A Framework based on SDN and Containers for Dynamic Service Chains on IoT Gateways

Morabito, Roberto, and Nicklas Beijar,
Introduction

- To enable efficient network architectures for the Internet of Things (IoT), they integrate **Software Defined Networking (SDN)** and **Container Virtualization** for coping with the complex requirements in networking.

- This paper proposed a framework aim for application deployment, and scalability in IoT scenarios.
Software-defined networking

- Features of SDN:
  - Decouple control plane and data plane
  - Centralized controller
- SDN enables dynamic reconfiguration of routing.
Container Virtualization

- Containers offer application packaging, deployment method, and application isolation with low overhead.
  - can be more efficient than VMs.
Framework Design

- Architectures: IoT Devices, Capillary Gateway, Data Center
- IoT data process in **Service Function Chain**.
Service Function Chaining

- Services implemented as independent components, interact with each other through specific APIs.
  - In IoT, the software often takes the form of a chain.
- Allow dynamic scaling and reconfiguration.

![Diagram showing the process of service function chaining with various components such as physical devices, virtual device drivers, and data centers.](image-url)
Design Features

- **Chains for IoT data processing**
  - Instantiate a separate chain for each IoT device.

- **Edge computation**
  - Using container virtualization to run separated services.

- **Seamless network reconfiguration**
  - Using SDN for remaining connection, when re-deploying the chain.
  - Using *pseudo addresses* to denote the preceding and the succeeding component in the chain.

- **Isolation**
  - Allows gateway to be shared between several tenants.
Implementation

- The data center is in Lund (Sweden), 850 km from the test site (Jorvas, Finland).
- The gateway is a RaspberryPi 3 with 1GB of memory and a 4 core CPU.
  - Running hostapd to create 4 SSID for IoT devices.
- Both data center and gateway run a Open vSwitch, which is managed by the SDN controller.
- Service components are running in containers on Docker.
  - Using their own Docker Network Driver Plugin, allow container connecting to the virtual switch.
Implementation (cont.)
Evaluation Goals

- Demonstrate the efficient deployment on low-power devices.
- Understand the capabilities of the gateway scalability.
- Evaluate the impact of the different software components used to build the platform.
Experiments Setting

● The IoT device generates video stream, processed locally on the gateway.
  ○ Implemented as a RPi connected to an USB camera.
  ○ Using VLC to capture the stream, encode it to MPEG-4, and serve it over HTTP.

● The processing in the data center is omitted, and consists of a simple streaming server to which clients can connect to.
Experiments

- Evaluate the performance, and increase the number of containers.

*Figure 3: Gateway’s network interfaces setup.*
Performance Evaluations

(a) CPU Usage (%)

(b) Power Consumption (Watts)

(c) System Load

- AVG 1-min
- MAX 1-min
- AVG 5-min
- MAX 5-min

- #1 Container
- #2 Container
- #3 Container
- #4 Container
- #5 Container
Software Components Evaluations

(a) CPU Usage (%) for different components:
- OVS others
- OVS vswitchd
- OVS db-server
- OpenVPN
- Hostapd

(b) CPU Usage (%) for different components:
- Motion Detection
- Recolor
- Grey Scale
- Invert
- Driver

(c) RAM Memory Usage (%) for different components:
- Motion Detection
- Recolor
- Grey Scale
- Invert
- Driver
- OVS vswitchd
- OVS db-server
- OpenVPN
- Hostapd
Conclusion

- This paper present a framework that integrate SDN and Docker for constrained IoT scenarios, which can build service function chains in a novel way.
- Experiments show that the approach can deploy efficiently even on low computational resources devices.
- For future work, they intend to explore the framework performance on different devices, and the number of tenants that can share the same gateway.