HES-SO MSE


Design of communicative embedded systems
S1-2021

---

# DeSEm project

**10/01/2021**

---

Fabio Baldo

# Contents

# 1 Diagrams

## 1.1 DeseNET protocol documentation

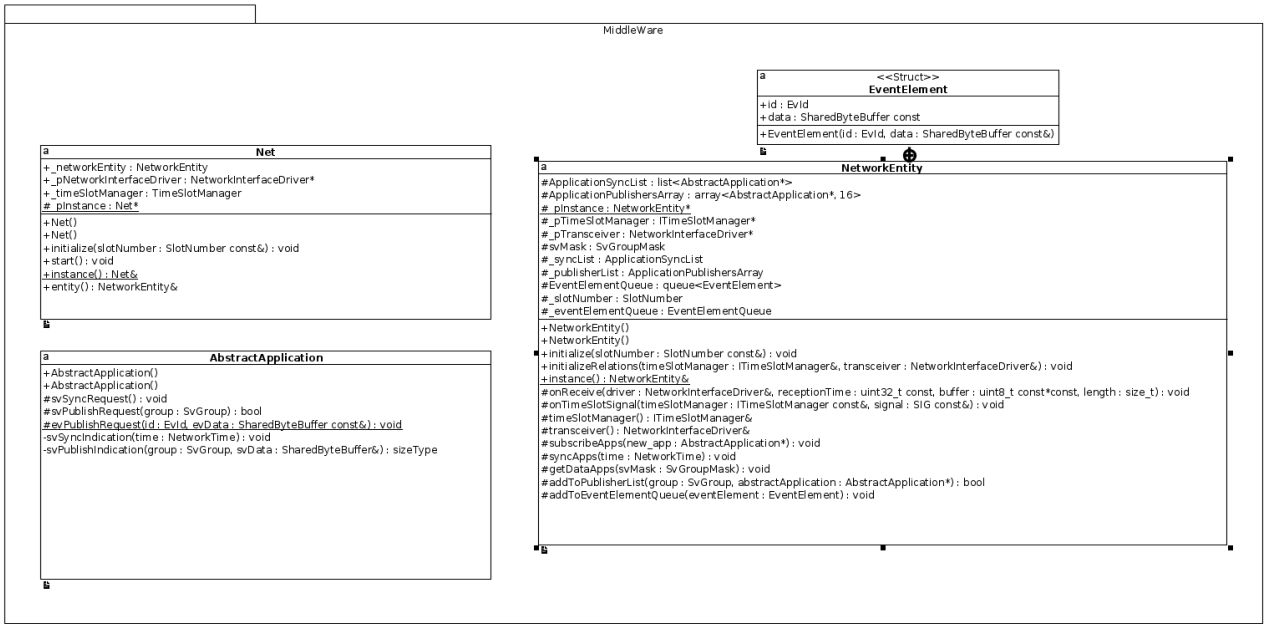


Figure 1: Class diagram of the sensor middleware

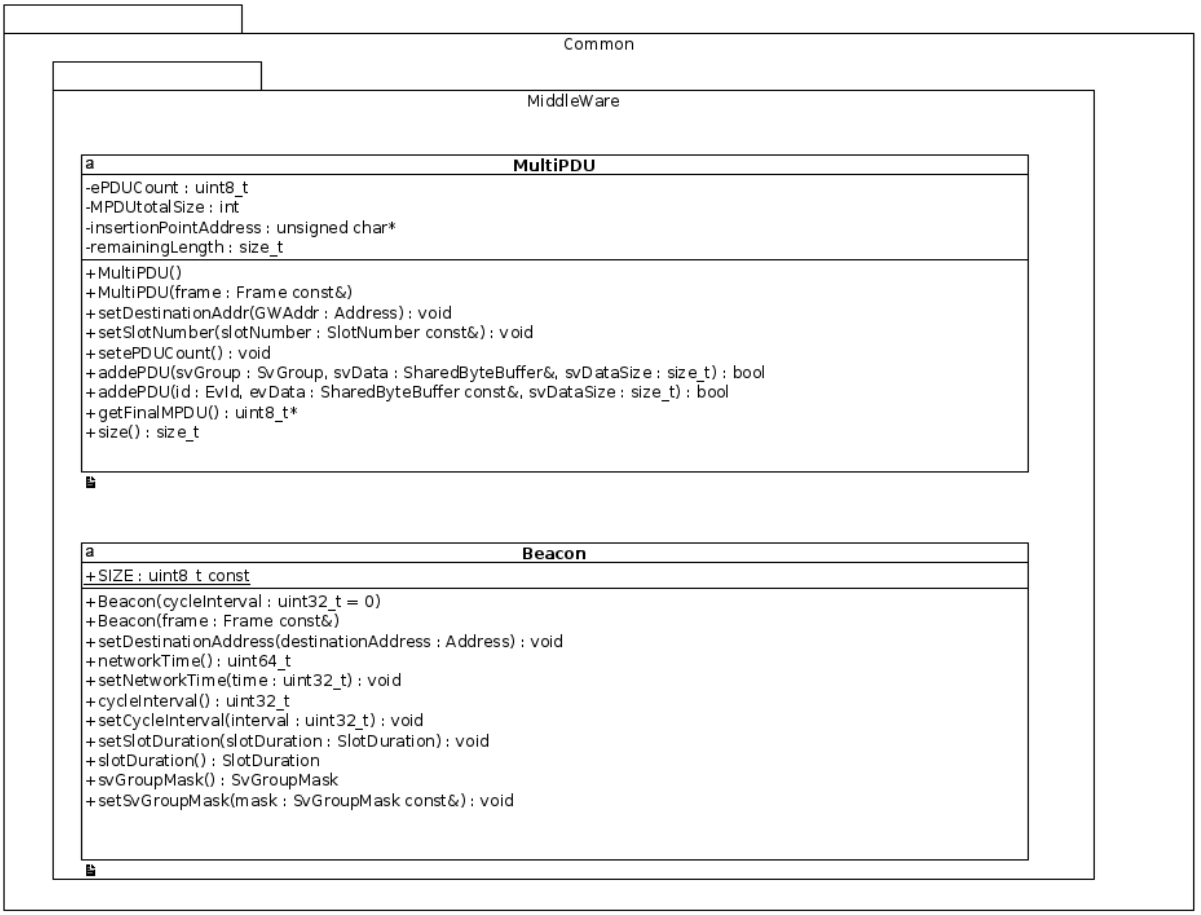


Figure 2: Class diagram of the two main classes of the Common middleware

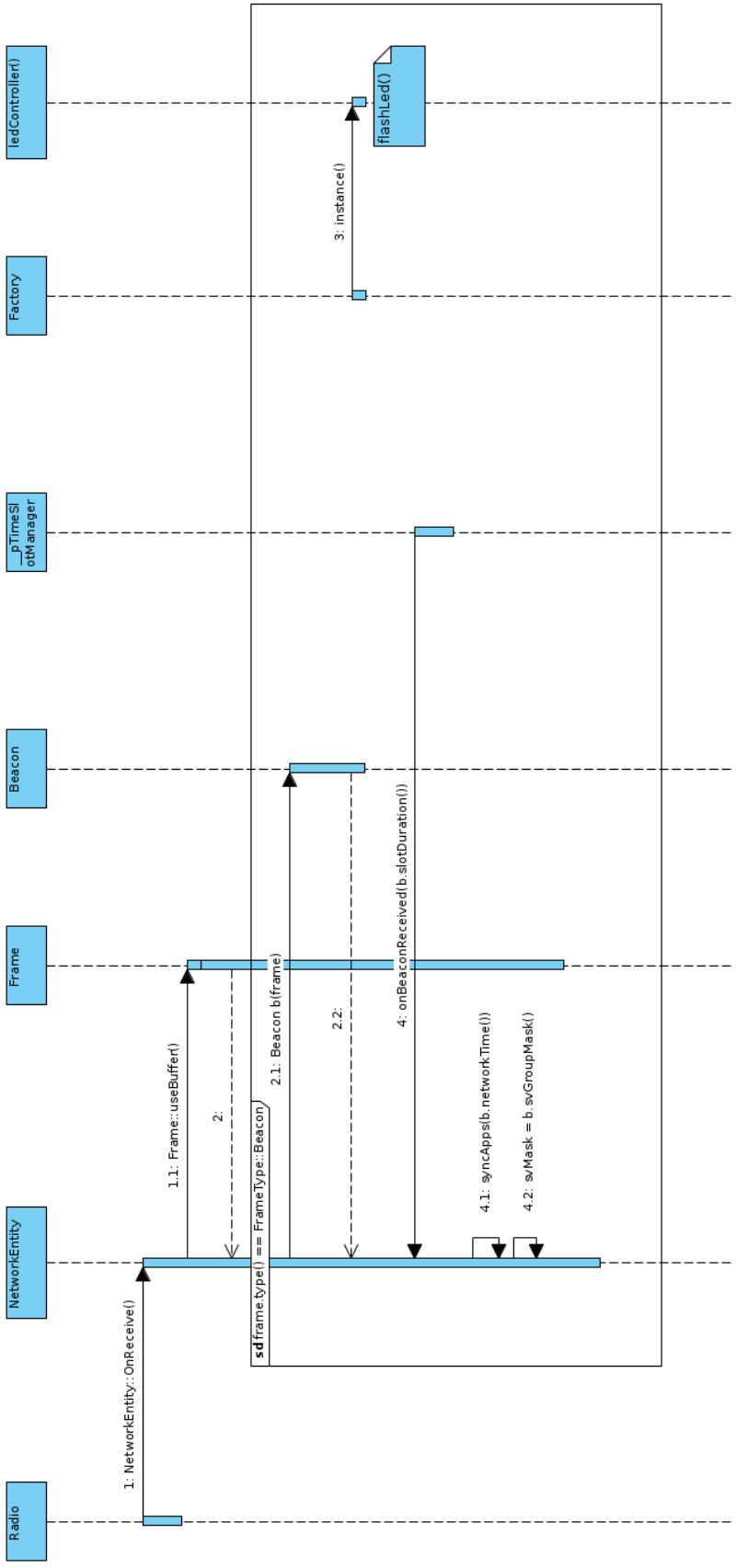


Figure 3: Action diagram of the function onReceive called when a Beacon is received

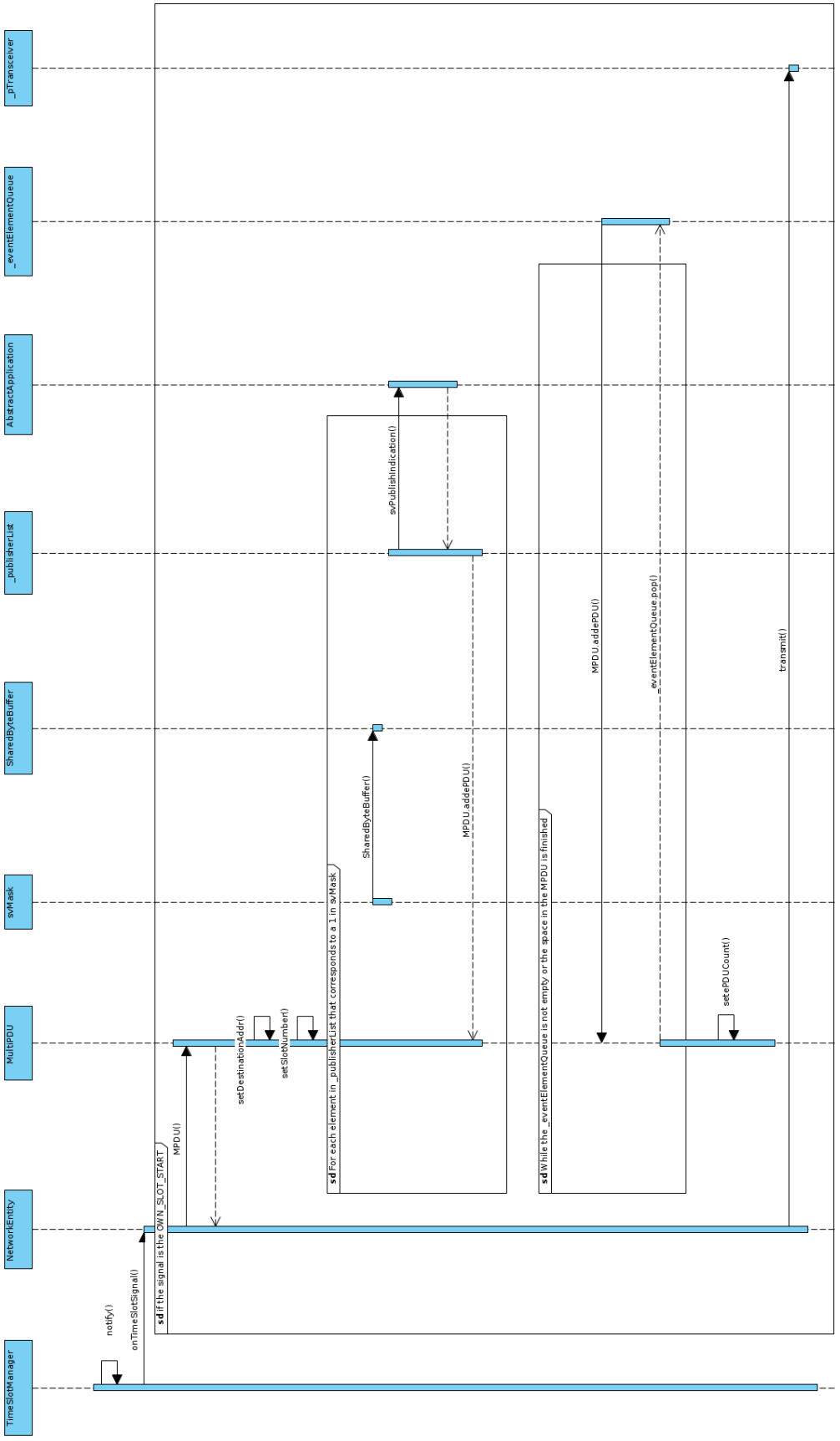


Figure 4: Action diagram of the function onTimeSlotSignal called when the timer for the slot runs out

## 1.2 Joystick application documentation

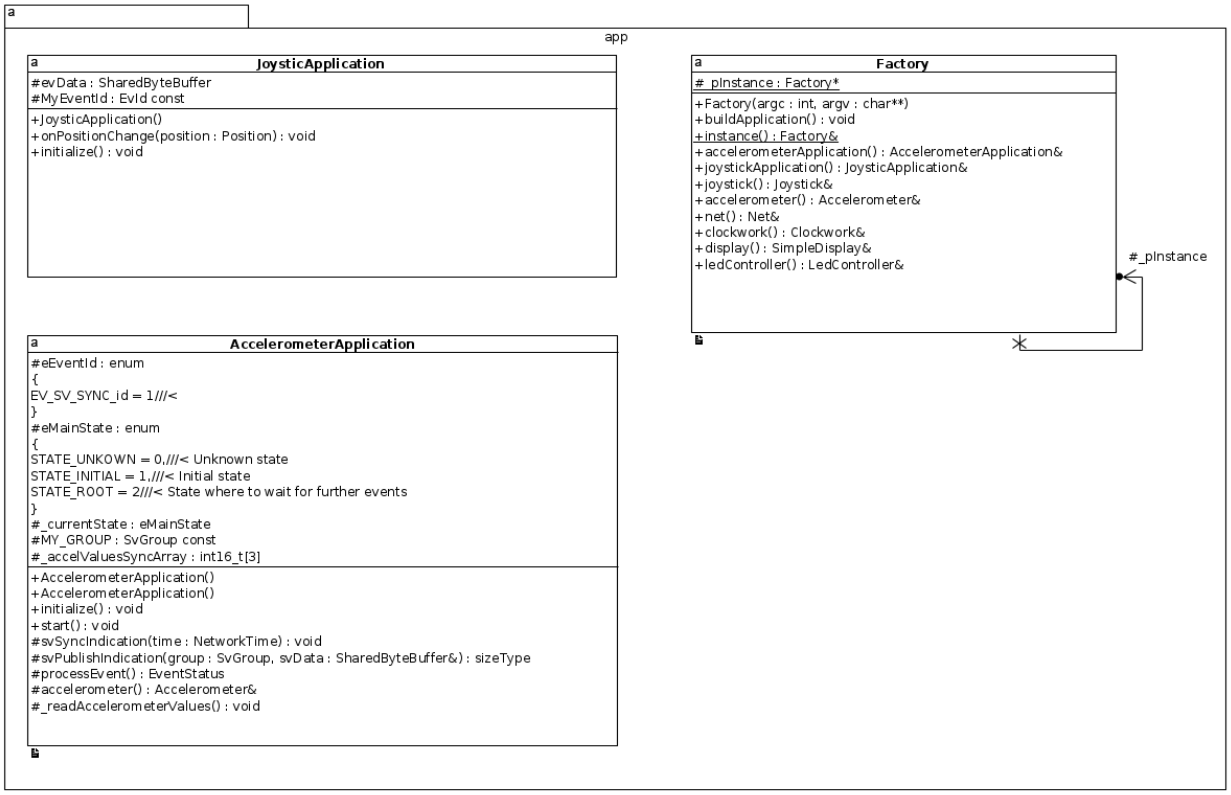


Figure 5: Class diagram of the two main classes of the Common middleware

# 2 Tests

## 2.1 DeseNET protocol test

In order to test the good functioning of the entire deseNET protocol a combination of Tracing and repeated test has been used. For controlling the correctness of the send data, using the Trace class and its method `outln` the content of the variables has been written to the console both when the data is sampled and send back to the Abstract application and when the MPDU is formed. The whole buffer hosting the MPDU data has been printed byte per byte in order to make it more readable for understanding its content. This MPDU eventually has been tested against the value shown in the mesh simulator for a final check. In the following listings and images is reported a sample of the testing results.

## 2.2 Joystick application test

Like in the previous section the correctness of the data of the events has been performed by a combination of tracing and repeated test with the double check of the result shown in the mesh simulator. The sample reported in the following images and listing represents a case in which the left button of the joystick has been pressed 3 times. Unfortunately the function, where

the evData are copied into the queue used for building the MPDU, is called two times. Many test have been performed in order to understand from where the error comes. Testing also the official demo it seams that when only one push is done to the button then two events are send to the GW. In order to try to avoid this problem during the tests of the rest of the application, the values of the push button have been set manually to `0xAB`. With this simple modification and a small one in the code where only one event over a couple is added to the the MPDU the result has been confirmed to be correct. In the reported example the official demo joystick has been pushed only once.

## 2.3 Results

Highlighted are the lines of the resulting MPDU containing the values.

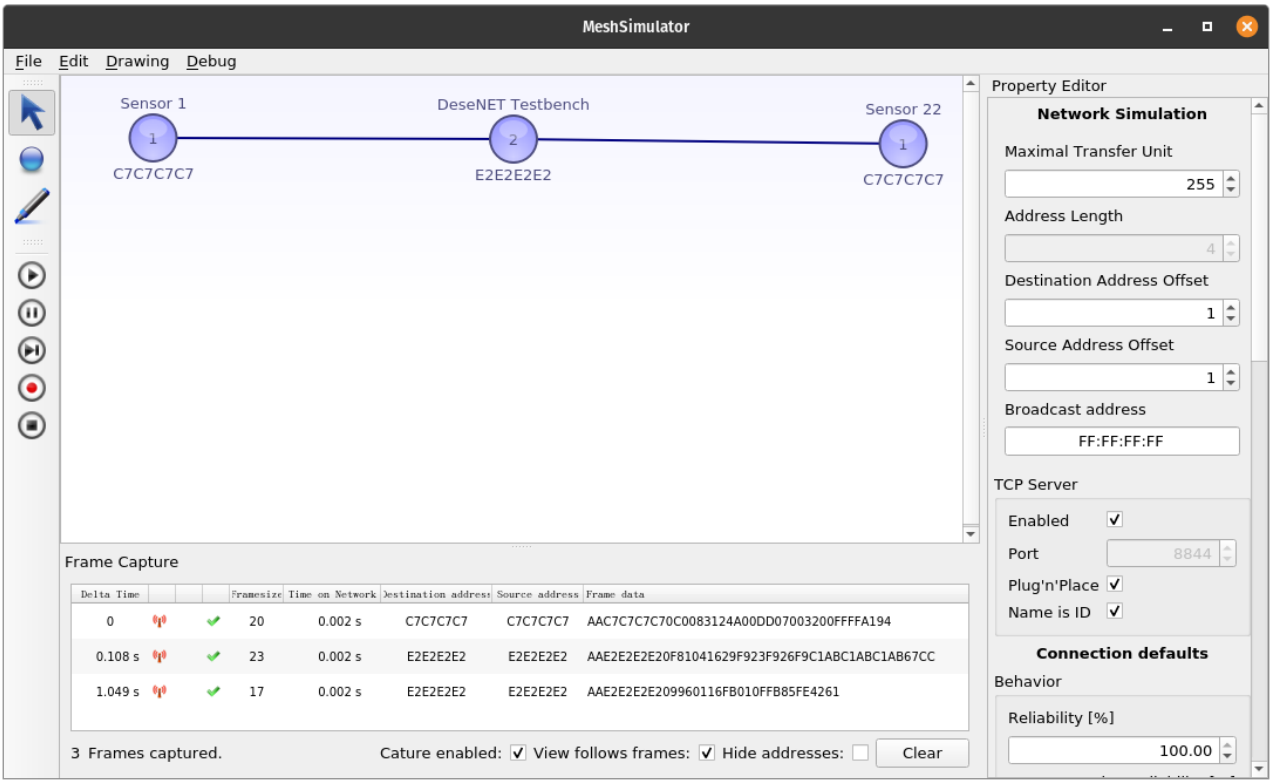


Figure 6: Screeshot of the simulator window after the MPDU was received from the Sensor back to the GW

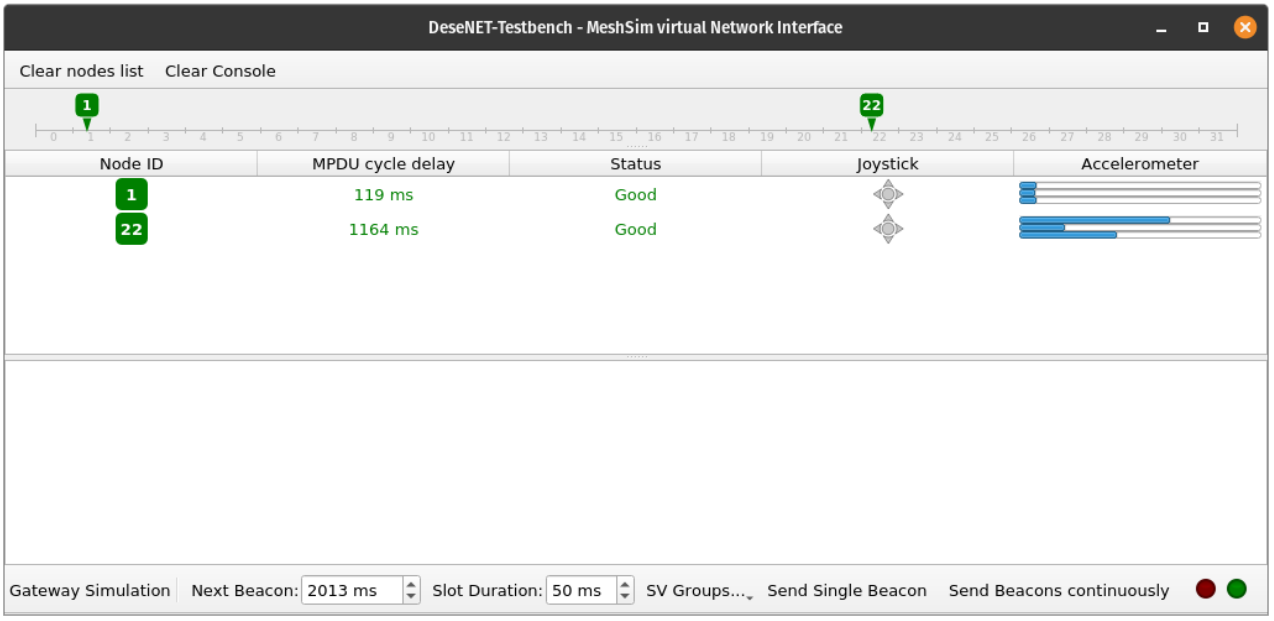


Figure 7: Screeshot of the testbench window after the MPDU was received from the Sensor back to the GW

```
AcceApplication data send: 29F923F926F9
NE_AcceApplication data send: 29F923F926F9
MPDU before sv-start
E2
E2
E2
E2
20
81
0
0
0
0
0
0
0
0
0
18
0
0
0
0
0
0
0
0
0
```

```
0
0
27
1
B5
1
E8
3
0
0
0
MPDU before sv-end
MPDU: svData 29F923F926F9
MPDU: mySVePDU 29F923F926F9
MPDU after sv-start
E2
E2
E2
E2
9
81
0
16
29
F9
23
F9
26
F9
MPDU after sv-end
NE_JoysticApplication: Data send: AB
MPDU before event-start
E2
E2
E2
E2
9
81
0
16
29
F9
23
F9
26
F9
MPDU before event-end
MPDU after event-start
E2
E2
E2
E2
B
81
```

```
0
16
29
F9
23
F9
26
F9
C1
AB
MPDU after event-end
NE_JoysticApplication: Data send: AB
MPDU before event-start
E2
E2
E2
E2
B
81
0
16
29
F9
23
F9
26
F9
C1
AB
MPDU before event-end
MPDU after event-start
E2
E2
E2
E2
D
81
0
16
29
F9
23
F9
26
F9
C1
AB
C1
AB
MPDU after event-end
NE_JoysticApplication: Data send: AB
MPDU before event-start
E2
E2
```

```
E2
E2
D
81
0
16
29
F9
23
F9
26
F9
C1
AB
C1
AB
MPDU before event-end
MPDU after event-start
E2
E2
E2
E2
F
81
0
16
29
F9
23
F9
26
F9
C1
AB
C1
AB
C1
AB
MPDU after event-end
MPDU-start
E2
E2
E2
E2
F
81
4
16
29
F9
23
F9
26
F9
```

```
C1
AB
C1
AB
C1
AB
MPDU-end

```