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

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

50 billion of connected devices by 2020

- Transportation
- Agriculture
- Building & Cities
- Hospitality
- Visual Security
- Wind Farms
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

- DataStorage
- Dashboard

ThingsController

- video
- water
- moisture
- fire
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

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

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

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

Fog Infrastructure
QoS Probabilities
Application
Deployment Policies
Things Binding

Monte Carlo simulator

Fog Infrastructure
Eligible deployments

 Eligible deployments

Fog resource consumption
QoS-assurance

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

• A QoS profile is a pair \( \langle \ell, \langle b_\downarrow, b_\uparrow \rangle \rangle \)

• They represent latency and bandwidth featured by a link or requested by a software interaction.
Application

• 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.

SD video

\[160 \text{ ms}, 0.5 \text{ Mbps}, 0.7 \text{ Mbps}\]
Infrastructure

98% (70 ms, 6 Mbps, 0.75 Mbps)
2% (70 ms, 0 Mbps, 0 Mbps)

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.
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*

*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.
FogTorch Results

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

<table>
<thead>
<tr>
<th>Deployment ID</th>
<th>Things Controller</th>
<th>Data Storage</th>
<th>Dashboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Δ1</td>
<td>fog2</td>
<td>cloud2</td>
<td>cloud1</td>
</tr>
<tr>
<td>Δ2</td>
<td>fog2</td>
<td>cloud2</td>
<td>cloud2</td>
</tr>
<tr>
<td>Δ3</td>
<td>fog2</td>
<td>cloud1</td>
<td>cloud2</td>
</tr>
<tr>
<td>Δ4</td>
<td>fog2</td>
<td>cloud1</td>
<td>cloud1</td>
</tr>
<tr>
<td>Δ5</td>
<td>fog3</td>
<td>cloud1</td>
<td>fog2</td>
</tr>
<tr>
<td>Δ6</td>
<td>fog2</td>
<td>cloud2</td>
<td>fog2</td>
</tr>
</tbody>
</table>
Is it possible to reduce resource consumption of some fog nodes, or avoid them?

E.g., avoid using fog_3 for deployment.
Is it possible to reduce resource consumption of some fog nodes, or avoid them?

E.g., avoid using fog_3 for deployment.
FogTorch 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.
Deploying HD video streaming without upgrade, leads to worse QoS-assurance.

Do I have to upgrade my infrastructure if the application requirements change?
Results FogTorch\(\Pi\) (3)

(a) Satellite 14 Mbps upgrade.

(b) 4G upgrade.
Results FogTorch (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