Fogify: A Fog Computing Emulation Framework

Moysis Symeonides*, Zacharias Georgiou*, Demetris Trihinas†,
George Pallis*, Marios D. Dikaiakos*

*Department of Computer Science
University of Cyprus
{msyme03, zgeorg03, gpallis, mdd}@cs.ucy.ac.cy

†Department of Computer Science
University of Nicosia
trihinas.d@unic.ac.cy
Fog Design and Deployment Challenges

**Fog Ecosystem Roles**

- IoT Service Developers
- Academic Researchers
- Fog Computing Operators

Costly device’s selection

- Infrastructure Realization
- Performance Analysis

Time-consuming configuration
- Monitoring
- Workload Alterations

Unpredictable faults
- Application Development
- Resource & network heterogeneity

Unpredictable faults
Fog Design and Deployment Challenges

**Fog Ecosystem Roles**
- IoT Service Developers
- Academic Researchers
- Fog Computing Operators

**Fog Design and Deployment Challenges**
- Costly device’s selection
- Time-consuming configuration
- Monitoring
- Workload Alterations
- Unpredictable faults
- Resource & network heterogeneity

*To this end, the users seek solutions to model and analyze the behavior of Fog infrastructure and IoT services…*
Emulators provide resource and network heterogeneity[1][2] but…

● usually, they do **not** provide **ad-hoc network changes** and **device failures**
● lack in applying fog-specific functions such as **node mobility**
● either do **not** provide **multi-host scalability** or are tightly-coupled with **specific cloud environments**[3]
● And *most importantly*, none of them supports...
  ○ **monitoring** of the emulated devices
  ○ **automated deployment**

---

[1] Fogbed: A rapid-prototyping emulation environment for fog computing, A. Coutinho et al. IEEE International Conference on Communications (ICC), 2018
Fogify: Fog Emulation Framework

- Resource Heterogeneity
- Network Link Heterogeneity
- Controllable Faults and Alterations
- Any-scale Experimentation
- Monitoring Capabilities
- Rapid Application Deployment
Fogify workflow

![Diagram of Fogify workflow]

- **Deployment**
  - Docker-compose file Application & Workload Generators
  - Fogify Modeling

- **Evaluation**
  - Actions & Scenarios

- **Analysis**
  - Graphs

**Client Side**

**Fogify SDK**

**Control Layer**

- **Fogify Controller**
  - API
  - Parser
  - Coordination
  - Orchestrator Connector

**Resource Management Layer**

- **Cluster Orchestrator**

**Execution Layer**

- **Fogify Agent**
  - Low-level Actions Module
  - Containers Listener
  - Monitoring Agent

**Overlay Networks**

- **Cluster Node**
  - Services
  - Workload Generators
  - Emulation

- **Cluster Node**
  - Services
  - Workload Generators
  - Emulation

- **Components of Docker Orchestrator Stack**
- **Components of Fogify Stack**
Fogify workflow
Fogify workflow
Fogify workflow
Fogify workflow
Fogify workflow
Modeling
Motivating Example

Use-case: A taxi company wants to analyse region-based data from its fleet.

In a real testbed, the company should:

- **Purchase** MECs and taxis sensors
- **Configure** the MECs and **place** them at their physical location
- **Setup** the network components
- **Monitor** the “health” of the infrastructure

At the end, the developers will **not be sure if they tackle every single obstacle of network changes and device’s failures.**
Use-case Modeling in Fogify

For the Fog Nodes:

- **5 MECs** (4 cores@1.4GHz, 4GB RAM) are placed in 5 different regions (region-{1-5})
- **Taxis/car-node** (1 core@700MHz, 256MB RAM) sending sensed data to nearby MEC
- **Cloud server** (8 cores@2.4GHz, 8GB RAM) computes the final results.

According to the **Network QoS**:

- **Regional Network**: 15ms latency and up to 10Mbps bandwidth
- **Edge-Cloud connection**: 100ms latency and up to 5Mbps bandwidth

---

services: ......
x-fogify: ......
nodes: ......
networks: ......
topology:
- label: mec-node-1
  service: mec-svc
  node: mec-node
networks:
- name: mec-net-1
  links:
  - car-node-at-mec-1:
    downlink: {
      latency: 50ms
    }
  - name: mec-2-cloud-net
  - name: mec-2-mec-net
  replicas: 1
  - label: mec-node-2
  service: .....
The initial Fog Templates of Fogify consist of:

- a set of Services,
- a set of Nodes,
- a set of Networks

A Fog Topology consists of Blueprints.

- A Blueprint is a combination of a Node, Service, set of Networks, replicas and a label.

Blueprints support the overriding of network-level QoS and specific links between Fog Nodes.
The initial **Fog Templates** of Fogify consist of:

- a set of *Services*,
- a set of *Nodes*,
- a set of *Networks*

A **Fog Topology** consists of Blueprints.

- A **Blueprint** is a combination of a *Node*, *Service*, set of *Networks*, *replicas* and a *label*.

Blueprints support the **overriding** of network-level QoS and specific links between Fog Nodes.
The initial **Fog Templates** of Fogify consist of:

- a set of *Services*,
- a set of *Nodes*,
- a set of *Networks*

A **Fog Topology** consists of Blueprints.

- A **Blueprint** is a combination of a *Node*, *Service*, set of *Networks*, *replicas* and a *label*.

Blueprints support the **overriding** of *network-level QoS* and *specific links* between Fog Nodes.
The initial **Fog Templates** of Fogify consist of:

- a set of *Services*,
- a set of *Nodes*,
- a set of *Networks*

**A Fog Topology** consists of **Blueprints**.

- A **Blueprint** is a combination of a *Node*, *Service*, set of *Networks*, *replicas* and a *label*.

Blueprints support the **overriding** of **network-level QoS** and **specific links** between Fog Nodes.
A **Fog Topology** consists of Blueprints.

- A **Blueprint** is a combination of a *Node*, *Service*, set of *Networks*, replicas and a *label*.

**Blueprints support the overriding of network-level QoS and specific links between Fog Nodes.**

The initial **Fog Templates** of Fogify consist of:

- a set of *Services*,
- a set of *Nodes*,
- a set of *Networks*
A **Fog Topology** consists of Blueprints.

- A **Blueprint** is a combination of a *Node*, *Service*, set of *Networks*, replicas and a *label*.

Blueprints support the **overriding** of *network-level QoS* and **specific links** between Fog Nodes.
Runtime Evaluation Model

The evaluation model consists of:

- **Actions** that change properties of a running Fog Topology. Actions can be:
  - *Scaling Actions* (horizontal or vertical)
  - *Network Actions*
  - *Stress Actions*

- **Scenario** is a *sequence of time scheduled actions* that Fogify will execute to emulate more complex user-driven experiments.

Scenario:
- label: Car-1
timestamp: 20s
type: network
parameters:
  - network: MEC-net-1
downlink:
    latency: 50ms
- label: Car-1
timestamp: 40s
type: network
parameters:
  - network: MEC-net-1
downlink:
    latency: 100ms
- label: Car-2
timestamp: 40s
type: scale-in
parameters:
  - num-of-instances: 1

- MEC-net-1 default connection
- Connection with 50ms network latency
- Connection with 100ms network latency
The evaluation model consists of:

- **Actions** that change properties of a running Fog Topology. Actions can be:
  - *Scaling Actions* (horizontal or vertical)
  - *Network Actions*
  - *Stress Actions*

- **Scenario** is a sequence of time scheduled actions that Fogify will execute to emulate more complex user-driven experiments.
Runtime Evaluation Model

The evaluation model consists of:

- **Actions** that change properties of a running Fog Topology. Actions can be:
  - *Scaling Actions* (horizontal or vertical)
  - *Network Actions*
  - *Stress Actions*

- **Scenario** is a *sequence of time scheduled actions* that Fogify will execute to emulate more complex user-driven experiments.
Runtime Evaluation Model

The evaluation model consists of:

- **Actions** that change properties of a running Fog Topology. Actions can be:
  - *Scaling Actions* (horizontal or vertical)
  - *Network Actions*
  - *Stress Actions*

- **Scenario** is a sequence of time scheduled actions that Fogify will execute to emulate more complex user-driven experiments.
System Details
Computational Resources

Fogify utilizes the **containers** as execution platform while they achieve *isolation*, through linux namespaces, and resource constraining by using Linux Control Groups (*cgroups*). Fogify maps the high-level device’s template to the host’s capabilities.

*For CPU cgroups give us only the ability to set a portion of host CPU. For that:*

- Fogify computes the **cumulative clock rate (CCR)** for both host and model as:
  \[
  CCR = \text{Cores} \times \text{Cycles}
  \]

- Then, it specifies the rate of the processing power that the emulated device will occupy from host:
  \[
  CPU_{rate} = \frac{emu\_node_{CCR}}{host_{CCR}}
  \]
Network Shaping

- Fogify realizes a virtual overlay mesh network (*Virtual Extensible Lan (VXLAN)*) per network description
- Then, Fogify agents apply the **network QoS**:  
  - Separate the *incoming and outcoming traffic*
  - Build a *tree-base structure* by utilizing Classful Queuing Disciplines (qdisc)
  - Traffic from root, follows the *filtering (based on containers’ IPs)* and reaches the leaves
  - Each leaf is represented by a *QoS Queue*, that is created by the *tc-tool*[5]

Monitoring System

A Monitoring Agent retrieves:

- *Basic Metrics* (CPU, RAM, network, disk)
- *Application's metrics*

Fogify SDK

FogifySDK is a *python library* that provides:

- *Deployment* of Topologies, Actions & scenarios
- Retrieval of the Monitoring data in a *High-performance data structure*
- Out-of-the-box *exploratory analysis*
- Connectivity with tools like *jupyter*

Evaluation
Scenario - Network Uncertainties

![Graphs showing network uncertainties](image-url)
Scenario - Network Uncertainties

We can observe the minimum execution utilization of all services and Fog nodes.

Common execution of the infrastructure.
Scenario - Network Uncertainties

2s network delay at MEC region-1 for 3 minutes

We identify a throughput degradation

Network traffic increases to the previous level, since the latency does not exceed the pre-defined 10s disconnect limit
Scenario - Network Uncertainties

A data transfer spike is appeared for the MEC and Car nodes due to queued requests.

Common execution for 3 minutes
**Scenario - Network Uncertainties**

When the network latency is boosted to 20s, every Car node request to the MEC fails.

The CPU and network effects of directly receiving requests from the taxis operating in region-1.

20s network delay at MEC region-1 for 3 minutes.
Scenario - Network Uncertainties

Common execution for 3 minutes

Cloud.cpu

Cloud.internet

MEC.region-1.cpu

MEC.region-1.local-net

MEC.region-1.internet

Car.region-1.cpu

Car.region-1.local-net

Car.region-1.internet
Scenario - Network Uncertainties

We observe a similar increase in the cloud CPU load and network traffic as in the previous period (75-110th interval)
In conclusion, with **network alterations** and **fault injections**, users:

- evaluate the execution of their services under *extreme conditions*
- identify *unpredictable outcomes* of imposed uncertainties to the service behavior
Other Scenarios

**Application-level metrics:** Operators can employ Fogify to produce and evaluate analytic insights, implementing adequate app-level metrics.

**Node profiling:** insights are highly beneficial to engineers for capacity planning, optimizing service and resource placement.

**Scaling Actions & Workload Changes:** released insights about service performance and resources utilization.
Emulation Accuracy

The emulated computing resources have only a small performance degradation for workloads approaching 100% CPU usage.

Fogify achieves near to real-world network link capabilities, with only outliers not captured and a slight overhead in low-latency connections.

The emulation results closely follow the real measurements with a 5%-8% deviation of the overall experiment time.
Conclusion

● We introduced **Fogify**: an open-source & all-in-one Fog emulator.
  ○ features a powerful model specification for Fog Deployments,
  ○ allows the controllable **Faults** and **Alterations**,  
  ○ offers enablers for **Any-scale Experimentation**, and 
  ○ provides **Monitoring Capabilities**

● We provided a detailed description of the implementation aspects, such as **resource management**, **network shaping** and **monitoring**.

● The **evaluation** illustrates:
  ○ We introduced **real-world IoT applications** that demonstrate the **usability**, **extensibility**, and **observability** of Fogify.
  ○ The **error displayed 3-8% deviation** between emulated and real Infrastructures.

● Our **future work** includes an extensive **scalability evaluation**, implementation of **new actions** and the evaluation of more **real-world scenarios**
Thank You

You are welcome to our live demonstration (Demo: Emulating Geo-Distributed Fog Services, Thursday, Nov 12, Poster Session II) and you will find the code of the Fogify at the following link https://github.com/UCY-LINC-LAB/fogify

Documentation: https://ucy-linc-lab.github.io/fogify/
Demo: https://github.com/UCY-LINC-LAB/fogify-demo
Video: https://www.youtube.com/watch?v=PthMM6rC89o