IoTNetEMU - A framework to emulate and test IoT applications

João Oliveira joao.oliveira@fraunhofer.pt Fraunhofer Portugal AICOS Porto, Portugal Filipe Sousa filipe.sousa@fraunhofer.pt Fraunhofer Portugal AICOS Porto, Portugal Luís Almeida Ida@fe.up.pt CISTER / FEUP - University of Porto Porto, Portugal

ABSTRACT

Testing and validating IoT applications is inherently complex due to the number of devices involved that should communicate with each other and with cloud infrastructures. Models, emulators and testbeds are used to validate parts of the solution, but the gap between simulations and live experiences is still deep. To minimize this gap, we propose IoTNetEMU, a modular framework that combines several open-source components to provide a validation tool for developers and researchers. This demonstration shows how to test and validate an IoT application with minimal changes using the framework.

1 INTRODUCTION

The Internet of Things (IoT) is a current paradigm where several interconnected devices are used to collect and process data from our environment and actuate in many use cases of our daily lives. The vision is for these devices to become ubiquitous and scale to massive numbers, supported by smart edge devices and cloud infrastructures. However, the inherent complexities and challenges of testing this type of applications pose significant hurdles for developers and researchers.

Some solutions are proposed in the literature to tame this complexity. One approach is to model nodes and applications to validate algorithms or policies. The CloudSim framework [5] is one solution that focuses on modelling Cloud deployments. Several related works were proposed focusing on specific applications such as iFogSim [6], which focuses on Fog scenarios, or EdgeCloudSim [8], which targets Edge-based scenarios. Nevertheless, these solutions are limited for validating IoT applications since not all components run actual code. Other solutions follow the emulation approach to reduce the gap to real deployments. One such solution is the Cooja Network Simulator [1], a tool for Contiki-based applications. This tool allows developers to run Contiki code in motes connected to a simulated network. Even though this solution is much closer to a real scenario it is still limited since it only supports the Contiki OS. Renode [3] is another proposal that allows developers to run actual code in an emulated device regardless of the operating system. However, it requires that the hardware models of the desired target are implemented and supported. Furthermore, the network model is limited, reducing the applications that can be used. Another approach is to use testbeds with real devices to validate the developed application. Since deploying a significant number of devices might not be feasible, a community testbed such as Fit IoT-Lab [4] can be used. Nonetheless, using a remote testbed is not always practical since it might not be available when required.

Even though several approaches are proposed, one is lacking that allows developers to emulate a significant number of nodes running real code that can then interconnect with application services in a controllable network. In this demonstration, we present the IoTNetEMU framework and how it can be used to test and validate generic IoT scenarios involving constrained nodes and containerized applications. It combines previous approaches, such as emulation and network simulation, to reduce the difference between testing applications using modelling or testbeds with real devices. The rest of this paper describes the framework and how it can be used to validate a simple IoT application.

2 IoTNetEMU DESCRIPTION

IoTNetEMU is a Python framework that integrates several opensource emulation and simulation tools to create a more realistic testing and development scenario for IoT applications. It allows developers to easily validate and test an application using simple configurations and actual code with minimal adaptations.

The framework divides the entities of an IoT application into three main abstract components:

- Node: defines IoT nodes that connect to Networks and communicate with Applications. Such Nodes are end-nodes, edge devices or even gateways and are used to emulate a complete device running real code.
- Application: define IoT applications deployed in a gateway or cloud server. This component emulates containerized software components independent of the underlying hardware.
- **Network:** represents the networks that connect Nodes to Nodes, Nodes to Applications or Applications to Applications. Multiple Networks can exist in an IoT application scenario.

Currently, IoTNetEMU features at least one implementation for each component. Since our focus is on networks of constrained devices, the current Node implementation uses QEMU [2] to emulate a microcontroller (Cortex-M3) running a Zephyr application. We leverage QEMU's virtual network interface and Zephyr's mps2_an385 board definition to use a TAP to connect the emulated device to the Network components and provide connectivity to the firmware. The current Application implementation is focused on running software components on Docker containers. It creates a network for the applications using a dedicated bridge and controls the container lifecycle. A TAP interface is attached to the bridge that is then used to connect the containers to the Network components. The basic Network implementation simply connects all TAP interfaces from the Nodes to the bridge of the Docker containers and sets static IPs according to the configuration. Naturally, this allows all Nodes to communicate with the Applications and between themselves but it is limited since it cannot be used to emulate realistic network scenarios. The second implementation integrates the ns-3 network simulator [7]. It uses the TapBridge model to connect the Nodes and Applications to the ns-3 simulation and allows the

definition of custom network topologies using the ns-3 wired and wireless models.

The IoTNetEMU architecture and components allow users to emulate devices running actual code with minor modifications and connect it to containerized applications through a fully controllable network to test and validate the envisioned application. This eases the validation effort of such solutions and reduces the gap between simulations and live experiences. Although the current number of implementations is not extensive, the modular architecture enables the integration of other tools to expand its capabilities further, if needed.

3 TESTING AN IOT APPLICATION

For this demonstration, we detail the steps to test and validate a simple application where multiple nodes communicate with an LwM2M server running in a cloud. We use the *lwm2m_client* sample application available in Zephyr and the Leshan LwM2M Server.

```
nodes :
  node1:
    type: 'zephyr'
    image: '/workspaces/zephyr/samples/net/lwm2m_client'
   params:
      - '-DCONFIG_BOOTSTRAP_ID=\"node1\"'
      - '-DCONFIG_NET_CONFIG_PEER_IPV4_ADDR=\"10.10.10.253\"'
      - '-DCONFIG_LWM2M_PEER_PORT=5685'
    networks
      ns3:
        ipv4_address: 10.10.10.1
        ipv4 gateway: 10.10.10.254
  node2:
    type: 'zephyr
    image: '/workspaces/zephyr/samples/net/lwm2m_client'
    params:
        '-DCONFIG_BOOTSTRAP_ID=\"node2\"'
      - '-DCONFIG_NET_CONFIG_PEER_IPV4_ADDR=\"10.10.10.253\"'
      - '-DCONFIG_LWM2M_PEER_PORT=5685'
    networks:
        ipv4 address: 10.10.10.2
        ipv4_gateway: 10.10.10.254
  node3:
    (...)
  node4:
    (...)
 node5:
    (...)
applications:
  leshanserver
    type: 'docker'
image: 'iotnetemu/leshan-server:1.0.0'
    networks:
      ns3:
        ipv4_address: 10.10.10.252
        ipv4_gateway: 10.10.10.254
  leshanbsserver:
    type: 'docker'
    image:
            'iotnetemu/leshan-bs-server:1.0.0'
    networks:
      ns3:
        ipv4_address: 10.10.10.253
        ipv4_gateway: 10.10.10.254
networks:
  ns3:
   type: 'ns3'
script: 'ns3_topology.py'
```

Listing 1: Configuration file for the demonstration scenario.

In the first phase, we perform the needed modifications to the Node and Application code. We modify the Zephyr sample by adding an overlay to support the QEMU network interface and a configurable LwM2M bootstrap ID through a *Kconfig* file. Afterwards, we create two container images containing the Leshan bootstrap and LwM2M servers.

The second phase focuses on creating the testing scenario. Each scenario is stored in a workspace that is nothing more than a folder that contains all the needed files. One such file is the configuration file that defines the Nodes, Applications and Networks and how they interact. The configuration file for this demonstration can be observed in Listing 1.

This file defines several *zephyr* nodes with the path for the Zephyr application and the parameters for its compilation. Furthermore, it defines two *docker* applications with the images to be used by the containers. Finally, a *ns3* type network is defined, and each Node and Application set its network configurations (e.g. the IP address) using a YAML key matching the corresponding Network. Additionally, the *ns3* node defines the path to a script that defines the network topology for the simulation. Such script can be observed in Listing 2.

from ns import ns

```
def setup_network_topology(nodes, applications):
    # Create nodes
    node_container = ns.network.NodeContainer()
    node_container.Create(len(nodes) + len(applications))
    # Create a CSMA network between nodes and applications
    csma = ns.csma.CsmaHelper()
    netdevice_container = csma.Install(node_container)
    # Add ns-3 devices to IotNetEmu objects. They will be connected
        to the TapBridge automatically.
    for i, node_name in enumerate(nodes):
        nodes[node_name]['ns3_device'] = node_container.Get(i)
    for i, application_name in enumerate(applications):
        applications[application_name]['ns3_node'] =
            node_container.Get(i + len(nodes))
        applications[application_name]['ns3_device'] =
            netdevice_container.Get(i + len(nodes))
```

Listing 2: ns-3 network topology script.

For the sake of simplicity for this demonstration, the topology script connects every device using a CSMA network. Thus this is not a limitation since we can use ns-3 models to simulate diverse wireless links.

The final phase is to run IoTNetEMU with the scenario workspace folder. The tool starts by loading the configuration file and creating all the defined components. Then, the Zephyr application is built for each node, the application images are pulled, the containers are created, and the ns-3 simulation is set up using the network topology script. Finally, a QEMU instance is launched per node, the containers are started, and the ns-3 simulation is launched. The architecture of this application can be observed in Figure 1.

Visiting the Leshan LwM2M web interface, we can verify that the application works correctly (Figure 2) and the nodes can communicate with the server.

The host can communicate with each application components using their IP address, as shown in Figure 3. Additionally, the tool creates a host-accessible serial port for each node, allowing the user to interact with the emulated device directly.

Finally, to collect data, users can use Wireshark or similar tools to capture data on the network interfaces created by IoTNetEMU or

IoTNetEMU - A framework to emulate and test IoT applications



Figure 1: High-level architecture diagram of the demonstration.

🥧 💩 leshan-server-demo	× +				~	- ×
	10.10.10.252:8080/#/clien			습	⊌ «	ර ≡
💩 LESHAN		CLIENTS	SERVER	😗 SECURIT		ABOUT
Registered Clients	Search					Q
Client Endpoint	Registration ID	Registration Date \downarrow	Last Update			
node3	Q2IWW7xFKT	Jul 14, 4:50:43 PM	Jul 14, 5:04:07 PM		0	
node5	lglwdlw0Cb	Jul 14, 4:50:42 PM	Jul 14, 5:04:09 PM		0	
node2	W8rVhxMdRI	Jul 14, 4:50:42 PM	Jul 14, 5:04:05 PM		0	
node4	DcOWvLwY54	Jul 14, 4:50:41 PM	Jul 14, 5:04:09 PM		0	
node1	VpyKxX0z0V	Jul 14, 4:50:40 PM	Jul 14, 5:03:59 PM		0	
			Rows per page: 10		5 <	>

Figure 2: Containerized Leshan LwM2M server with emulated nodes.

1000.0livsira@Delllatitude7300:-\$ ping 10.10.10.1 -c 2 PING 10.10.10.1 (10.10.10.1) 56(84) bytes of data. 64 bytes from 10.10.10.1: icmp_seq=1 ttl=64 time=0.731 ms 64 bytes from 10.10.10.1: icmp_seq=2 ttl=64 time=0.783 ms
10.10.10.1 ping statistics 2 packets transmitted, 2 received, 0% packet loss, time 1027ms rtt min/avg/max/mdev = 0.731/0.757/0.783/0.026 ms jono-otiveirad0etLatitude/2001:-\$ ping 10.10.10.2 - C 2 PINS 10.10.10.2 (10.10.10.2) 56(84) bytes of data. 64 bytes from 10.10.10.2: icmg_seq=1 ttl=64 time=2.63 ms 64 bytes from 10.10.10.2: icmg_seq=2 ttl=64 time=1.67 ms
10.10.10.2 ping statistics 2 packets transmitted, 2 received, 0% packet loss, time 1062ms rtt min/avg/max/mdev = 1.671/2.152/2.634/0.481 ms remo-diversepetiuertinde/2001; -\$ ping 10.10.3 - C 2 PING 10.10.10.3 (10.10.10.3) 56(64) bytes of data 64 pytes from 10.10.10.3; icmp_seq=1 ttl=64 time=2.750 ms 64 pytes from 10.10.10.3; icmp_seq=2 ttl=64 time=0.778 ms
10.10.10.3 ping statistics 2 packets transmitted, 2 received, 0% packet loss, time 1002ms rtt min/avg/max/mdev = 0.778/1.681/2.585/0.903 ms

Figure 3: Ping result for nodes 1 to 3.

use any of the tracing mechanisms available in ns-3. Figure 4 shows Wireshark capturing LwM2M messages from one of the application nodes.

This simple demonstration shows that IoTNetEMU can be used to test and validate IoT applications with minor changes to real code. It provides a framework for reproducible testing scenarios and eases some of the challenges associated with the development of heterogeneous IoT applications.

Dia Dia <thdia< th=""> <thdia< th=""> <thdia< th=""></thdia<></thdia<></thdia<>				*5 interfaces		
Image: Contract of the state of the sta	File	Edit View Go Captur	e <u>Analyze</u> <u>Statistics</u> Telephony <u>Wireless</u>	Tools Help		
Imput Control (Control (Contro) (Control (Contro) (Control (Contro) (Contro) (C		📕 🙋 🔘 🖻 🗍) 🛛 🖸 🔍 🔄 🔷 🕉 🖾 🤅	1 🛛 🛛 🗮 📕 k		
No. Decision Protocol Length (no. 0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	000	φ				
0 5 0 2007200 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 2007201 10 200720	¥0.	Time Delt	a Source	Destination	Protocol Length Info	
a 1 ± 2 ± 2 ± 2 ± 2 ± 2 ± 2 ± 2 ± 2 ± 2 ±		92 56.246673282	0.000009235 10.10.10.4	10.10.10.252	CoAP 187 ACK, MID:2523	0, 2.05 Content, TKN:0F
01 0.00000000000000000000000000000000000		93 56.243526846	-0.003146436 10.10.10.252	10.10.10.4	CoAP 61 CON, MID:2523	0, GET, TKN:08 17 e9 71
1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2		94 56.246476588	0.002949742 10.10.10.4	10.10.252	CoAP 187 ACK, MID:2523	0, 2.05 Content, TKN:88
1111 0.00000000000000000000000000000000000		95 56.246651414	0.000174826 10.10.10.4	10.10.10.252	CoAP 187 ACK, NID:2523	0, 2.05 Content, TKN:80
11 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 <td< td=""><td>-</td><td>116 62.697751895</td><td>1.395910652 10.10.10.252</td><td>10.10.10.5</td><td>CoAP 63 CON, MID:4339</td><td>, GET, TKN:0c a2 c1 a8</td></td<>	-	116 62.697751895	1.395910652 10.10.10.252	10.10.10.5	CoAP 63 CON, MID:4339	, GET, TKN:0c a2 c1 a8
11 10 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		117 62.700357396	0.002005501 10.10.10.5	10.10.10.252	COAP 66 ACK, MID:4330	, 2.05 Content, IKN:0C
111 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 1		118 62.697773530	-0.002583866 10.10.10.252	10.10.10.5	CoAP 63 CON, MID:4339	, GET, TKN:0c a2 c1 a8
111 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 121 1		119 62.700384281	0.002610/51 10.10.10.5	10.10.10.252	COAP 65 ACK, MID:4339	, 2.95 Content, IKN:0C
121 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 </td <td></td> <td>128 62.697762735</td> <td>-0.002621546 10.10.10.252</td> <td>10.10.10.5</td> <td>CoAP 63 CON, MID:4339</td> <td>, GET, TKN:0c a2 c1 a8</td>		128 62.697762735	-0.002621546 10.10.10.252	10.10.10.5	CoAP 63 CON, MID:4339	, GET, TKN:0c a2 c1 a8
Control Control <t< td=""><td></td><td>121 62.700371666</td><td>0.002508931 10.10.10.5</td><td>10.10.10.252</td><td>COAP 65 ACK, MID:4339</td><td>, 2.95 Content, IKN:0C</td></t<>		121 62.700371666	0.002508931 10.10.10.5	10.10.10.252	COAP 65 ACK, MID:4339	, 2.95 Content, IKN:0C
121 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 <td>_</td> <td>122 02.09//83209</td> <td>-0.002008400 10.10.10.202</td> <td>10.10.10.5</td> <td>COAP 63 CON, MID:4338</td> <td>, GET, TKN:00 82 01 88</td>	_	122 02.09//83209	-0.002008400 10.10.10.202	10.10.10.5	COAP 63 CON, MID:4338	, GET, TKN:00 82 01 88
212 0 2 70 0002000 0 0 0000000 0 0 00000000	-	123 62.700151181	0.002357921 10.10.10.5	18.10.10.252	COAP 65 A.K, MID:4330	, 2.85 Content, TKN:80
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10		124 02.09//33008	-0.00241/023 10.10.10.202	10.10.10.5	00AP 63 00W, MID:4338	, GET, TKN:00 a2 C1 a8
127 127 127 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 128 <td></td> <td>125 02.700340030</td> <td>0.0020004/2 10.10.10.0</td> <td>10.10.10.202</td> <td>00 ACK, H10:4330</td> <td>, 2.05 concent, INV.0C</td>		125 02.700340030	0.0020004/2 10.10.10.0	10.10.10.202	00 ACK, H10:4330	, 2.05 concent, INV.0C
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10<		130 188.474720903	49.772907053 19.18.19.1	10.10.10.252	COMP 68 COW, MID:3964	, PUSI, IKN:al 42 20 61
110 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 <td></td> <td>137 100.470001990</td> <td>0.001103027 10.10.10.202</td> <td>10.10.10.1</td> <td>00 ACK, H10:3904</td> <td>, 2.64 Changed, INN.al</td>		137 100.470001990	0.001103027 10.10.10.202	10.10.10.1	00 ACK, H10:3904	, 2.64 Changed, INN.al
Late definition Out Definition Definitio		138 188.474688784	-0.001193200 10.10.10.1	10.10.10.252	COMP 68 COW, MID: 3964	, PUSI, IKN:al 42 20 61
[01: Path; 74/40] [02: Path; 74/40] [03: Path; 74/40] [03: Path; 74/40] [04: Path; 74/40] [05: Path; 74/40] [06: Path; 74/40] [07: Path; 74/40]		133 1991410941101	0.00110341/ 10.10.10.202	10.10.10.1	COMP 00 ACK, H10:3964	, 2.04 changed, INStat
000 92 42 8a 8a 8a 7c 62 88 68 6c aa 55 88 68 🔛 66 18	, L ,	[Uri-Path: /3/0/0] [Response Inm: 0.00 [Response Inm: 0.00 [Retransmission of rr Payload: Payload Comb [Nume: Device [00] Kanufacturer: 24 [Nanufacturer: 24] TLV header [Resource Name: Maufacturer: 24 Manufacturer: 25657068797 Manufacturer: 25657068797	2399286 seconds] <u>speces in: 3137</u> vice (1 sizes) phyr unfacture] 2 yr	+tlv, Length: 8		
Z Constrained Application Protocol: Protocol Profile: Default Profile: Default	9999	02 42 8a 8a 6a fc 82	00 00 0c aa 55 08 00 🏭 00 - B	· · · · U · · 🗄 ·		
	0 7	Constrained Application	Protocol: Protocol		Packets: 155 · Displayed: 70 (45.2%)	Profile: Default

Figure 4: Wireshark capture of node 5 response to the server read request of resource /3/0/0.

ACKNOWLEDGMENTS

This work is co-financed by Component 5 - Capitalization and Business Innovation, integrated in the Resilience Dimension of the Recovery and Resilience Plan within the scope of the Recovery and Resilience Mechanism (MRR) of the European Union (EU), framed in the Next Generation EU, for the period 2021 - 2026, within project Microeletrónica, with reference 19.

Also supported by the "MLSysOps Project" (Grant Agreement 101092912), funded by the European Community's Horizon Europe Programme and by the Research Centre in Real-Time and Embedded Computing Systems – CISTER, funded by national funds through the FCT/MCTES (PIDDAC): Base Funding - UIDB/04234/2020.

REFERENCES

- 2023. An Introduction to Cooja. Retrieved Jul 12, 2022 from https://github.com/ contiki-os/contiki/wiki/An-Introduction-to-Cooja
- [2] 2023. QEMU A generic and open-source machine emulator and virtualizer. Retrieved Jul 12, 2022 from https://www.qemu.org/
- [3] 2023. Renode. Retrieved Jul 12, 2022 from https://renode.io/
- [4] Cédric Adjih, Emmanuel Baccelli, Eric Fleury, Gaetan Harter, Nathalie Mitton, Thomas Noel, Roger Pissard-Gibollet, Frédéric Saint-Marcel, Guillaume Schreiner, Julien Vandaele, and Thomas Watteyne. 2015. FIT IoT-LAB: A Large Scale Open

Experimental IoT Testbed. Milan, Italy. https://hal.inria.fr/hal-01213938

- [5] Tarun Goyal, Ajit Singh, and Aakanksha Agrawal. 2012. Cloudsim: simulator for cloud computing infrastructure and modeling. *Internation Conference on modelling optimization and computing* 38 (2012), 3566–3572. https://doi.org/10. 1016/j.proeng.2012.06.412
- [6] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya. 2016. iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments. CoRR

abs/1606.02007 (2016). arXiv:1606.02007 http://arxiv.org/abs/1606.02007

- [7] George F. Riley and Thomas R. Henderson. 2010. The ns-3 Network Simulator. Springer Berlin Heidelberg, Berlin, Heidelberg, 15–34. https://doi.org/10.1007/ 978-3-642-12331-3_2
- [8] Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. 2018. EdgeCloudSim: An environment for performance evaluation of edge computing systems. *Trans*actions on Emerging Telecommunications Technologies 29, 11 (2018). https: //doi.org/10.1002/ett.3493