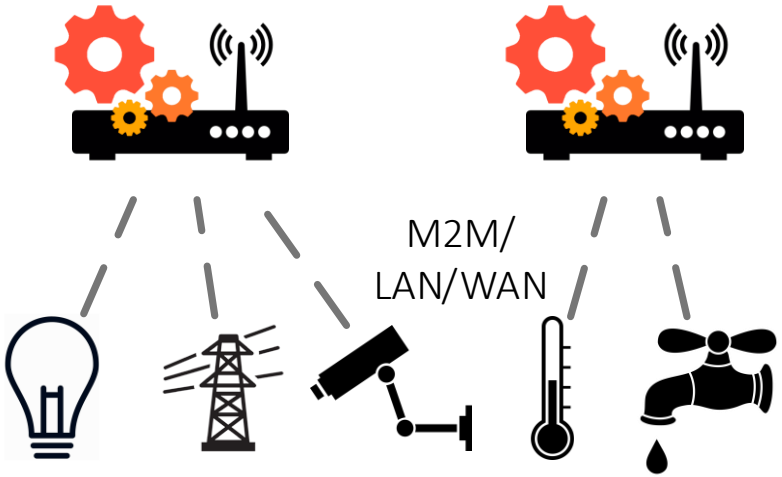
Fog Director Simulator

Don’t destroy your production environment

13 December 2018

Alessandro Pagiaro
https://apagiaro.it

IoT Deployment Models



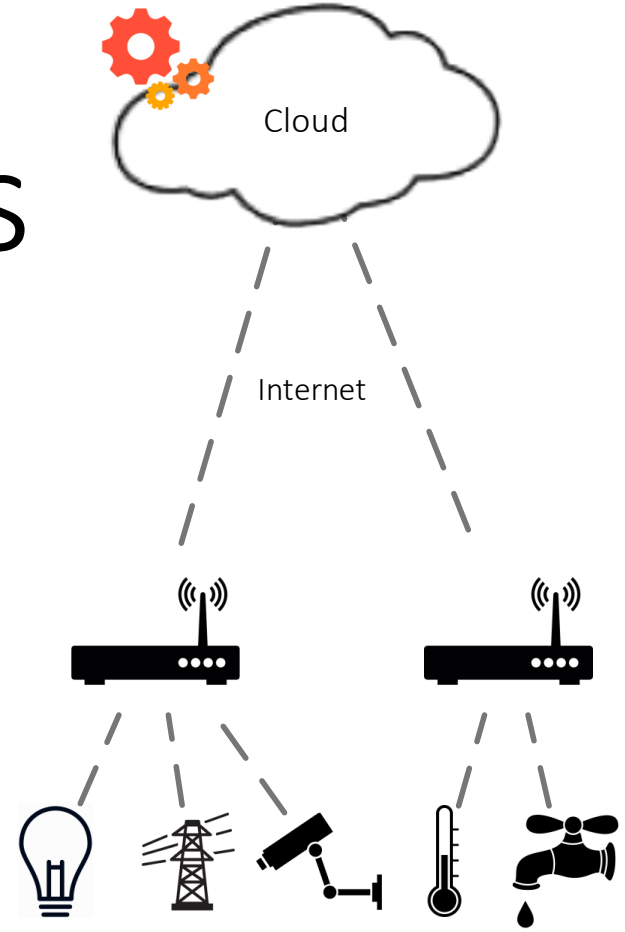
IoT+Edge

- Low latencies, but
- Limited capabilities,
- Difficulties in sharing data

Not sufficient *per se* to support the **IoT momentum** alone.

There is a need for **filtering** and **processing** *before* the Cloud.

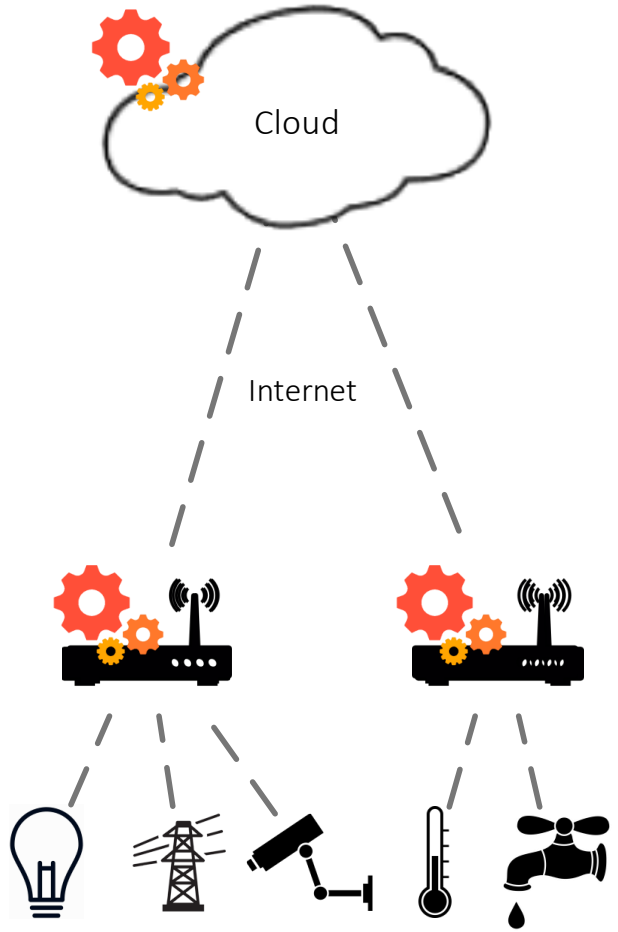
Processing should occur wherever it is ***best-placed*** for any given IoT application



IoT+Cloud

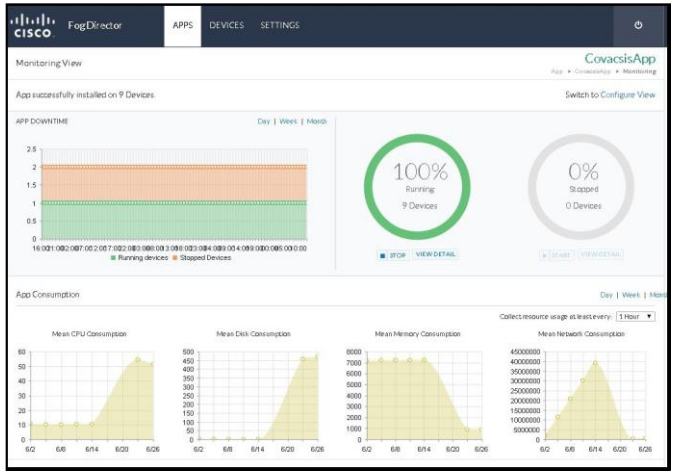
- Huge computing power, but
- Mandatory connectivity,
- High latencies,
- Bandwidth bottleneck.

Fog Computing



Fog Director

A single pane of glass to **manage application lifecycle** on Fog devices.



GUI

app admin

```
# management.py
…
publish(id,A)
n = choose(get_info_N())
d = deploy(id,n)
start(d)
…
on alert do
   stop(d)
   undeploy(d);
   m = choose(get_info_N()))
   deploy(id, m)
…
```

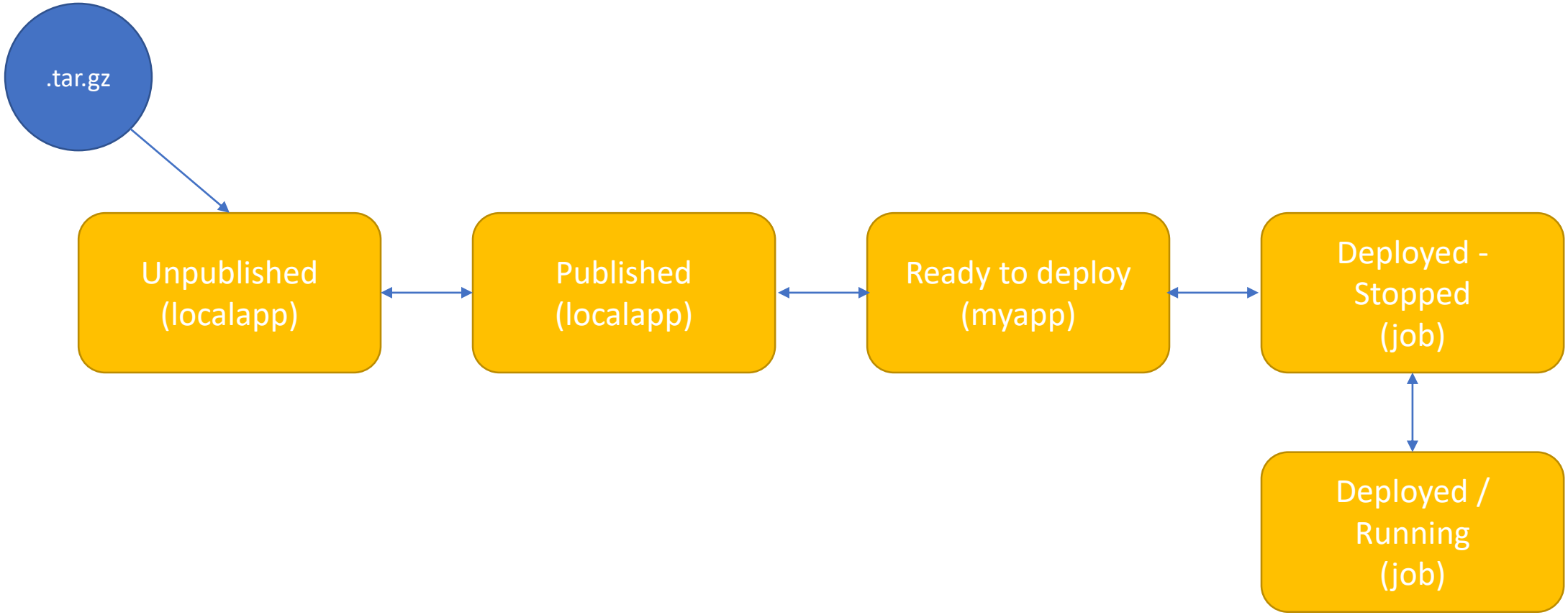
REST API

API

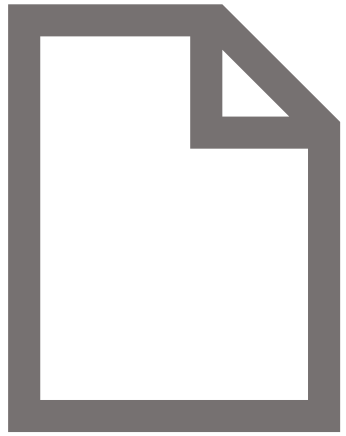
CISCO FogDirector

infrastructure

Fog Director Application Lifecycle



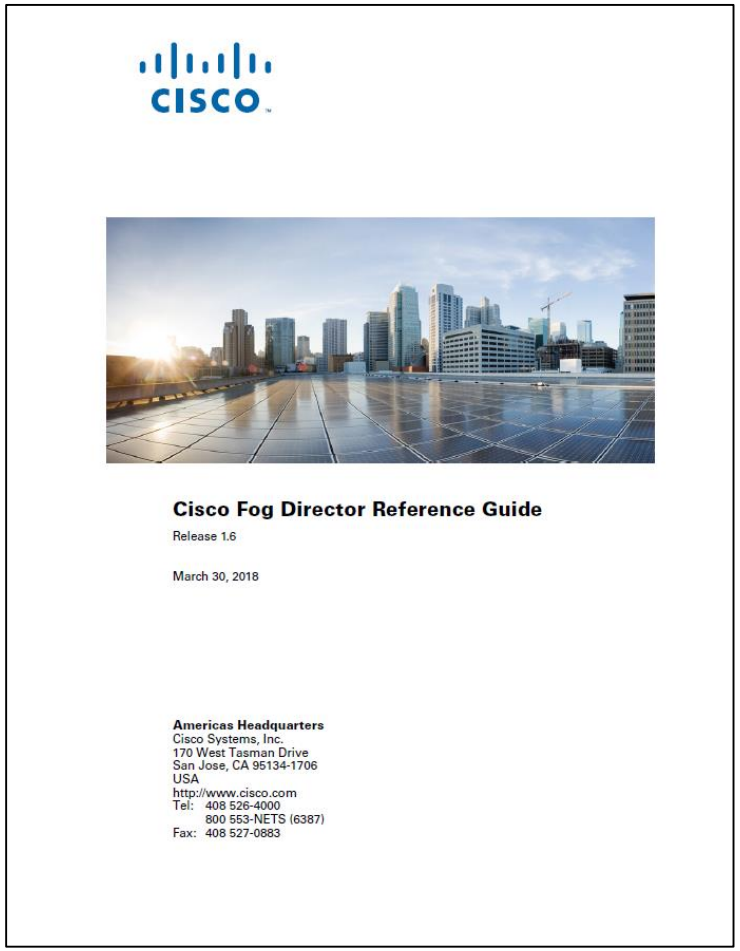
Problems



#1 (quickly) **understand FogDirector** functioning

#2 write **correct** and **effective** management

Problem #1



cisco

Cisco Fog Director Reference Guide

Release 1.6

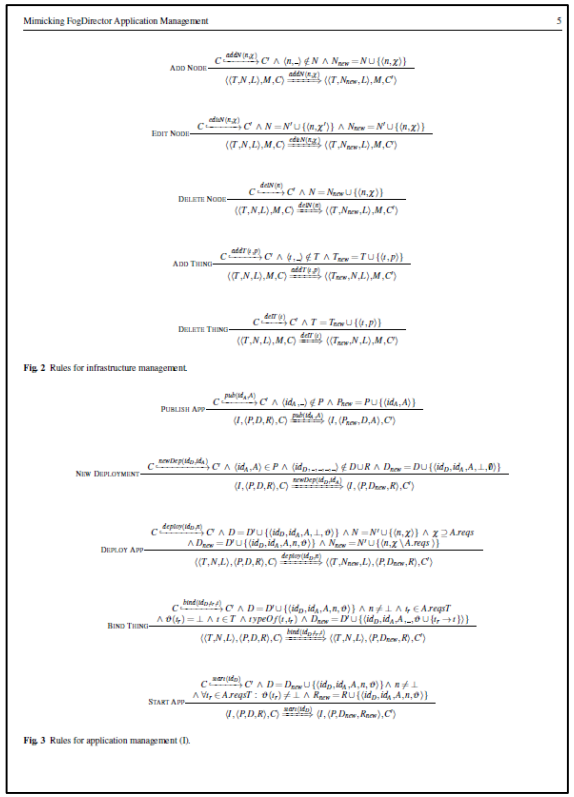
March 30, 2018

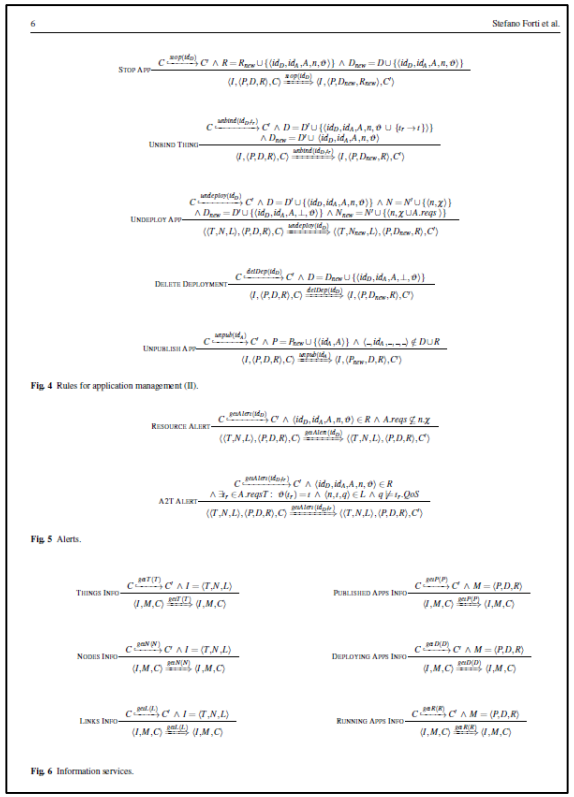
Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

!!!
160 PAGES
!!!

Solution #1

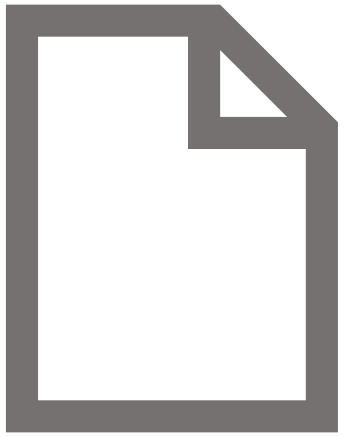
- **Operational semantics** of all **basic functionalities** of FogDirector.
- **Compact** and **unambiguous** (basic) reference.





Forti, Stefano, Ahmad Ibrahim, and Antonio Brogi. "Mimicking FogDirector Application Management.», Software-Intensive Cyber-Physical Systems (In Press).

Problem #2

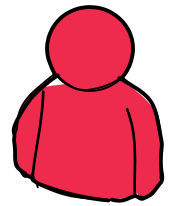


write **correct** and **effective** management

Solution #2

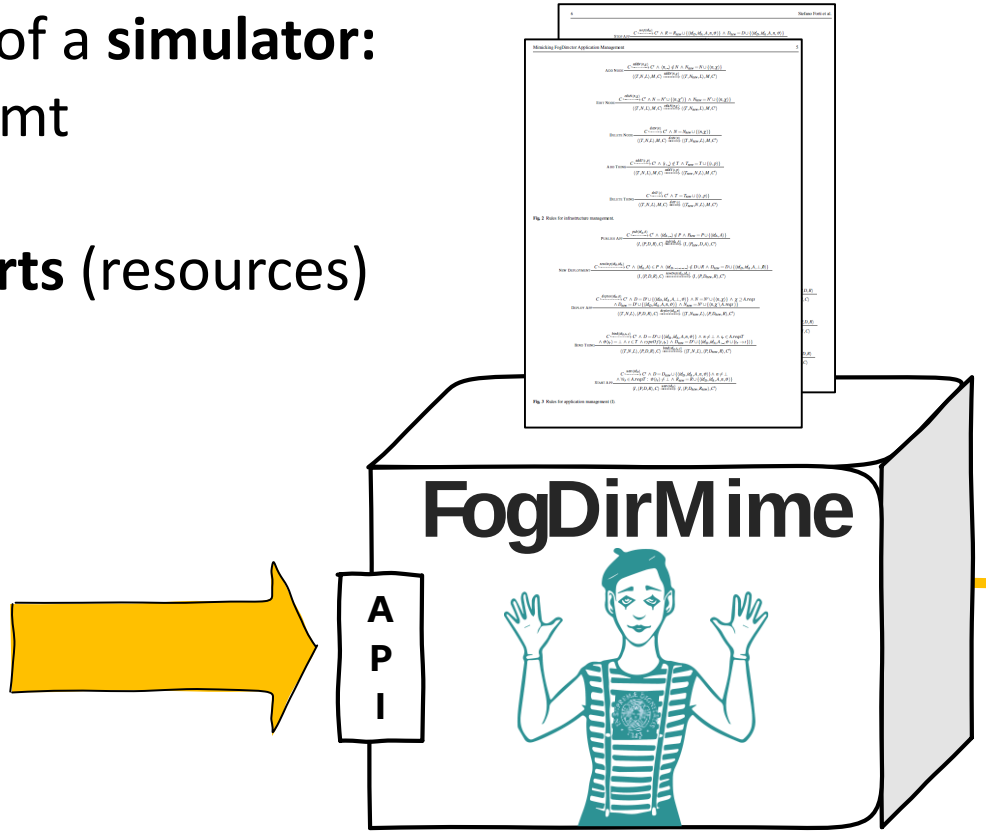
FogDirMime is the core of a **simulator:**

- **Infrastructure** mgmt
- **App** mgmt
- **Monitoring** & **Alerts** (resources)



app admin

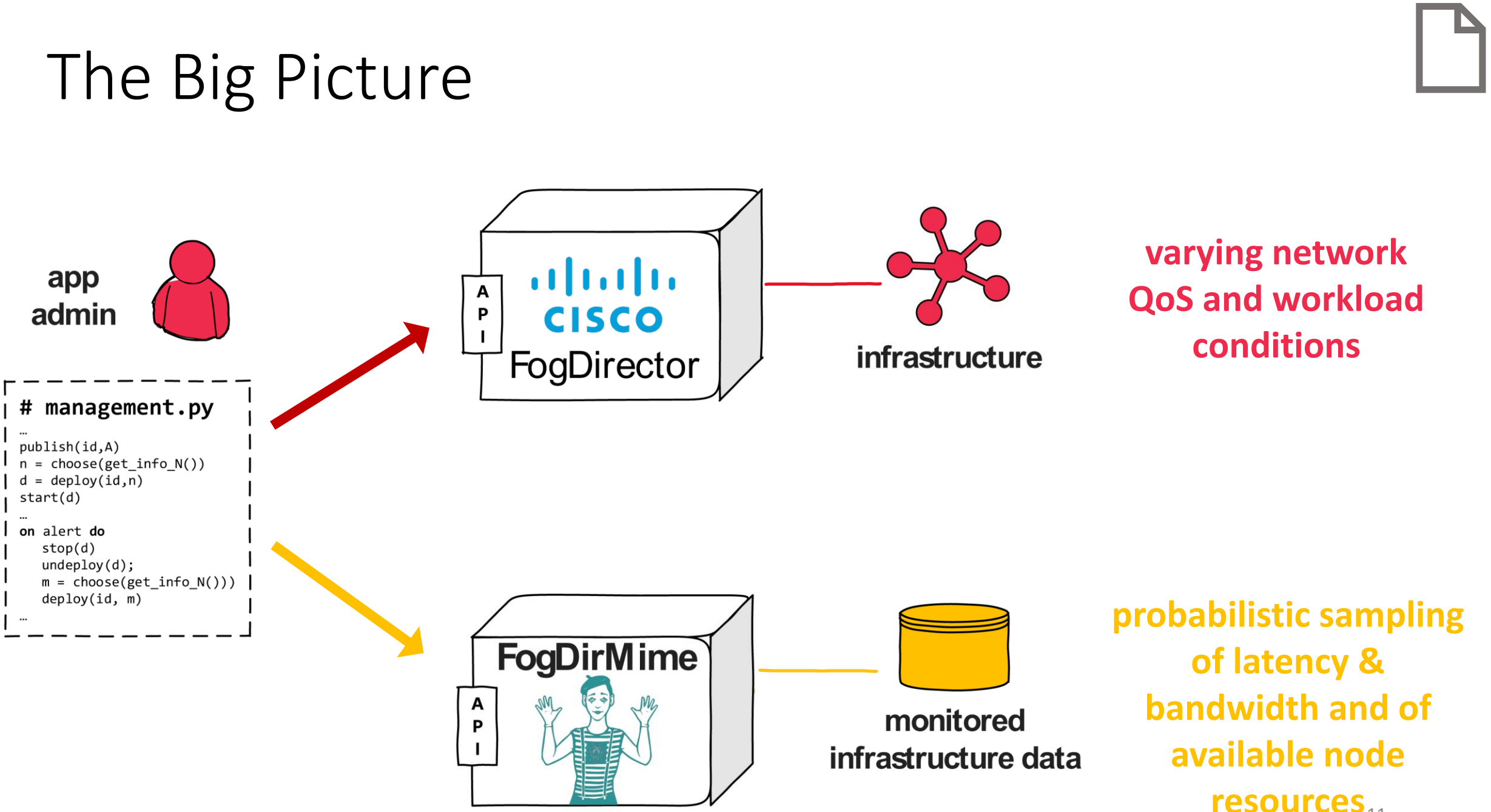
```
# management.py
…
publish(id,A)
n = choose(get_info_N())
d = deploy(id,n)
start(d)
…
on alert do
   stop(d)
   undeploy(d);
   m = choose(get_info_N()))
   deploy(id, m)
…
```



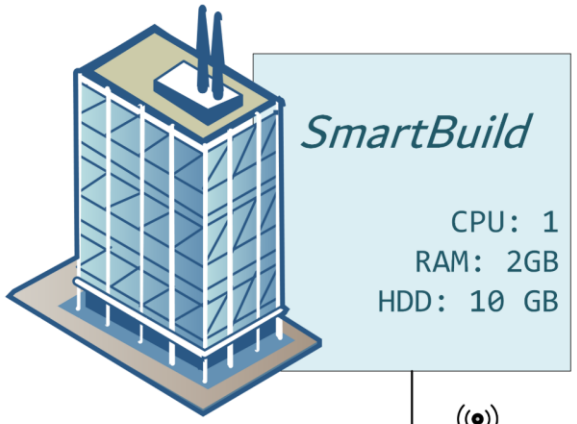
monitored infrastructure data

https://github.com/di-unipi-socc/FogDirMime

The Big Picture



A (simple) example

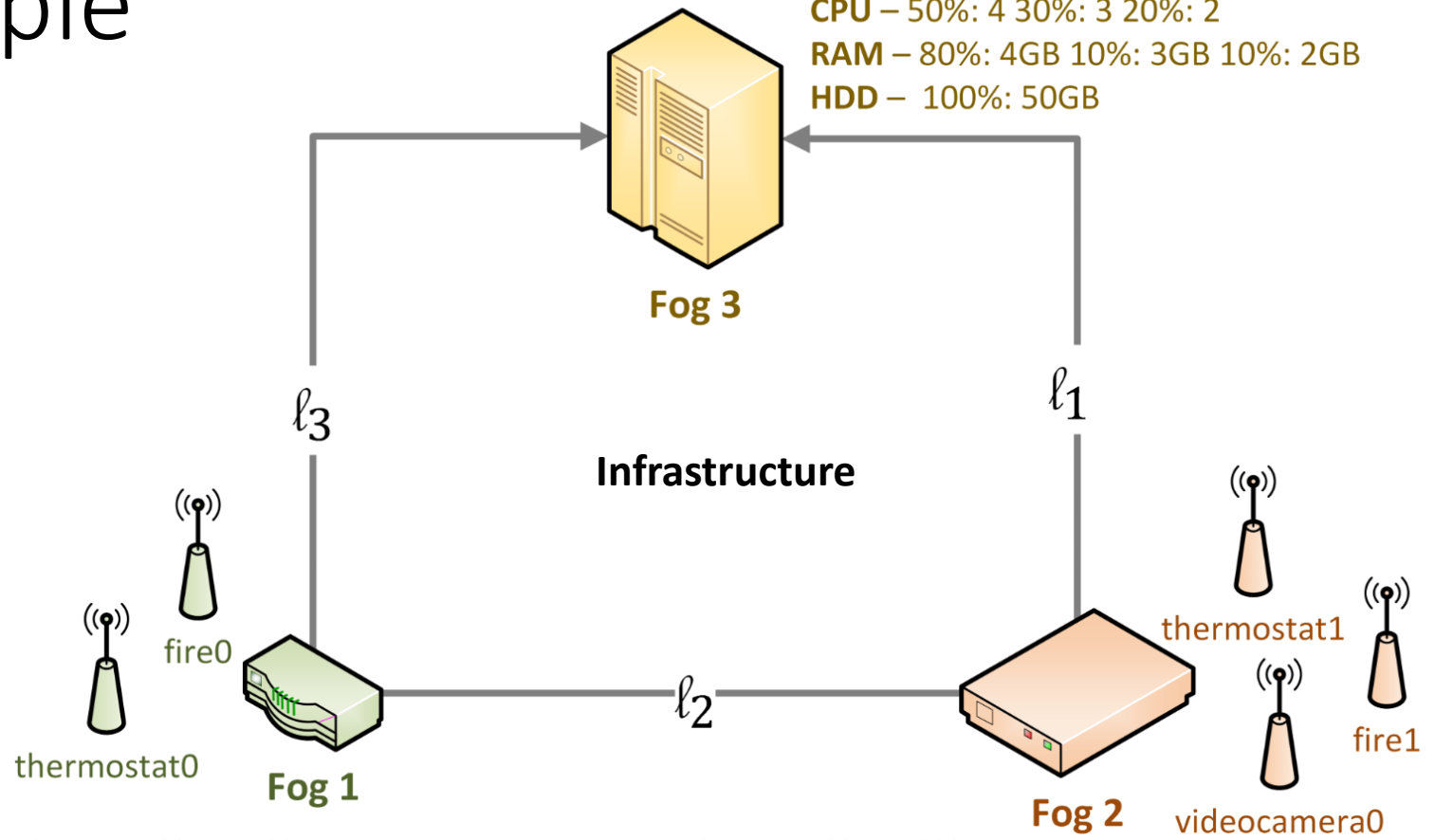


thermostat:
latency: 1 s
bw_a2t: 0.1 Mbps
bw_t2a: 0.1 Mbps

fire:
latency: 65 ms
bw_a2t: 0.1 Mbps
bw_t2a: 0.1 Mbps

video:
latency: 30 ms
bw_a2t: 0.1 Mbps
bw_t2a: 5.0 Mbps

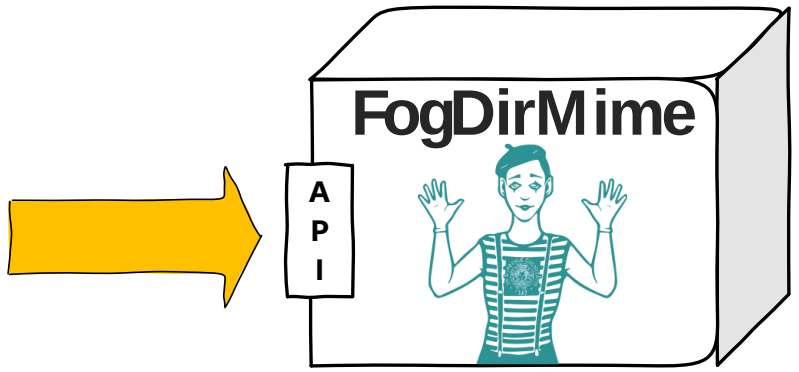
App

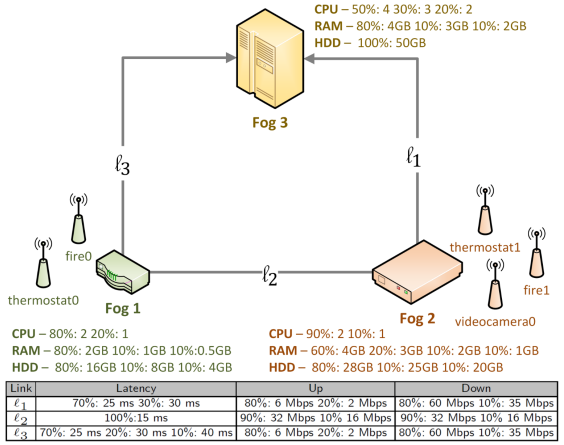


Link	Latency	Up	Down
ℓ_1	70%: 25 ms 30%: 30 ms	80%: 6 Mbps 20%: 2 Mbps	80%: 60 Mbps 10%: 35 Mbps
ℓ_2	100%:15 ms	90%: 32 Mbps 10% 16 Mbps	90%: 32 Mbps 10% 16 Mbps
ℓ_3	70%: 25 ms 20%: 30 ms 10%: 40 ms	80%: 6 Mbps 20%: 2 Mbps	80%: 60 Mbps 10%: 35 Mbps

A (simple) example

```
for i in range(0, epochs):
    alerts1=fd.get_alert('dep1')
    alerts2=fd.get_alert('dep2')
    #[...] collecting user defined stats about the alerts [...]
    #In case of low resources at Fog2, move to Fog3
    for alert in alerts1:
        if alert['alert_type'] =='resources' and not(moved1):
            migrations1 +=1
            fd.stop_app('dep1')
            fd.undeploy_app('dep1', 'SmartBuild')
            while fd.deploy_app('dep1', 'SmartBuild', 'fog_3')
                !=1:
                continue
            fd.start_app('dep1')
            moved1 =True
            break
    #In case of low resources at Fog1, move to Fog2 and
        viceversa
    for alert in alerts2:
        if alert['alert_type'] =='resources':
            migrations2 +=1
            fd.stop_app('dep2')
            fd.undeploy_app('dep2', 'SmartBuild')
            if not(moved2):
                fog_node ='fog_2'
            else:
                fog_node ='fog_1'
            while fd.deploy_app('dep2', 'SmartBuild',
                fog_node) !=1:
                continue
            fd.start_app('dep2')
            moved2 =not(moved2)
            break
    alerts1, alerts2 =[], []
```





Link	Latency	Up	Down
ℓ_1	70%: 25 ms 30%: 30 ms	80%: 6 Mbps 20%: 2 Mbps	80%: 60 Mbps 10%: 35 Mbps
ℓ_2	100%:15 ms	90%: 32 Mbps 10% 16 Mbps	90%: 32 Mbps 10% 16 Mbps
ℓ_3	70%: 25 ms 20%: 30 ms 10%: 40 ms	80%: 6 Mbps 20%: 2 Mbps	80%: 60 Mbps 10%: 35 Mbps

```
Simulating management plan for 10000 epochs.
*** RESULTS ***
*** dep1 ***
        Resource alerts: 0.01%
        A2T alerts: 20.07%
        Migrations: 0.01%

*** dep2 ***
        Resource alerts: 16.11%
        A2T alerts: 0.0%
        Migrations: 16.11%
```

FogDirMime limitations

The mime uses an abstract representation of FD APIs

FogDirMime

```
deploy(appid, name, node)
```

FogDirector

```
POST /api/v1/appmgr/myapps/<myappid>/action
```

FogDirMime limitations

It does not return any metric by itself

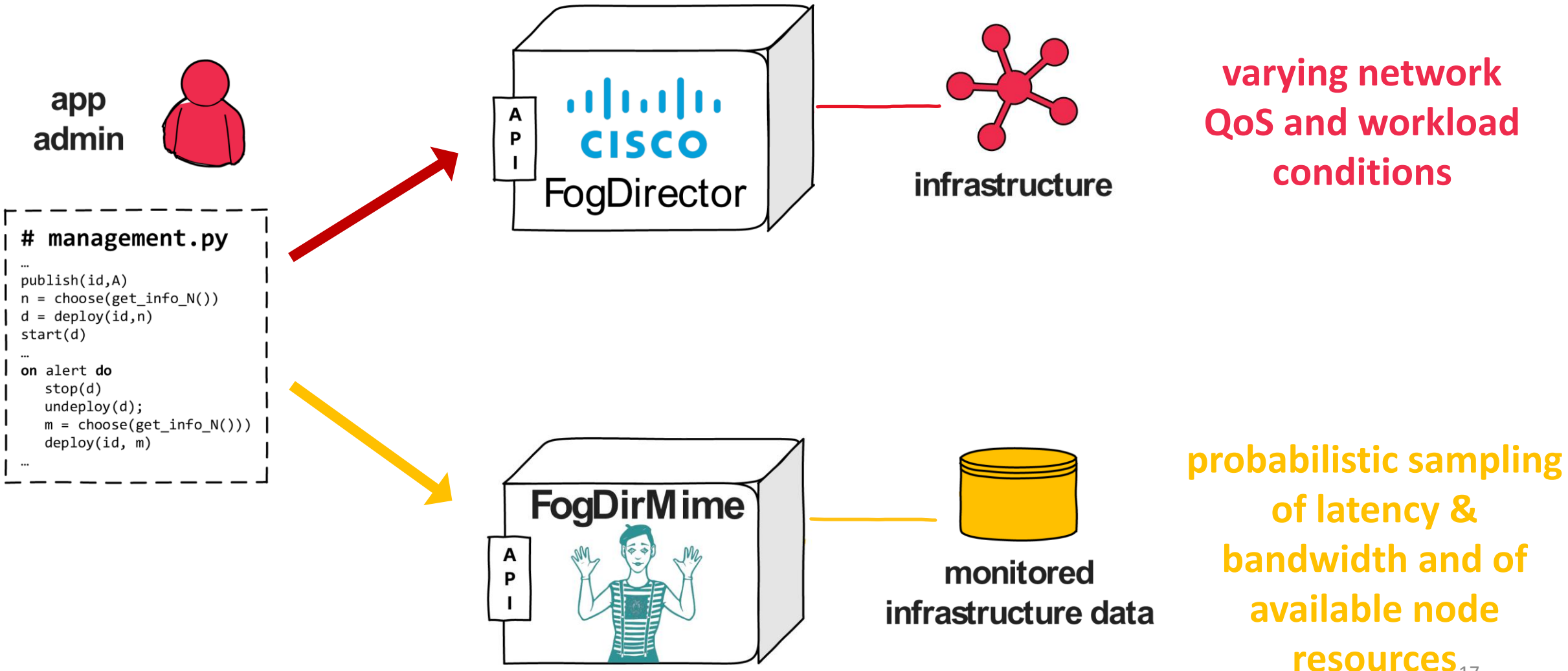
If you want, you can compute custom metrics by collecting and aggregating the data in the program you write

```
for i in range(0, epochs):
    alerts1=fd.get_alert('dep1')
    alerts2=fd.get_alert('dep2')
    #[...] collecting user defined stats about the alerts [...]
    #In case of low resources at Fog2, move to Fog3
    for alert in alerts1:
        if alert['alert_type'] =='resources' and not(moved1):
            migrations1 +=1
            fd.stop_app('dep1')
            fd.undeploy_app('dep1', 'SmartBuild')
            while fd.deploy_app('dep1', 'SmartBuild', 'fog_3')
                !=1:
                continue
            fd.start_app('dep1')
            moved1 =True
            break
    #In case of low resources at Fog1, move to Fog2 and
        viceversa
    for alert in alerts2:
        if alert['alert_type'] =='resources':
            migrations2 +=1
            fd.stop_app('dep2')
            fd.undeploy_app('dep2', 'SmartBuild')
            if not(moved2):
                fog_node ='fog_2'
            else:
                fog_node ='fog_1'
            while fd.deploy_app('dep2', 'SmartBuild',
                fog_node) !=1:
                continue
            fd.start_app('dep2')
            moved2 =not(moved2)
            break
    alerts1, alerts2 =[], []
```

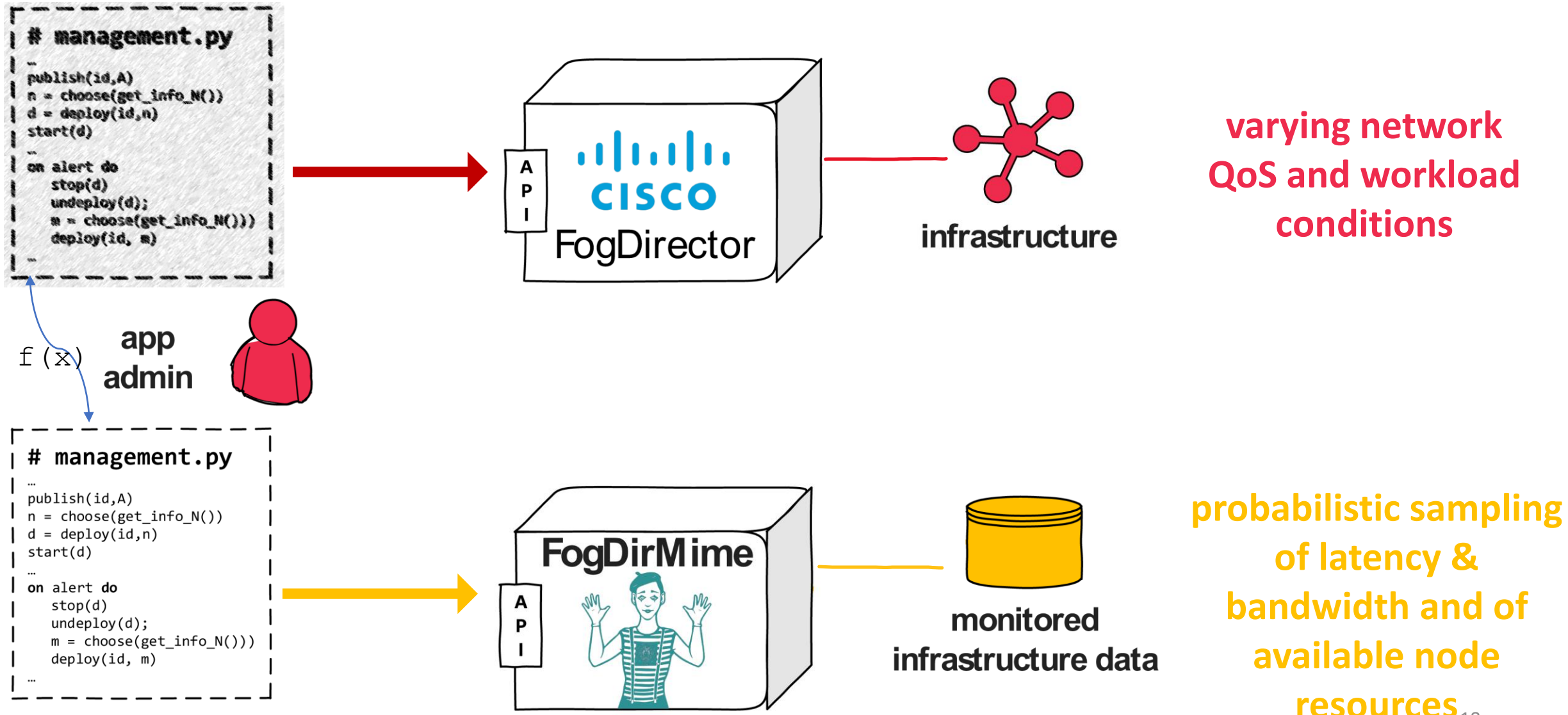
Objectives of FogDirSimulator

- FDS tackles FogDirMime limitations by
  - Offering the actual FogDirectorAPI so to be able to run real management scripts
  - Engineering the simulator s.t. automatically collect and aggregate data on management executions
- Create an enriched framework to evaluate a QoS-aware application management

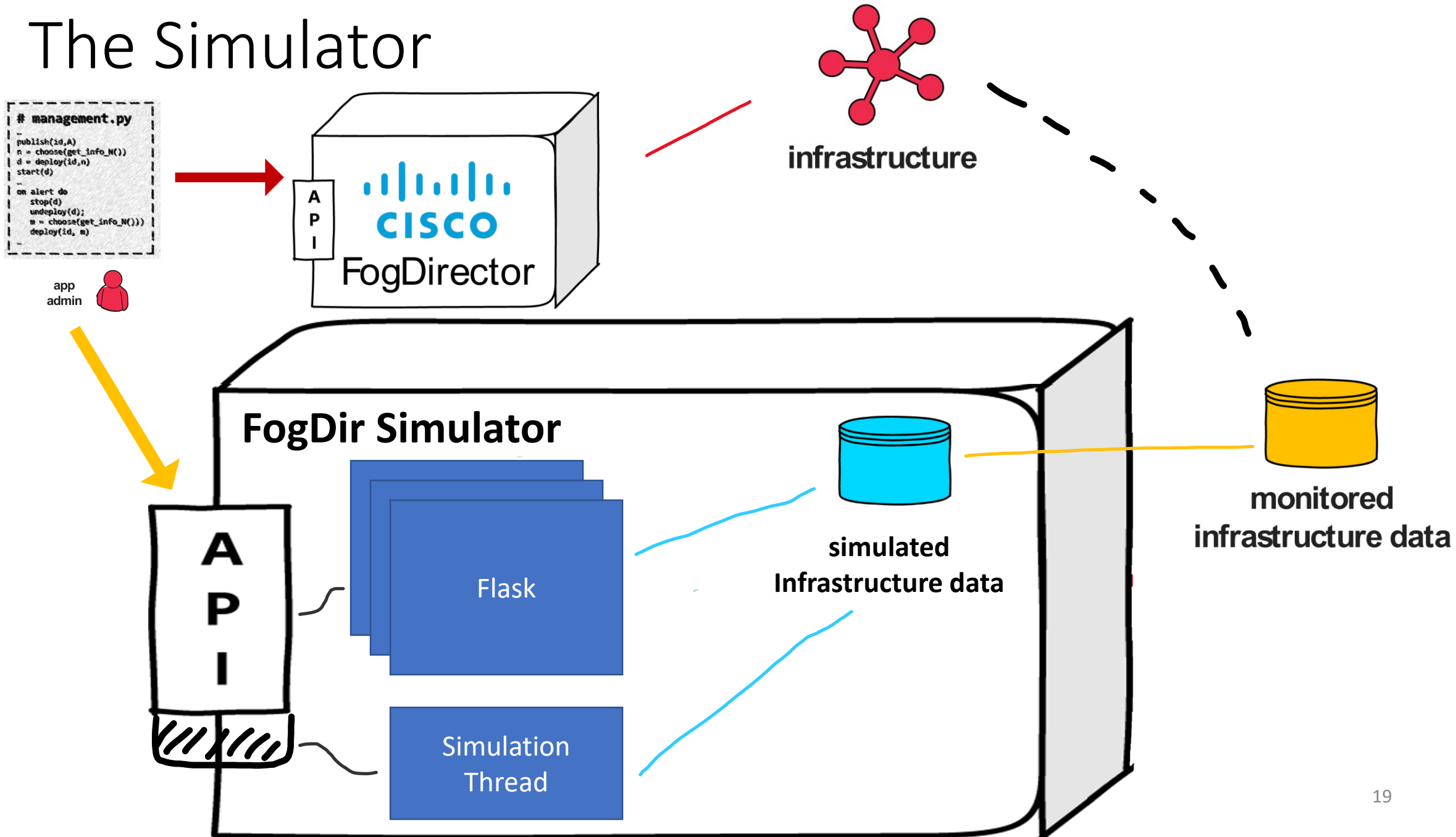
The Big Picture



The Big Picture



The Simulator

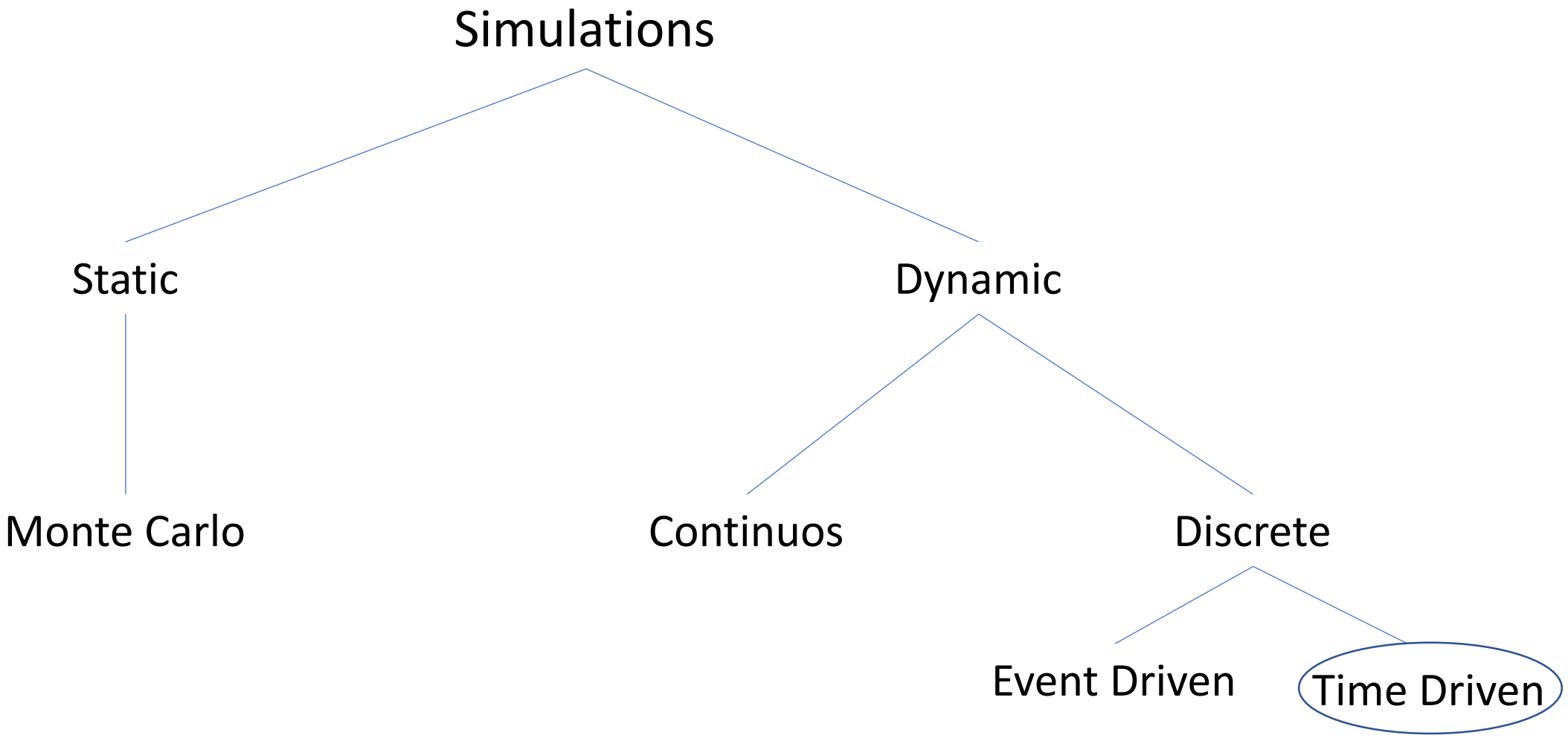


FogDirSimulator Semantics

$$\frac{I \rightsquigarrow I' \ \wedge \ \langle I', M, C \rangle \overset{\ell}{\Rightarrow} \langle I'', M'', C'' \rangle}{\langle I, M, C \rangle \Rrightarrow \langle I'', M'', C'' \rangle}$$

$$I = Infrastructure$$
$$M = Managed\ Apps$$
$$C = Client\ Management\ Script$$

Different Simulation Models



How to evaluate Script Management?

- We have enqueried the major players in the app management field::
  - Docker Swarm (Docker employee)
  - Kubernetes (Reddit & StackOverflow)

None of them have a model to evaluate their management policies.
They only look for the convergence.

Simulation Thread

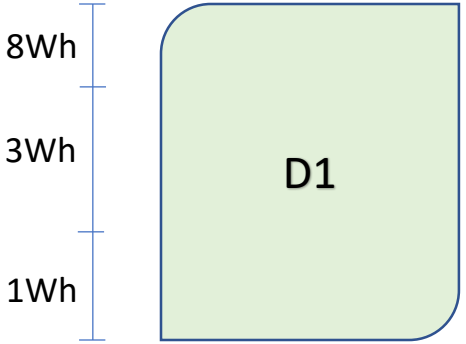
- It samples data from database and modifies the infrastructure
- It generates some alerts
- It estimates a vector of Key Performance Indicators (KPI) about management executions
  - Device Critical Alerts
  - Application Uptime and Downtime
  - Devices on which the application is installed
  - Migration counting for every application, computed with heuristics since migration is not a primitive function
- Stopping Criterion

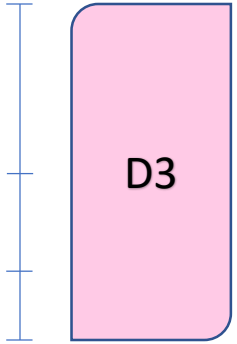
$$|KPI_t - KPI_{t-1}| < \varepsilon$$

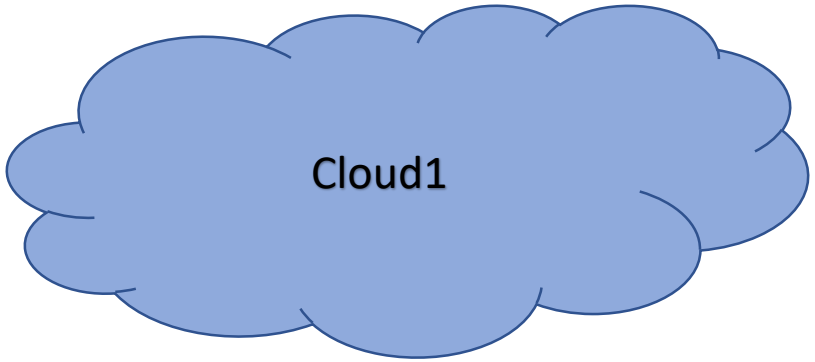
Application Uptime and Downtime

- It executes an unbounded loop, read data from database and increments some variables
- When you call the simulator API, it returns the computed KPIs
- It features a GUI to show you that data

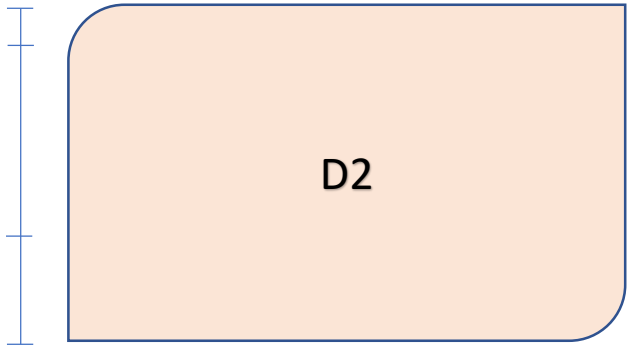
From KPI to useful data - Energy

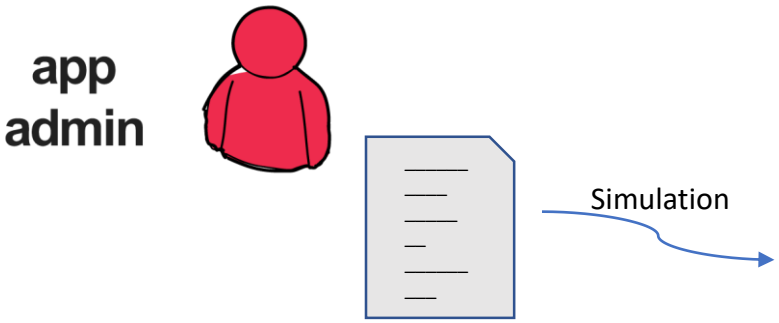






Application A probably consumes, in a week, 4kWh

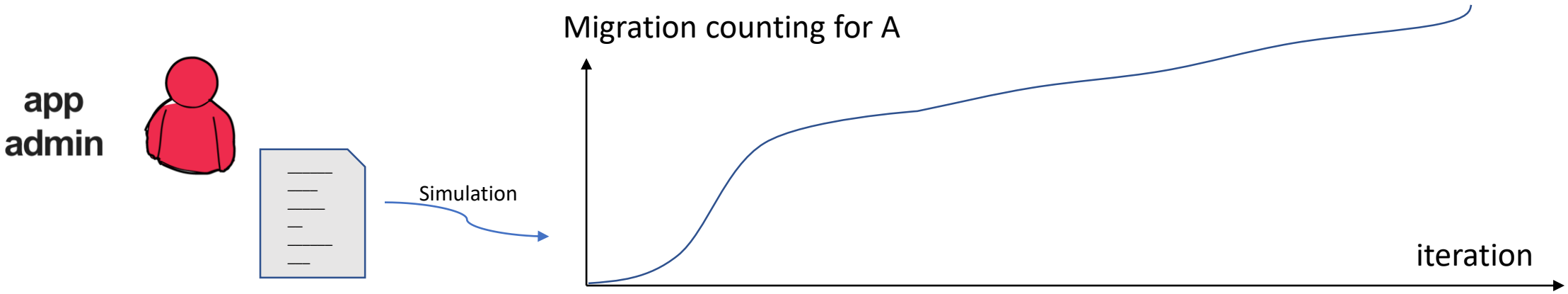




Application A:
75% -> D1
20% -> D2
5% -> D3

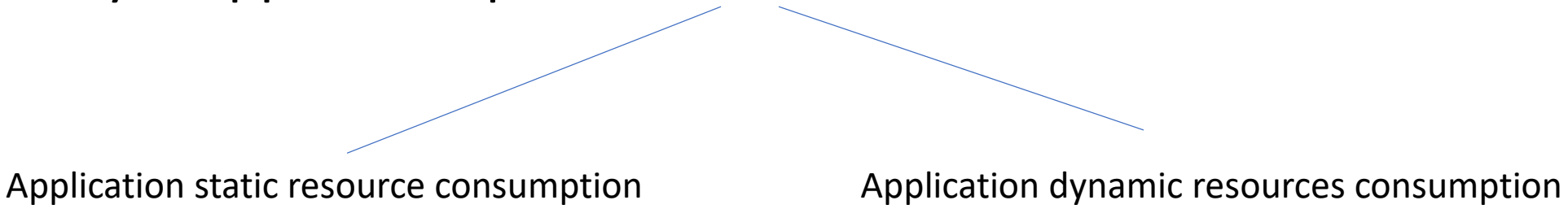
From KPI to useful data – Migration Analysis

- *Migration(t)* = returns the number of migration that app A went trough from iteration 0 to iteration t
- It is a monotonically non-decreasing function
- Since iteration and time are not related, we cannot use this values as absolute values
- We can analyse that function in order to understand if the management finds a equilibrium point



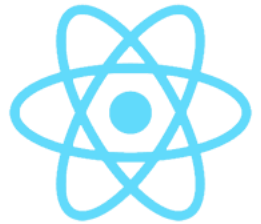
Differences between the Simulator and FogDirector

- FogDirector executes applications, we don’t.
- Since we don’t execute applications, we have found two methods to analyse application profile



- The simulator doesn’t simulate latency (application install is very fast)
- The simulator doesn’t simulate link fault

A brief overview on the tool set



React

FlaskRESTful

pytest


SHOW ME, DON'T TELL ME

A simple Management Script

```
from APIWrapper import FogDirector
import time

fg = FogDirector("127.0.0.1:5000")
code = fg.authenticate("admin", "admin_123")
if code == 401:
        print "Failed Authentication"

# Adding devices
_, device = fg.add_device("10.10.20.51", "cisco", "cisco")
fg.add_device("10.10.20.52", "cisco", "cisco")

# Uploading Application
code, localapp = fg.add_app("./NettestApp2V1_lxc.tar.gz", publish_on_upload=True)

# Creating myapp endpoint
myappname = "MyFirstApp"
_, myapp = fg.create_myapp(localapp["localAppId"], myappname)
```

```
# Deploying on Devices with default resources
code, res = fg.install_app(myappname, ["10.10.20.51"])
while code == 400:
        code, res = fg.install_app(myappname, ["10.10.20.51"])

code, res = fg.install_app(myappname, ["10.10.20.52"])
while code == 400:
        code, res = fg.install_app(myappname, ["10.10.20.52"])

fg.start_app(myappname)

fg.add_app("./TestApp2.tar.gz", publish_on_upload=True)
# Creating myapp endpoint
_, localapps = fg.get_apps()
app = localapps["data"][1]
localAppId = app["localAppId"]
myappname = "TestApp"
_, myapp = fg.create_myapp(localAppId, myappname)

time.sleep(5)
code, res = fg.install_app(myappname, ["10.10.20.52"])
while code == 400:
        code, res = fg.install_app(myappname, ["10.10.20.52"])
```

```
def otherDevice(actual):
    if actual == "10.10.20.52":
        return "10.10.20.51"
    else:
        return "10.10.20.52"

while True:
    time.sleep(5)
    _, alerts = fg.get_alerts()
    for alert in alerts["data"]:
        if 0 == alert["type"]: #
            if alert["appName"] == "FirstMyApp":
                fg.stop_app("FirstMyApp")
                fg.uninstall_app("FirstMyApp", alert["ipAddress"])
                fg.install_app("FirstMyApp", [otherDevice(alert["ipAddress"])])
                fg.start_app("FirstMyApp")
```

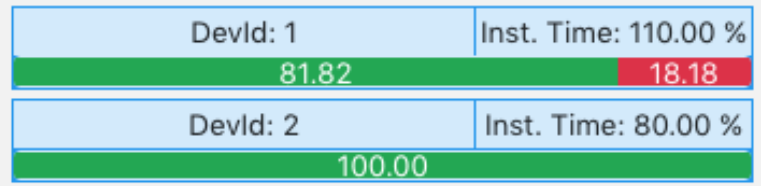
```
db.Rdevices.insert_one({
            "ipAddress": "10.10.20.51",
            "port": 8443,
            "deviceId": 1,
            "totalCPU": 1000,
            "totalMEM": 128,
            "distributions": {
                "CPU": [
                    {
                        "timeStart": 0,
                        "timeEnd": 24,
                        "mean": 90,
                        "deviation": 10
                    }
                ],
                "MEM": [
                    {
                        "timeStart": 0,
                        "timeEnd": 24,
                        "mean": 110,
                        "deviation": 2
                    }
                ]
            }
        })
```

```
db.Rdevices.insert_one({
            "ipAddress": "10.10.20.52",
            "port": 8443,
            "deviceId": 2,
            "totalCPU": 1000,
            "totalMEM": 128,
            "distributions": {
                "CPU": [
                    {
                        "timeStart": 0,
                        "timeEnd": 24,
                        "mean": 600,
                        "deviation": 100
                    }
                ],
                "MEM": [
                    {
                        "timeStart": 0,
                        "timeEnd": 24,
                        "mean": 80,
                        "deviation": 2
                    }
                ]
            }
        })
```

Home Devices MyApps Cost

#	Device ID	IP Address	Port	CRITICAL CPU	MEMORY UNDER HIGH PRESSURE
0	1	10.10.20.51	8443	53.72233400402414 %	0 %
1	2	10.10.20.52	8443	0 %	0 %

Home Devices MyApps Cost

#	MyApp ID	Name	Installed Time	Uninstalled Time	Distribution Over Device
0	5c1225bc65d09008626782c9	MyFirstApp	45.45 %	54.55 %	
1	5c1225bf65d09008626782f0	TestApp	0.00 %	100.00 %	

Home Devices MyApps Cost

#	MyApp ID	Name	Installed Time	Uninstalled Time	Distribution Over Device
0	5c1225bc65d09008626782c9	MyFirstApp	79.31 %	20.69 %	DevId: 1 Inst. Time: 102.17 % 95.74 4. DevId: 2 Inst. Time: 97.83 % 100.00
1	5c1225bf65d09008626782f0	TestApp	62.79 %	37.21 %	DevId: 2 Inst. Time: 100.00 % 100.00

Home Devices MyApps Cost

#	MyApp ID	Name	Installed Time	Uninstalled Time	Distribution Over Device
0	5c12251c65d09007a3c20546	MyFirstApp	99.44 %	0.56 %	
1	5c12251d65d09007a3c2055c	TestApp	96.00 %	4.00 %