FogTorchΠ:

How to best deploy your Fog applications, probably*

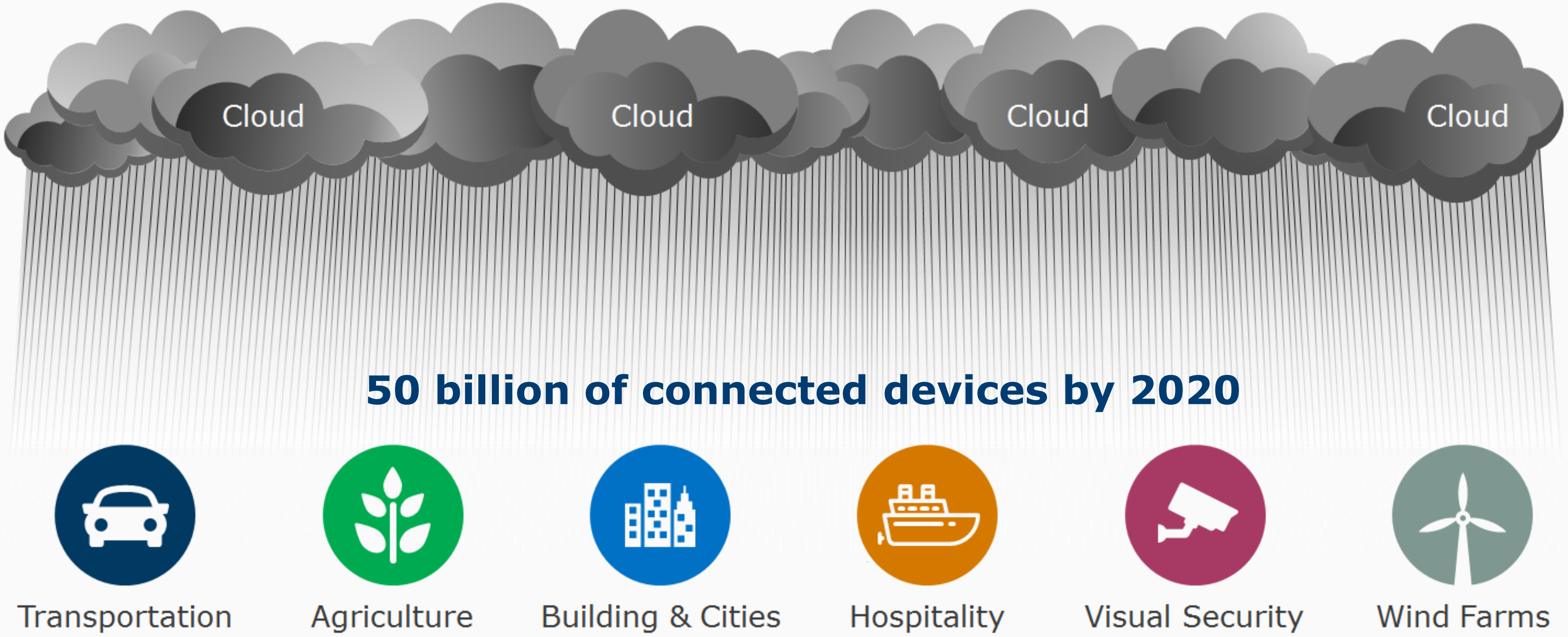
Antonio Brogi

Stefano Forti

Ahmad Ibrahim

*Antonio Brogi, Stefano Forti, Ahmad Ibrahim, **How to best deploy your Fog applications, probably.** *Accepted at* ICFEC'17.

IoT and Cloud Computing



50 billion of connected devices by 2020

- The Cloud alone cannot support the **IoT momentum**.
- There is a need for **filtering** and **processing** *before* the Cloud.

Fog Features

QoS-awareness

- App deployments dynamically adapt to the **state** of the network.

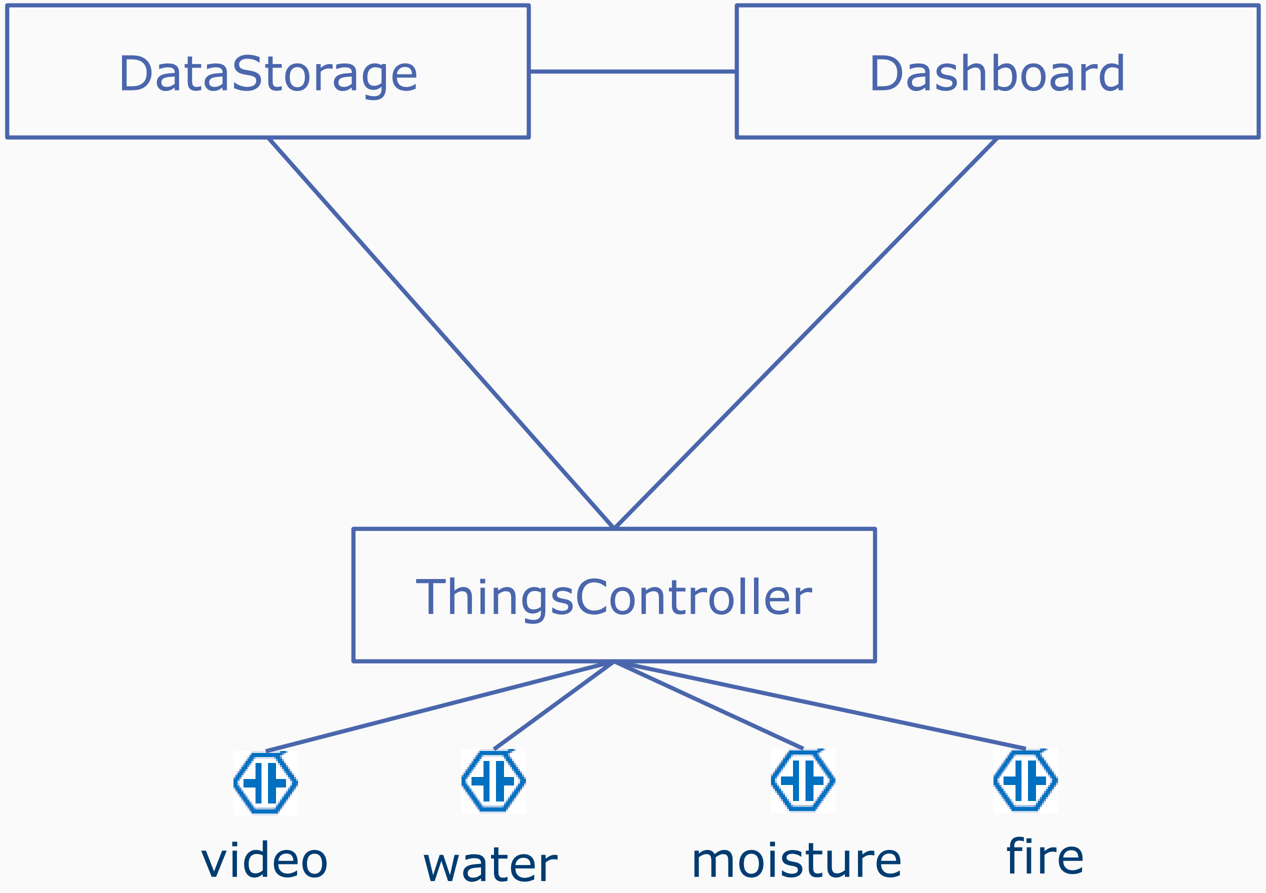
Location-awareness

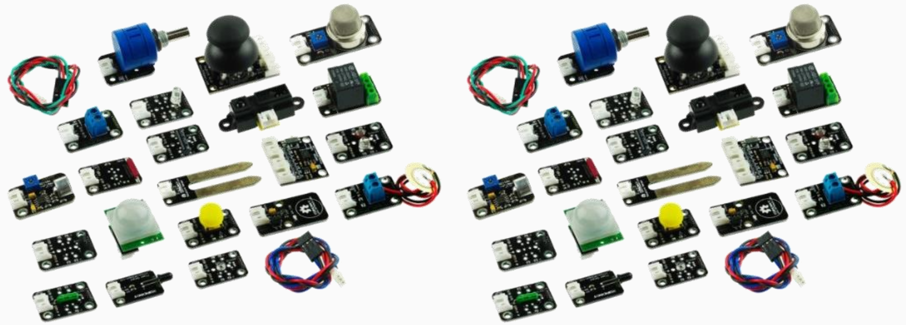
- **Position** is known so to handle fluid and mobile computation.

Context-awareness

- Discover and use available resources, **cooperating** horizontally.

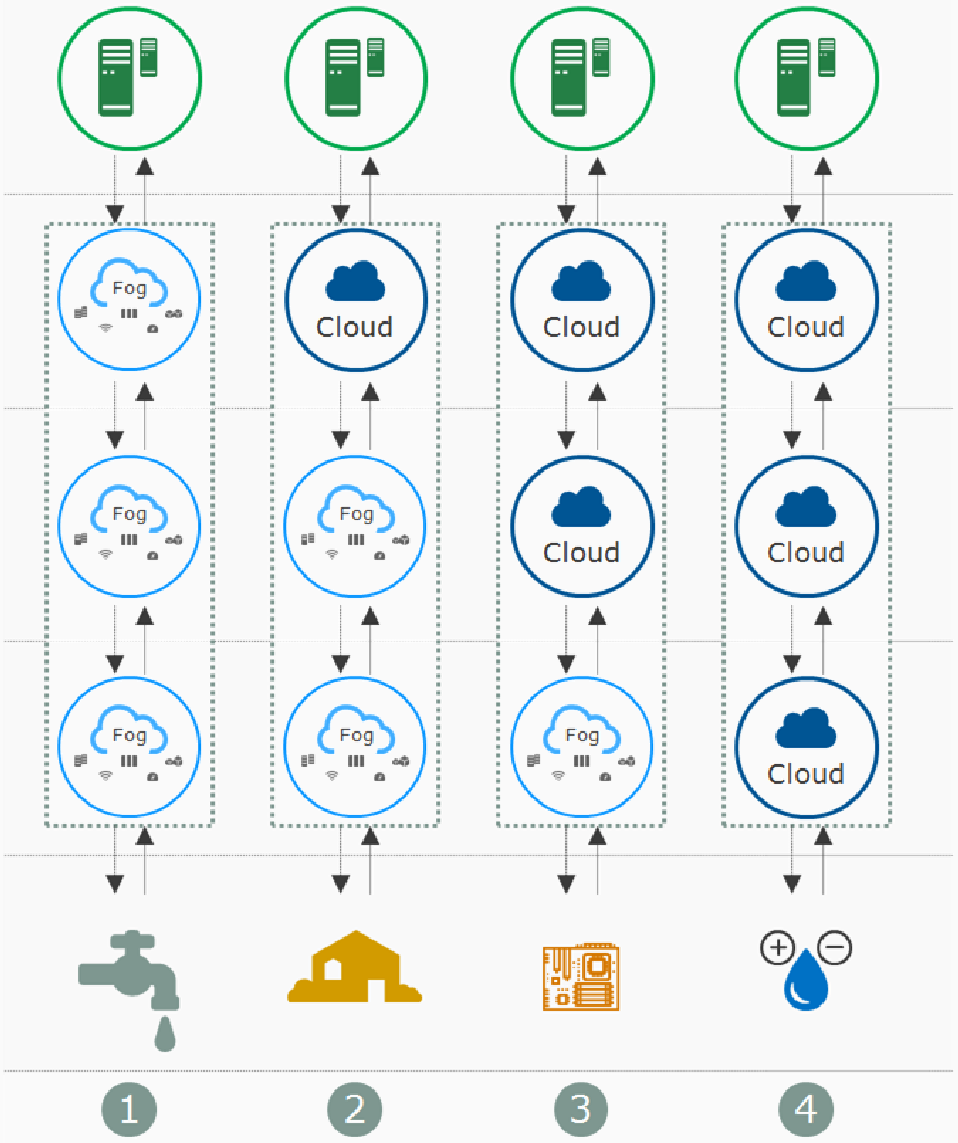
Motivating example





Open Problems

- How to **automatically** decide *where* to deploy each component of an application by exploiting QoS-, location-, and context-awareness?
- How to estimate **QoS-assurance** of a candidate deployment?



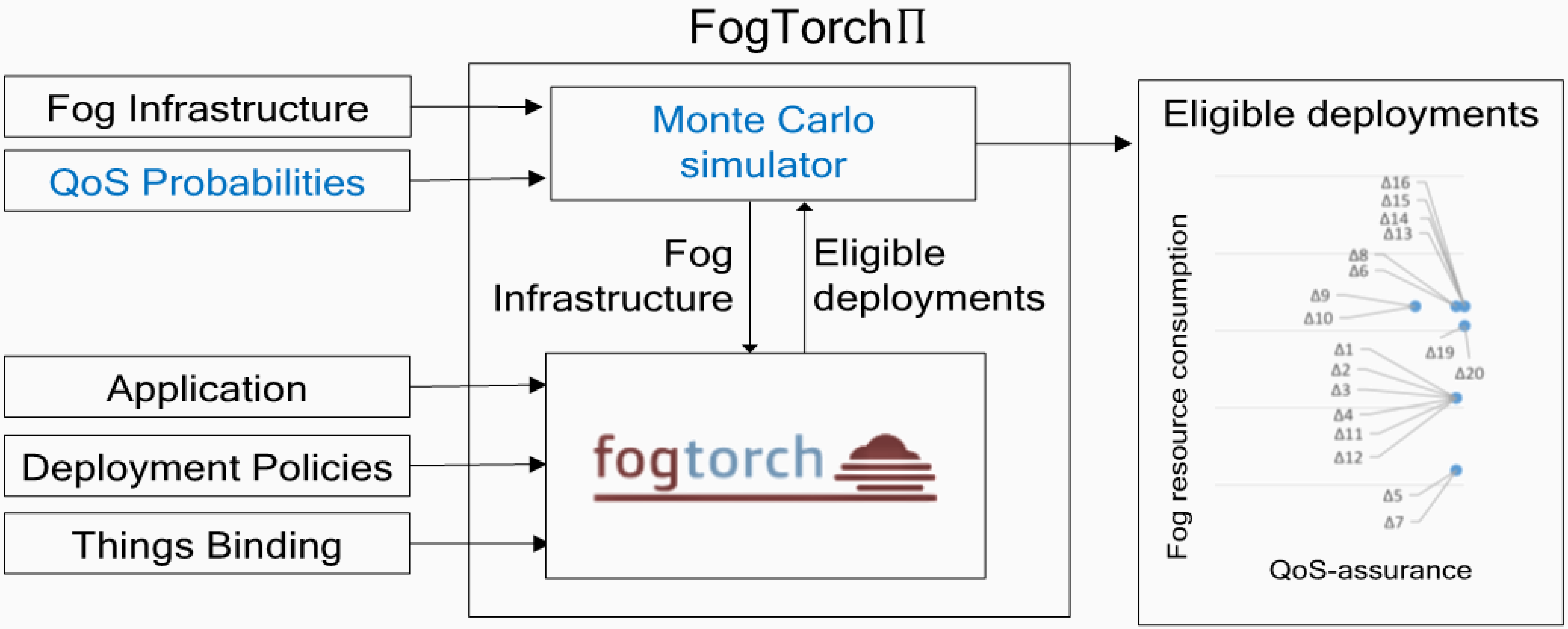
Concretely

Do I have to **upgrade** my infrastructure if the application requirements change?

Is it possible to **reduce** resource consumption of some Fog nodes, or **avoid** them?

Which are the eligible deployments that **comply most** with the required **QoS**?

FogTorchΠ



FogTorchΠ

Fog Infrastructure

QoS Probabilities

Application

Deployment Policies

Things Binding

Monte Carlo simulator

Fog Infrastructure

Eligible deployments

fogtorch

Eligible deployments

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

QoS Profiles

- A **QoS profile** is a pair

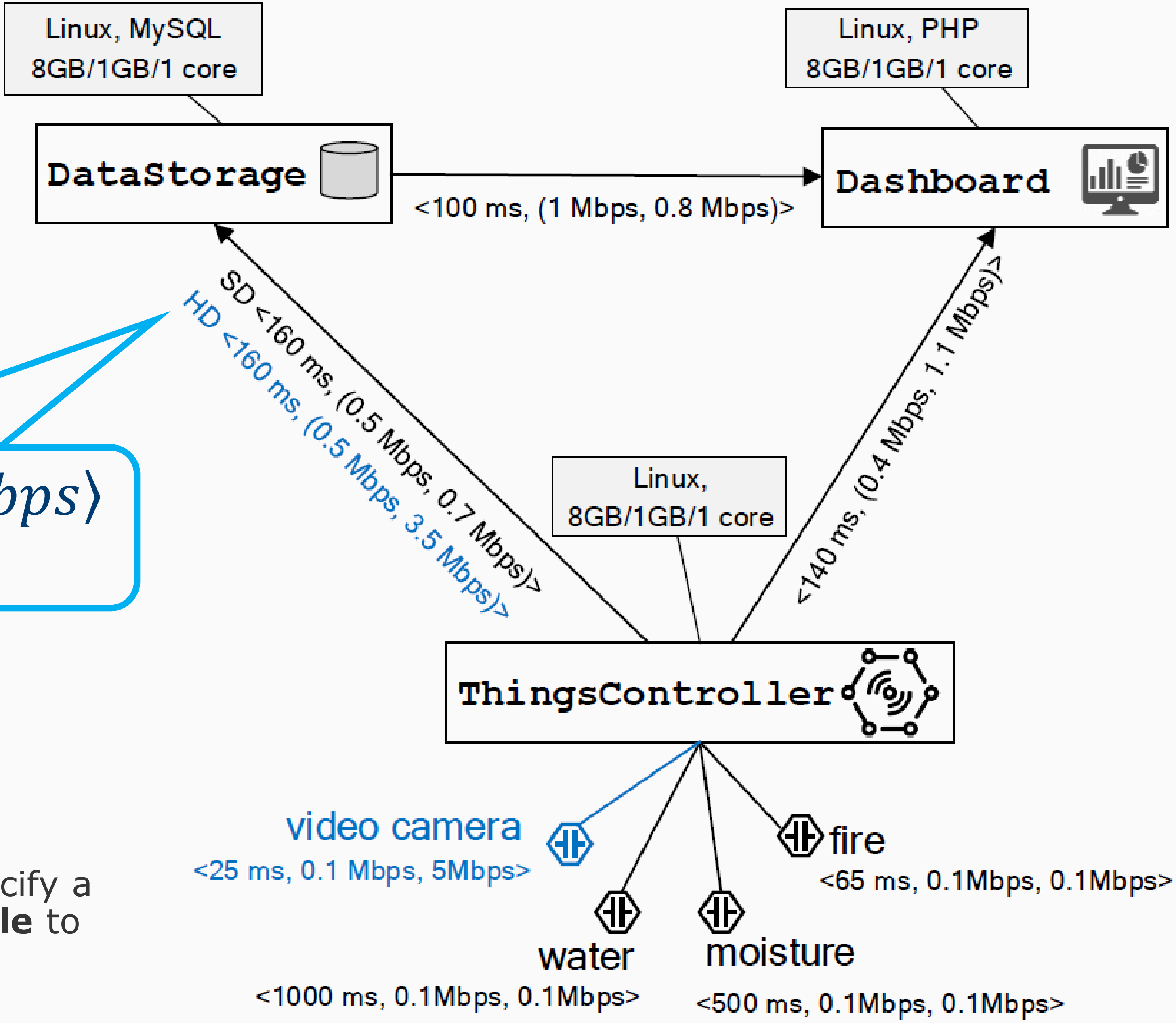
$$\langle \ell, \langle b_{\downarrow}, b_{\uparrow} \rangle \rangle$$

AVERAGE **LATENCY**

AVERAGE DOWNLOAD & UPLOAD **BANDWIDTH**

- They represent latency and bandwidth **featured by** a link **or requested** by a software interaction.

Application


Linux, MySQL
8GB/1GB/1 core
DataStorage
Linux, PHP
8GB/1GB/1 core
Dashboard
<100 ms, (1 Mbps, 0.8 Mbps)>
SD <160 ms, (0.5 Mbps, 0.7 Mbps)>
HD <160 ms, (0.5 Mbps, 3.5 Mbps)>
<140 ms, (0.4 Mbps, 1.1 Mbps)>
Linux,
8GB/1GB/1 core
ThingsController
video camera
<25 ms, 0.1 Mbps, 5Mbps>
fire
<65 ms, 0.1Mbps, 0.1Mbps>
water
<1000 ms, 0.1Mbps, 0.1Mbps>
moisture
<500 ms, 0.1Mbps, 0.1Mbps>

$\langle 160\ ms, 0.5\ Mbps, 0.7\ Mbps \rangle$
SD video

- **Multicomponent** applications.
- **Interactions** between components associated to a desired **QoS profile**.
- Things requests for each component specify a **type of Thing** with a desired **QoS profile** to access it.

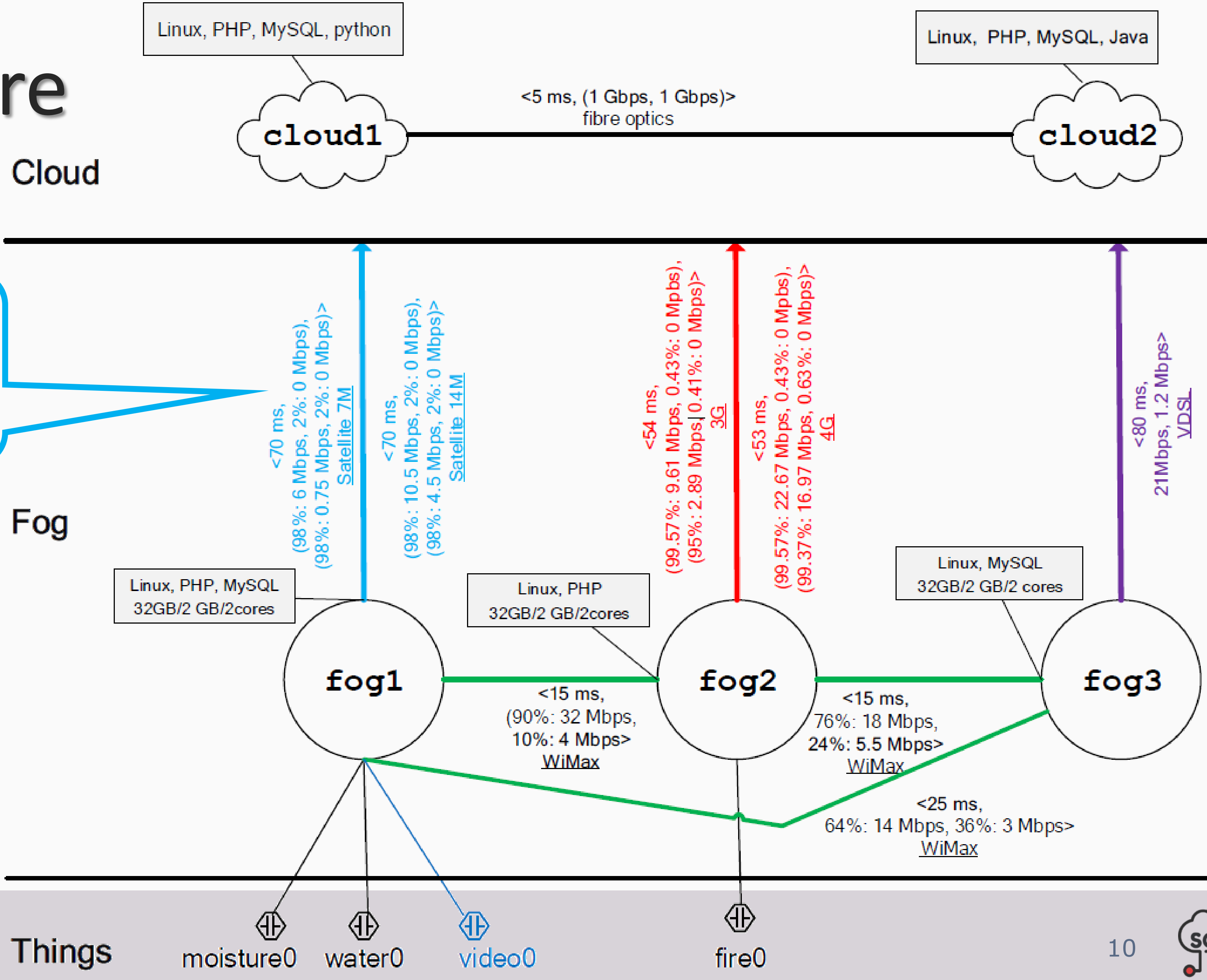
Infrastructure

$98\% \ \langle 70\ ms, 6\ Mbps, 0{,}75\ Mbps \rangle$

$2\% \ \langle 70\ ms, 0 Mbps, 0\ Mbps \rangle$

Satellite 7M

- Things, Fog and Cloud nodes have a **location** (e.g., GPS).
- Fog nodes feature **hardware**, **software** and **connected Things**.
- Clouds feature **software**, **hardware** is not considered (**unbounded**).



Deployment Policy

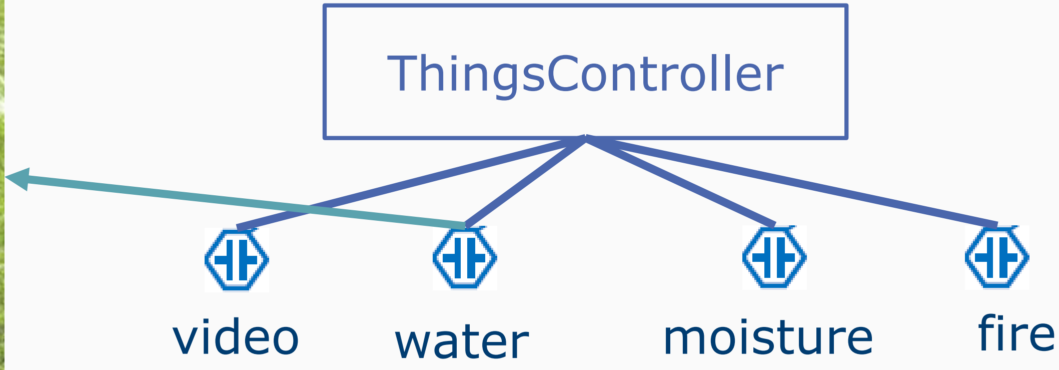
- A **start-up** sponsored by a specific Cloud provider,
- an **automated industrial** plant,
- an invoked **third party service**...

...may enforce **legal**, **commercial** or **political** constraints for deploying an application.

- We allow specification of a *whitelist* of nodes permitted for installing each component.

Things Binding

- Software components may have Things requests.
- Each request is bound to a **specific Thing** before deployment.



ThingsController

video

water

moisture

fire

Eligible Deployments

- An **eligible deployment** for an application over a Fog infrastructure

  1. satisfies **compatibility** and **deployment policies**,
  2. does not exceed **hardware capacity** at each Fog node,
  3. satisfies **Things requests binding**,
  4. does not exceed **available links bandwidth** for interactions and remote Things access.

Backtracking strategy to explore the search space.

NP-hard Problem*

[Garey, Michael R., and David S. Johnson. Computers and Intractability (1979)]

"I can't find an efficient algorithm, but neither can all these famous people."

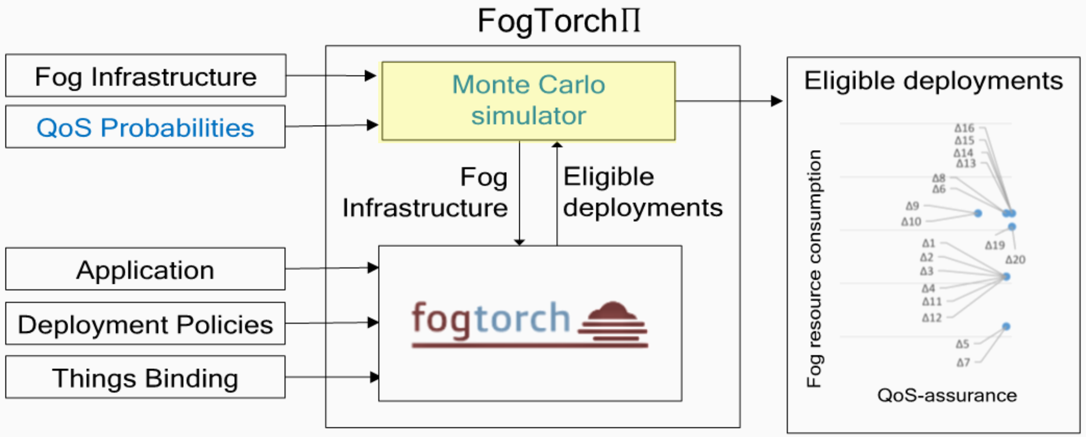
* By reduction from Subgraph Isomorphism.

Monte Carlo Simulator

Repeat a sufficiently large number of times:

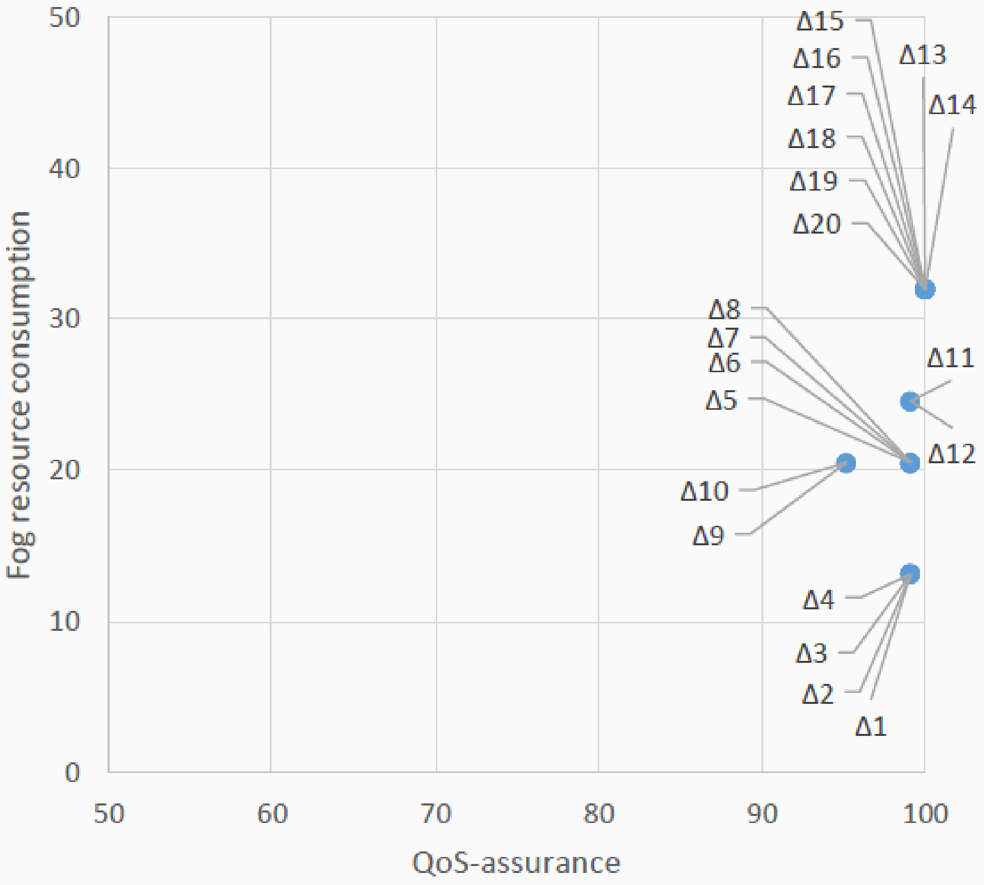
1. Sample a QoS profile for each link in the infrastructure.
2. Run backtracking algorithm.

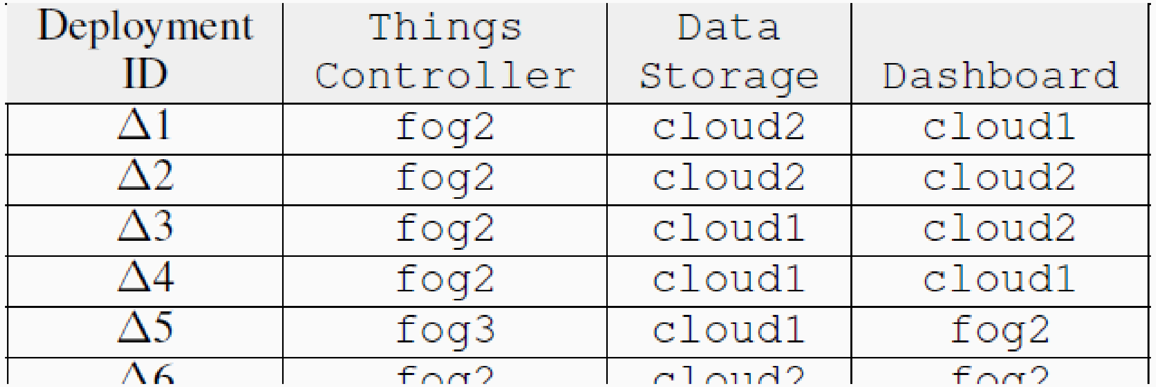
Compute statistics on generated deployment.



FogTorchΠ Results

Which are the eligible deployments that **comply most** with the required **QoS**?



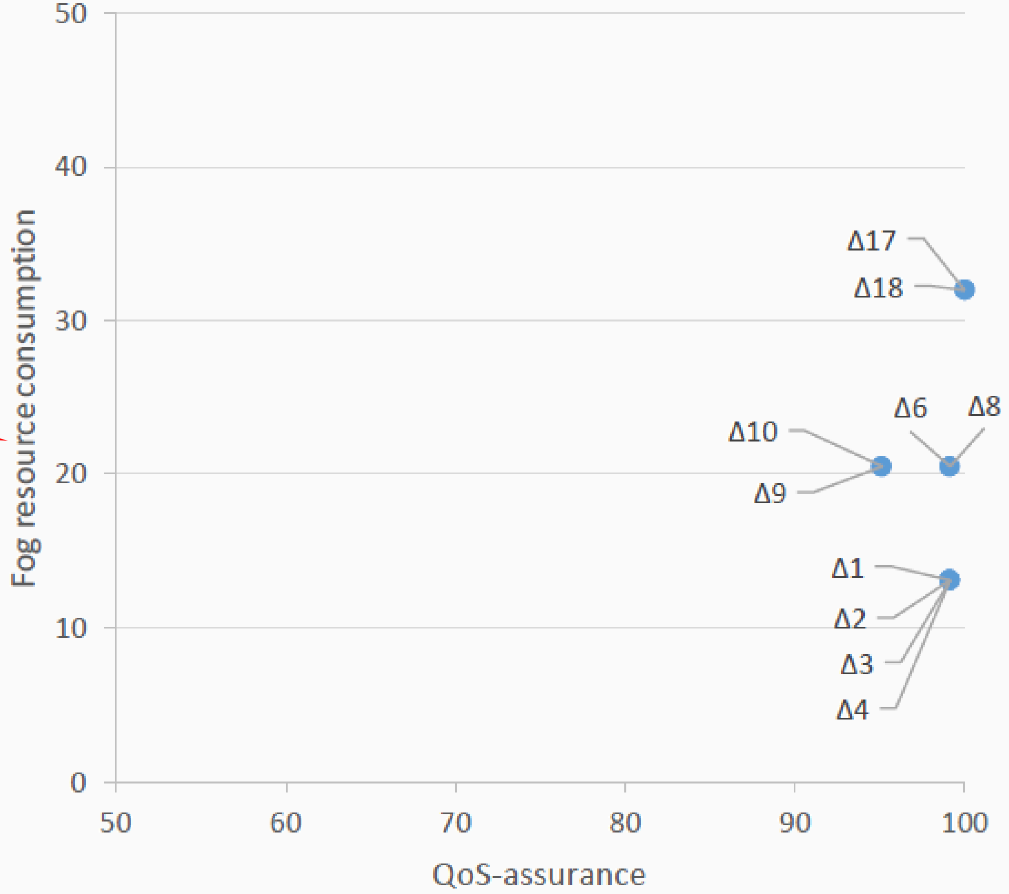


Deployment ID	Things Controller	Data Storage	Dashboard
Δ1	fog2	cloud2	cloud1
Δ2	fog2	cloud2	cloud2
Δ3	fog2	cloud1	cloud2
Δ4	fog2	cloud1	cloud1
Δ5	fog3	cloud1	fog2
Δ6	fog2	cloud2	fog2

FogTorchΠ Results (1)

Is it possible to **reduce** resource consumption of some Fog nodes, or **avoid** them?

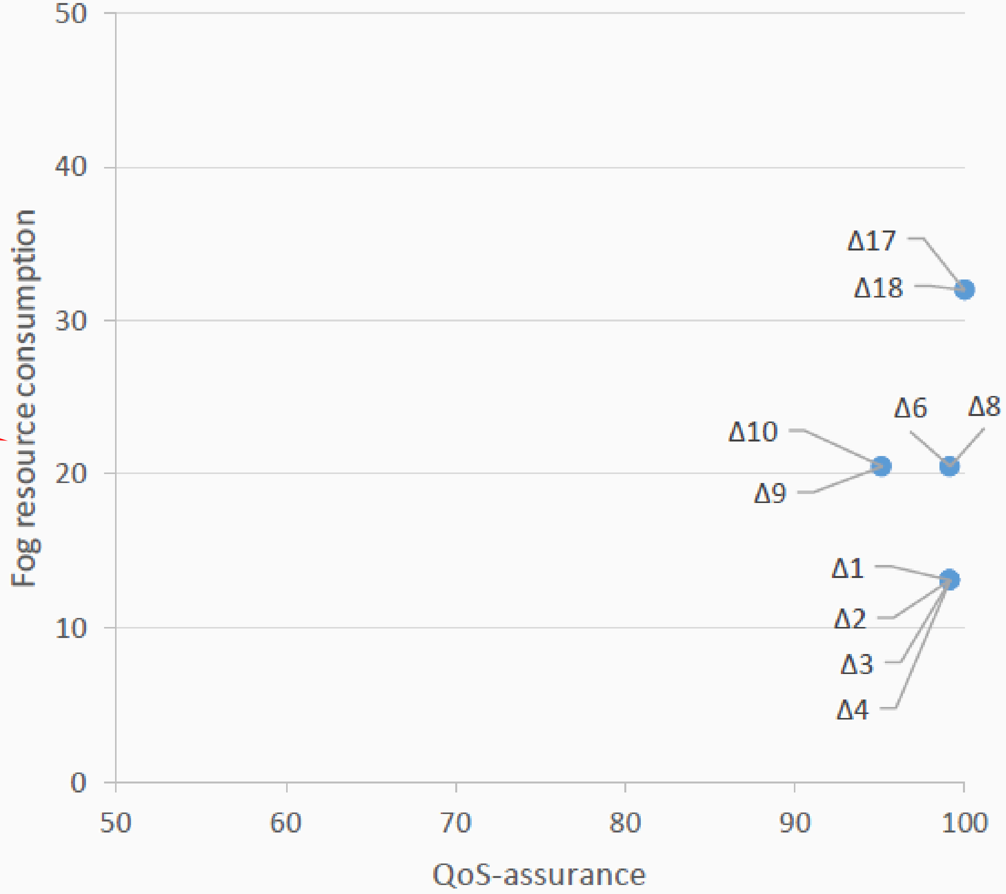
E.g., avoid using fog_3 for deployment.



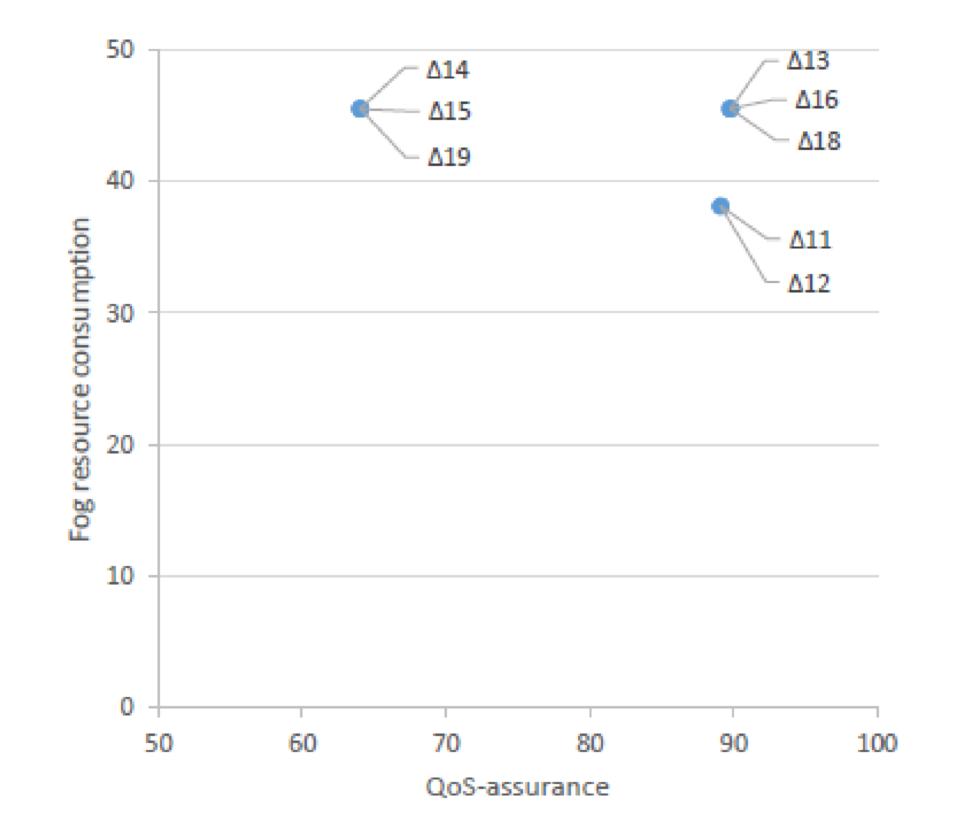
FogTorchΠ Results (1)

IS IT POSSIBLE TO **REDUCE** RESOURCE CONSUMPTION OF SOME FOG NODES, OR **AVOID** THEM?

E.g., avoid using fog_3 for deployment. ✓



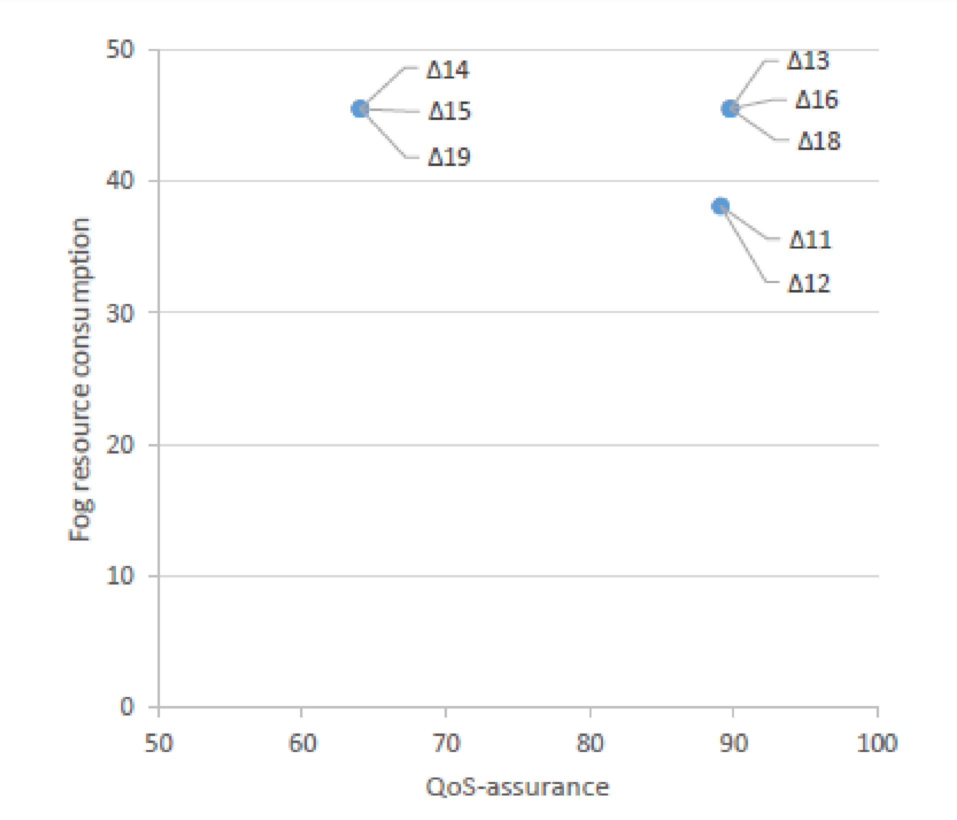
FogTorchΠ Results (2)



Do I have to upgrade my infrastructure if the application requirements change?

E.g., deploying HD video streaming without upgrade, leads to same QoS-assurance.

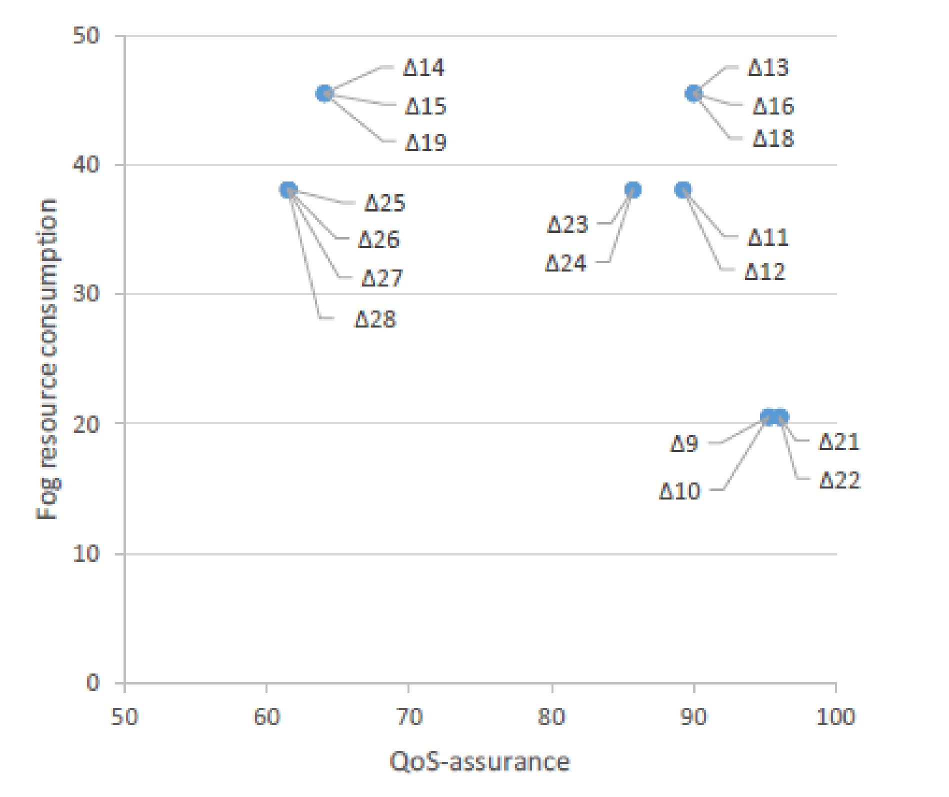
FogTorchΠ Results (2)



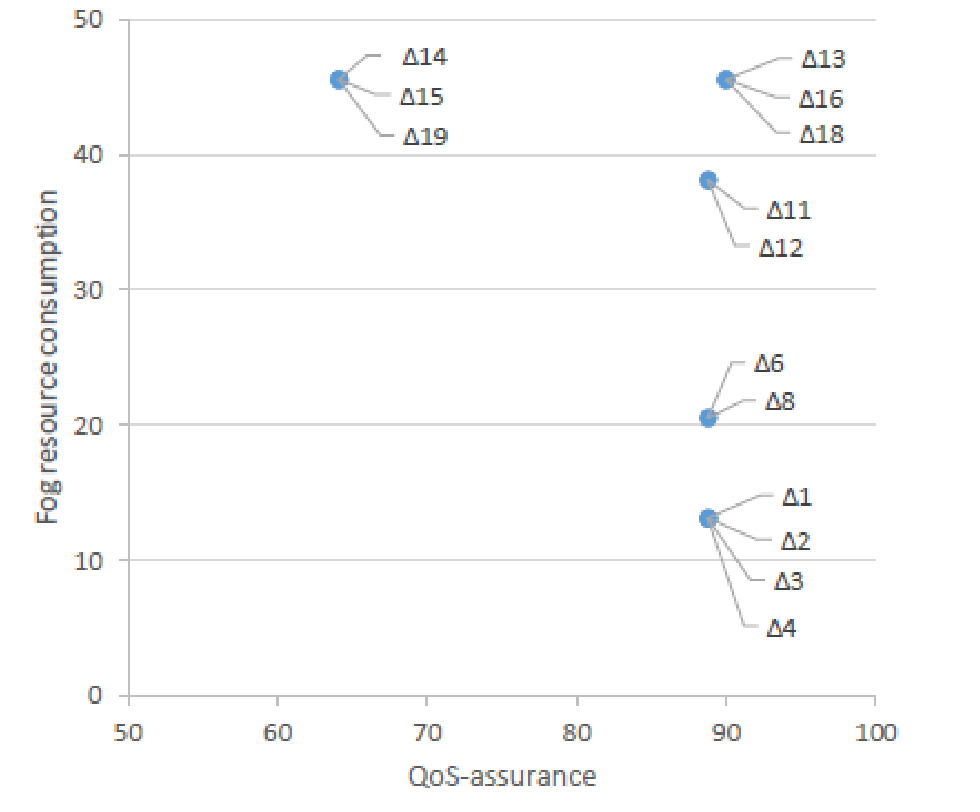
Do I have to upgrade my infrastructure if the application requirements change?

Deploying HD video streaming without upgrade, leads to worse QoS-assurance.

Results FogTorchΠ (3)

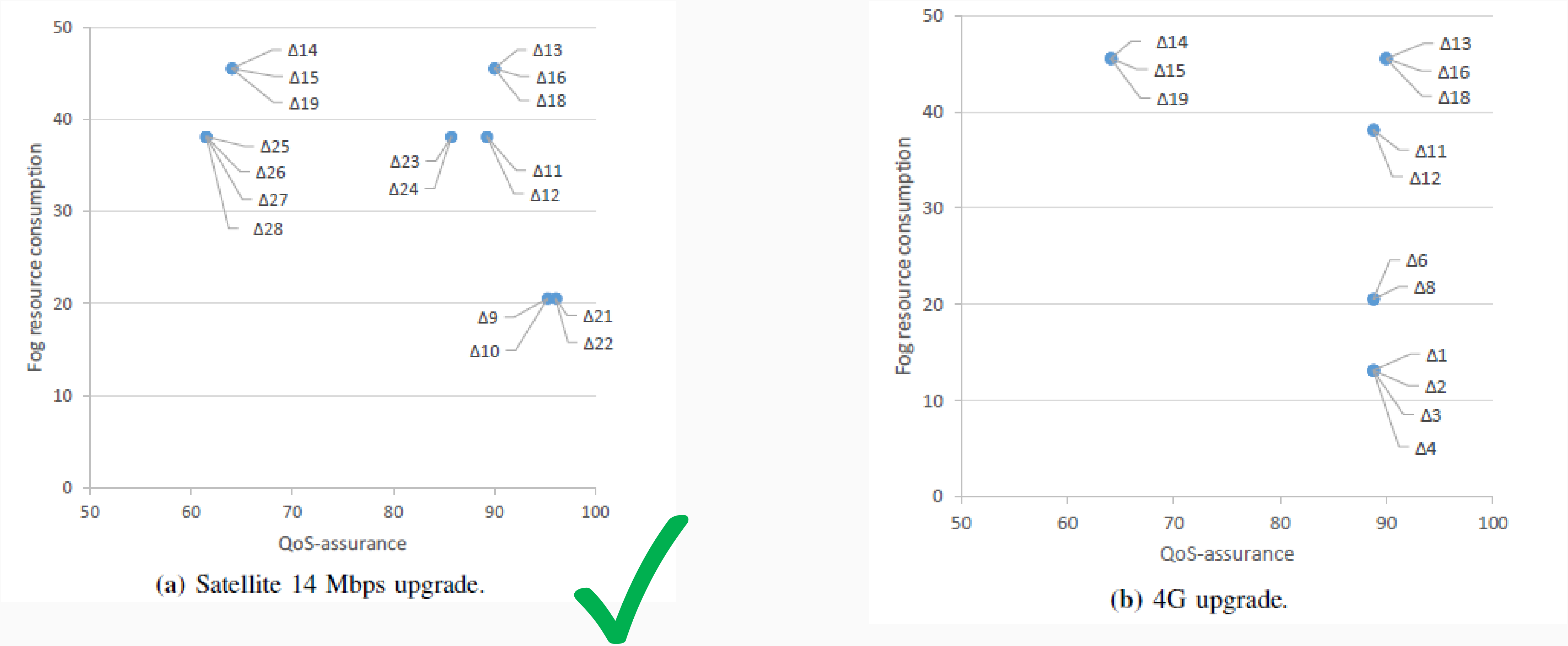


(a) Satellite 14 Mbps upgrade.



(b) 4G upgrade.

Results FogTorchΠ (3)



(a) Satellite 14 Mbps upgrade.

✓

(b) 4G upgrade.

Conclusions

- FogTorchΠ can **simulate and compare** different Fog scenarios at **design time**, determining **QoS-aware deployments** of Fog applications.
- It takes into account both **processing** (e.g., CPU, RAM, storage, software) and **QoS** (e.g., latency, bandwidth) constraints.
- It estimates **QoS-assurance** of deployments based on **probability distributions** of QoS featured by communication links.

Future Work

- Add new **QoS** attributes and include **cost** information.
- **Multiple** and multi-tenant **deployments** on the same infrastructure.
- Testing over real **case studies** and **heuristic** reduction of search space to permit **scalability.**

Thanks for your attention

Q&A