

# Beacon response

July 21st 2015

## Contents

<b>1</b>	<b>Scenario</b>	<b>1</b>
<b>2</b>	<b>Code</b>	<b>1</b>

## 1 Scenario

In the actual code, we send the response to the first beacon received.

So, at the beginning the node F doesn't have any beacon received and the authentication is not done yet. When the node R send a beacon, the node F start a random timer (30 sec max) and switch the value of *is\_beacon\_receive* to 1 and when this timer is finish, send the response. The value of *is\_associated* is 1.

If the node F receive a new beacon, the node doesn't send any response.

## 2 Code

The main code is in `/platform/lorafabian/apps/frame_manager/frame_manager.c`:

```
while(1) {
    PROCESS_WAIT_EVENT();

    //send respond to beacon
    if(is_beacon_receive && !is_associated && etimer_expired(&timer_payload_beacon))
    {
        etimer_stop(&timer_payload_beacon);
        coap_beacon_send_response();
    }

    if(etimer_expired(&rx_timer)) {
        leds_toggle(LED_S_ALL);
        pending = layer802154_pending_packet();
        printf("pending_packet: %d\n\r", pending);

        if(pending) {
            frame802154_lora_t frame = layer802154_read(&rx_msg, sizeof(rx_msg));
            size = frame.payload_len;
```

```

for(i = 0; i<size; i++)
    printf("%02x", frame.payload[i]);
printf("\n\r");

if(frame.header_len == -1)
    printf("Error: buffer is too small for headers");
else {
    //For the arduino
    int packetSize = size + frame.header_len;
    //Verify the destination of a message
    bool br_msg = is_broadcast_addr(&frame);
    bool my_mac = is_my_mac(&frame);
    if(br_msg) {
        printf("Broadcast message");
        if(!is_signaling(&frame) || debug_on_arduino)
            set_arduino_read_buf(rx_msg, packetSize);
    }
    else if(my_mac) {
        printf("Message is for me");
        set_arduino_read_buf(rx_msg, packetSize);
    }
    else {
        printf("Message is not for me");
        if(debug_on_arduino)
            set_arduino_read_buf(rx_msg, packetSize);
    }

    //Timer used to avoid collision
    if(respond_if_coap_beacon(frame.payload, size) && !is_associated)
    {
        is_beacon_receive = 1;
        //30 seconds max
        int random_timer = (random_rand()%30);
        etimer_set(&timer_payload_beacon, random_timer*CLOCK_SECOND);
    }
}
etimer_reset(&rx_timer);
}
}

```