Feasibility of Fog Computing Deployment based on Docker Containerization over RaspberryPi

Paolo Bellavista, Alessandro Zanni Dept. Computer Science and Engineering (DISI) Alma Mater Studiorum - University of Bologna

ICDCN '17, January 04-07, 2017, Hyderabad, India

Introduction

- Two layers cloud-centric architectures (sensors/actuators and the cloud) are inadequate in IoT domains
- Additional gateway nodes relatively local to sensors/actuators, can significant enrich the flexibility
- Innovative fog computing solution
  - Scalability extensions of the IoT gateway provided by the open-source Kura framework
  - Docker-based containerization over resource-limited Raspberry Pi

The Kura Framework

- Kura aims at offering a Java/OSGi-based container for M2M applications running in service gateways
- Uses MQTT as its central protocol

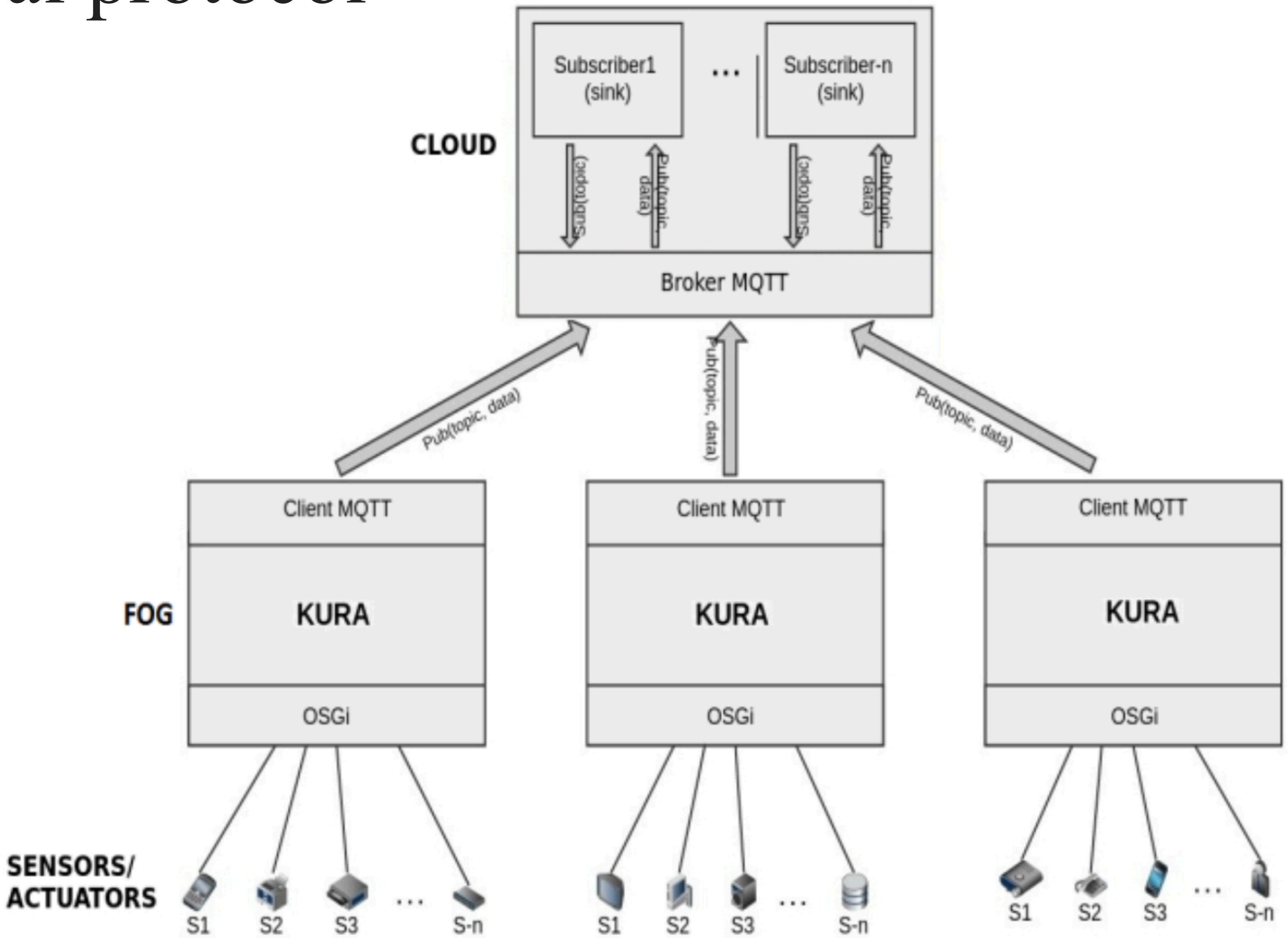

Figure 1. A Cloud-integrated Architecture based on Kura

Weaknesses of Kura

- Single MQTT broker on the cloud
  - Performance slowdown in case of high load
  - No fog-oriented processing operation performed locally
  - Persistent sockets produce waste of resources
- Flat topology
  - Gateways organized in a flat topology can perform only relatively limited operations

Gateway-side MQTT Brokers

- Enabling hierarchical topologies
- Gateway-level MQTT message aggregation
- Real-time message delivery and reactions

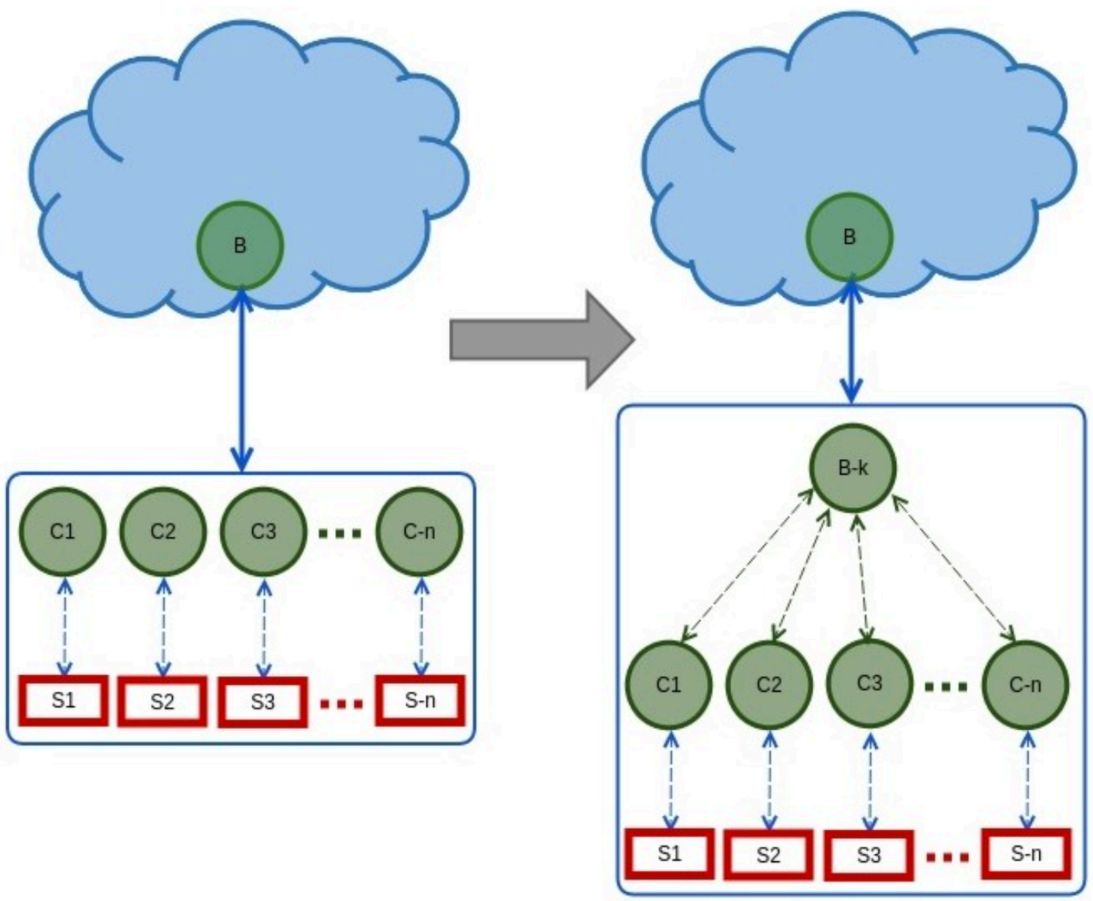

Figure 2. Adding Gateway-side MQTT brokers

Gateway-side MQTT Brokers

- Actuation capacity and message priorities
  - Determine the situations when it is necessary an immediate actuation or not
- Locality awareness and locality-oriented optimization
- Gateway-cloud connection optimization
  - Dynamically established only when necessary

Enabling Cluster/Mesh Topologies for Kura Gateways

- Combine multiple physical gateways and aggregate of their resources

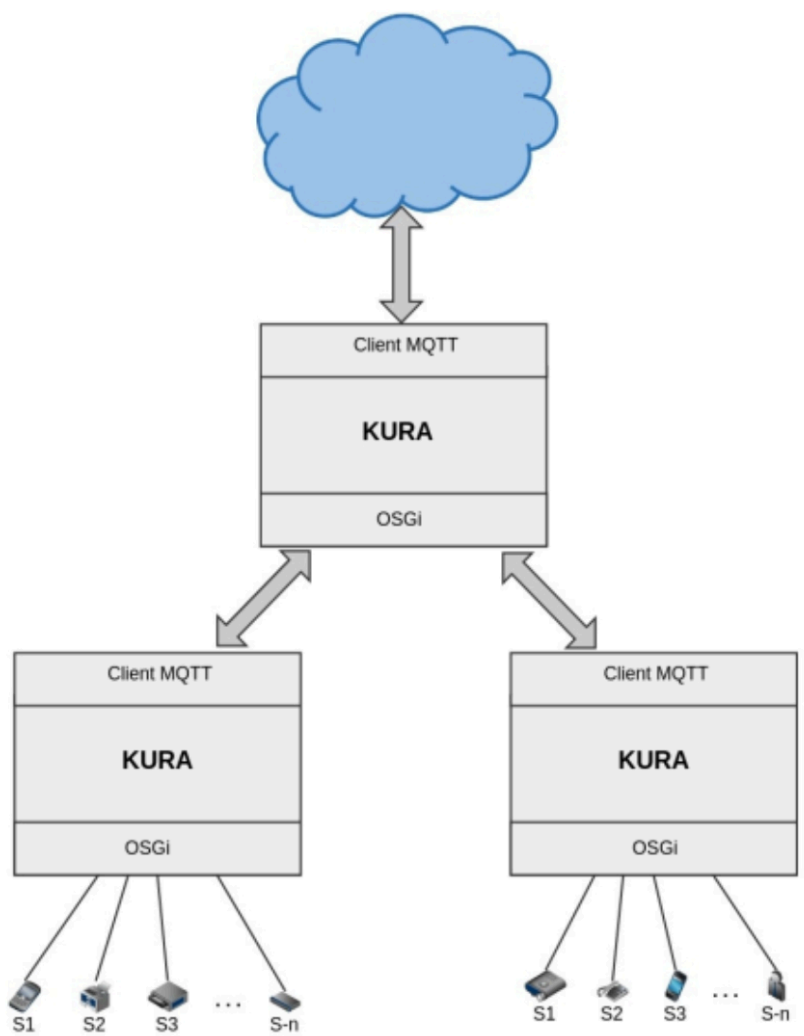

Figure 3. The Supported Cluster Organization of IoT Gateways

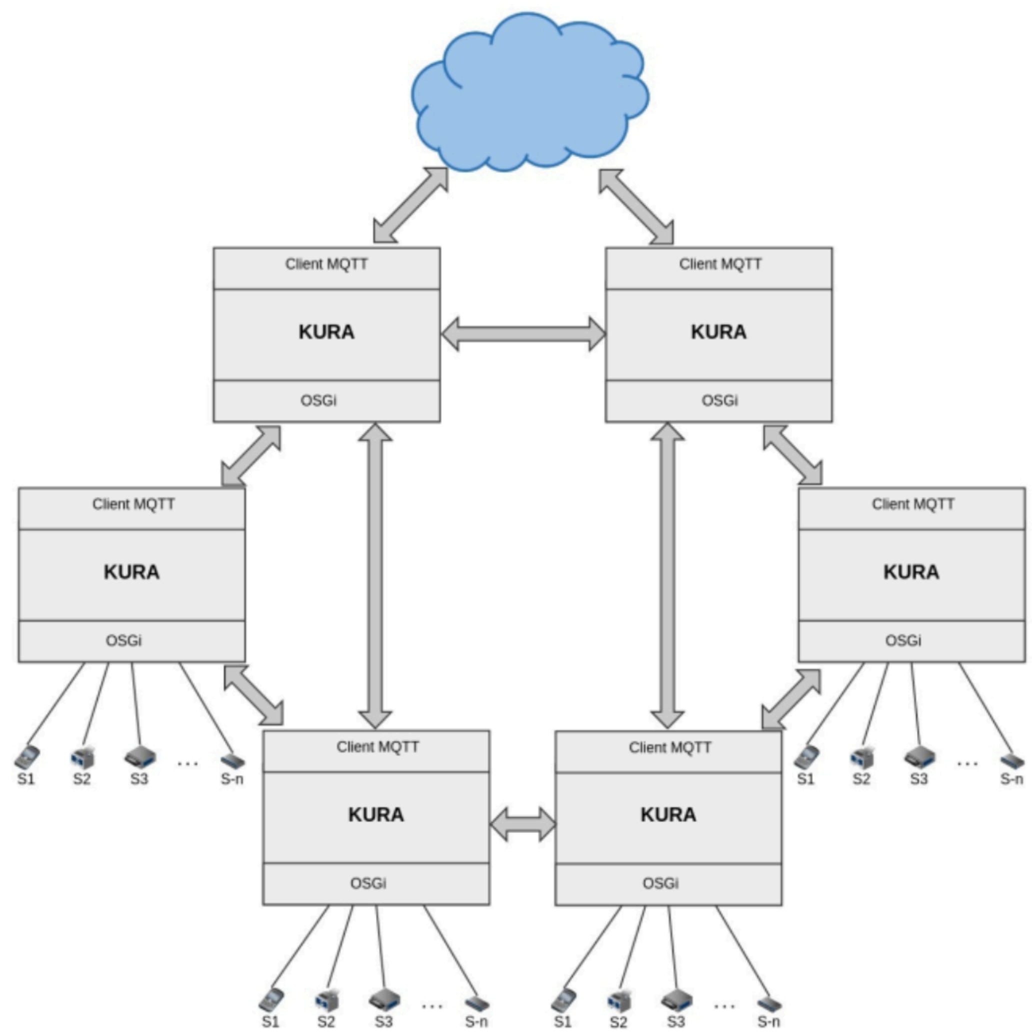

Figure 4. The Supported Mesh Organization of IoT Gateways

Enabling Cluster/Mesh Topologies for Kura Gateways

- Kura gateway specialization
  - Some gateways are more suitable to perform some tasks
- Locality exploitation and data quality
  - Performing more accurate and complex analytics
- Geo-distribution
  - Manage dense sensor localities and to make the overall distributed deployment scale better
- Scalability
- Security and privacy

Configuration and Management of IoT Gateways

- Gateway standard base configuration
  - They define a standard gateway configuration
  - Every fog node has the same base configuration and the same skeleton Container-based services

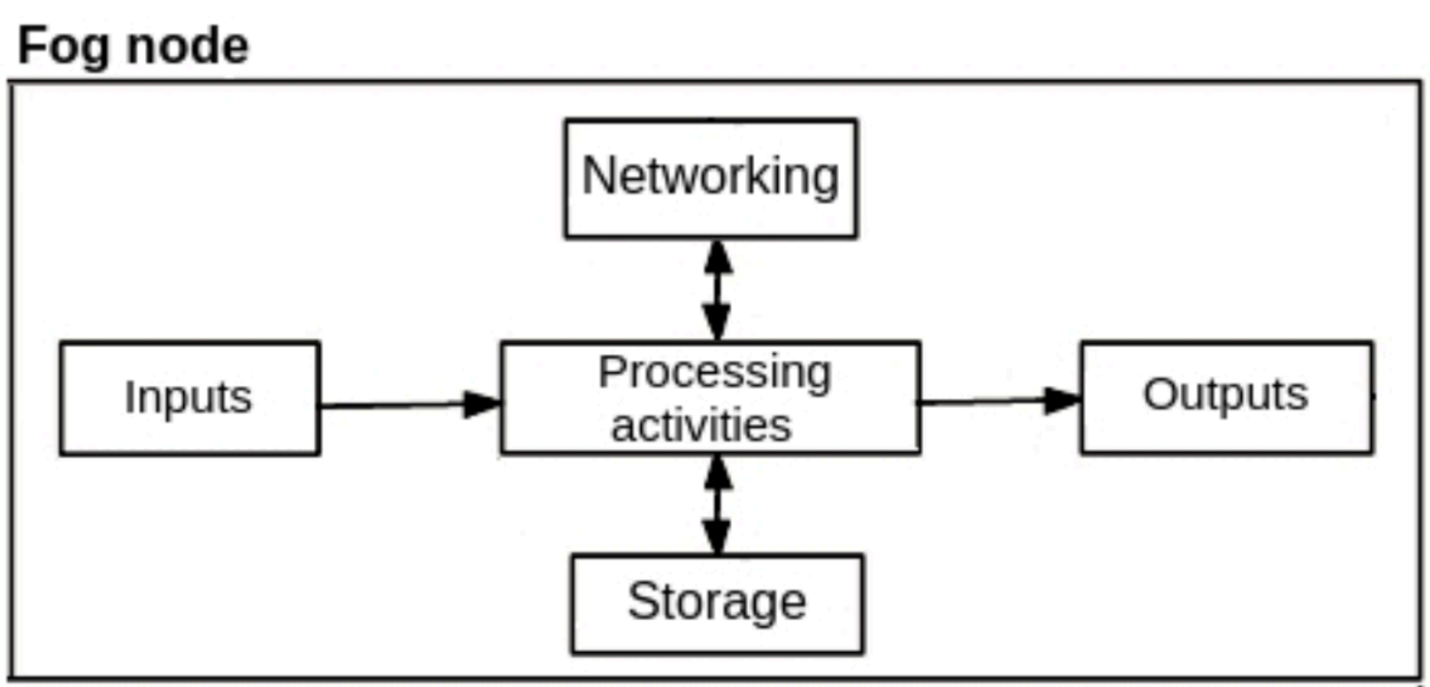

Figure 5. Fog node skeleton

Configuration and Management of IoT Gateways

- Container-based services
- Management and orchestration
  - Docker Swarm
  - Kubernets
  - Apache Mesos

Filesystem Selection and Impact

- AUFS
  - Layered filesystem
  - Copy-on-Write (CoW)
- Device-mapper
  - Block level
- OverlayFS
  - Two main layers

Containerization Overhead Performance Results

- Smart Connected Vehicles (SVC)
  - Fog node acts as a mobile sink collecting data from a dynamically determined set of heterogeneous sensors
  - Mobile sink can decide to spread valuable concise information to other SVC participants

Containerization Overhead Performance Results

Operation category	native	Docker + AUFS	Docker + Device mapper	Docker + OverlayFS
Start Container	-	3.5 s	9.1 s	3.3 s
I/O Operations	1.6 s	4.3 s	4.7 s	4.3 s
CPU Operations	3.1 s	3.4 s	4.2 s	3.5 s
Total Execution	4.7 s	12.5 s	21 s	11.8 s

Table 1. Native-code and container execution time

Containerization Overhead Performance Results

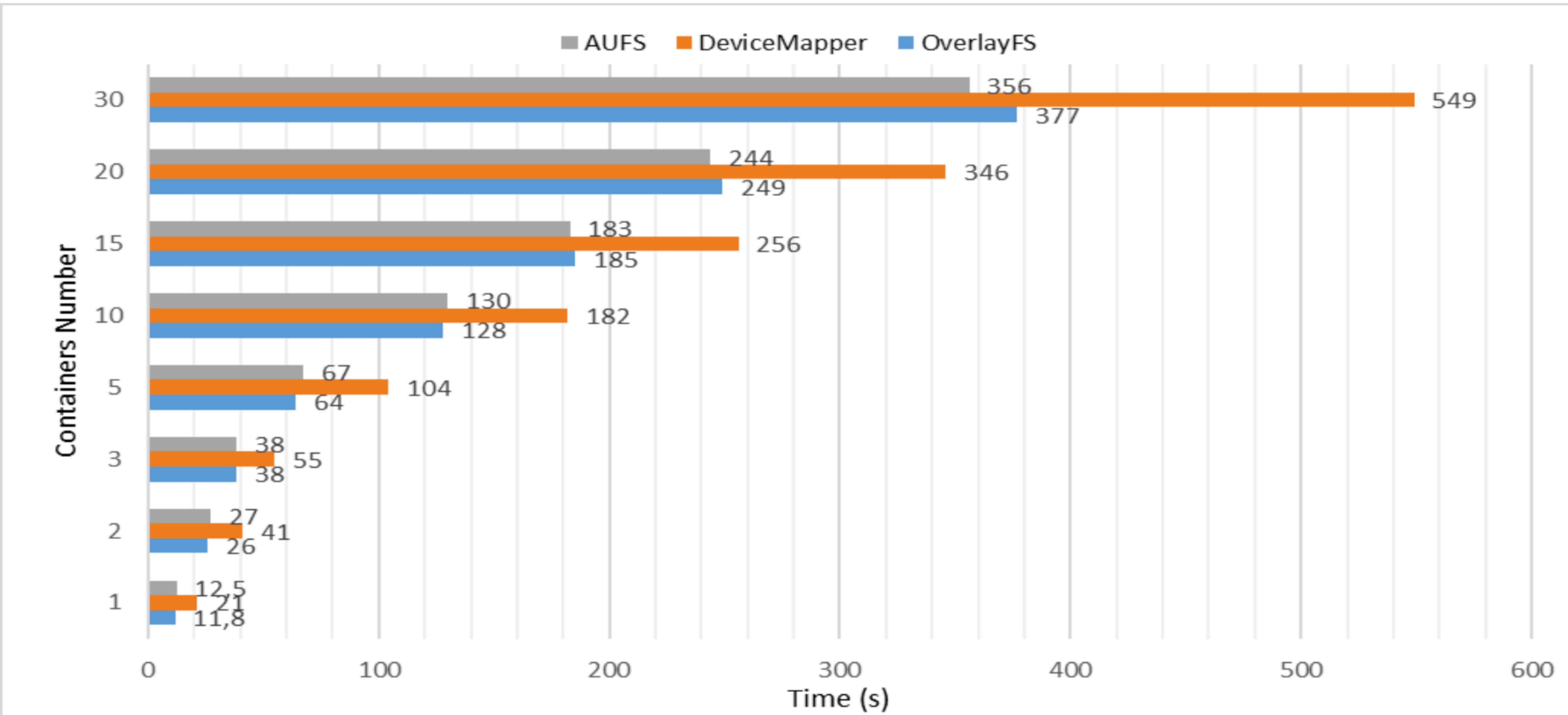

Figure 6. SVC Execution Time over Multiple Containers

Conclusion

- Address how to build a real fog middleware support by IoT gateways along two directions
  - Decentralizing the MQTT broker functionality of the Kura framework from the cloud to the involved edges
  - Exploiting containerization to facilitate interoperability and portability via node configuration standardization