IoT 2023 CHALLENGE 3

(1) Name: Nicolò Pisanu Person Code: 10827469

(2) Name: Marco Viviani Person Code: 10528650

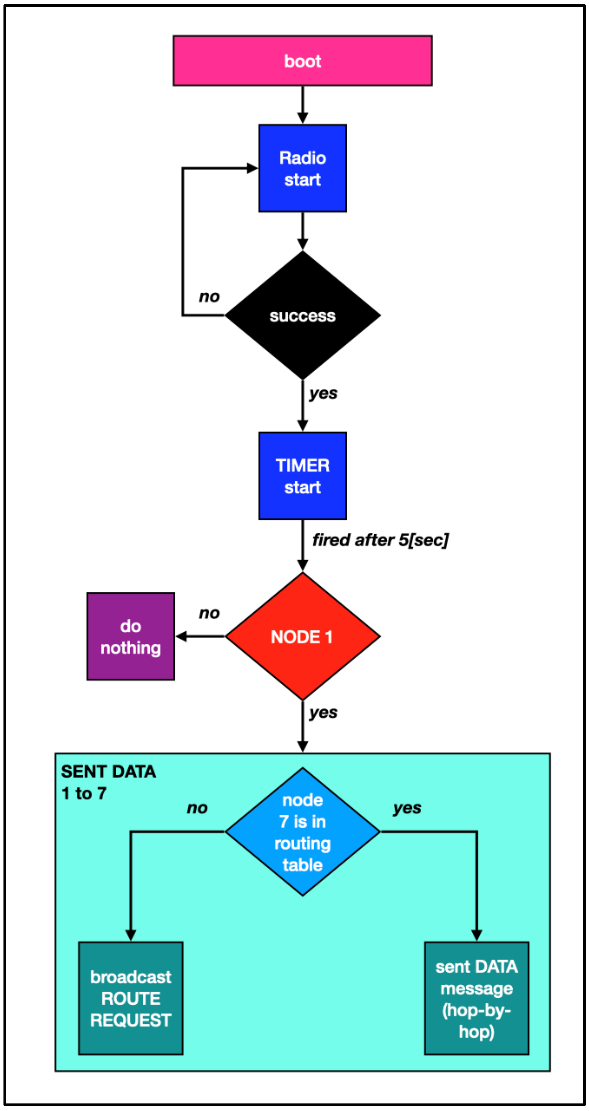
BOOT THE APPLICATION

Boot the application. If the booting has been successful turn on the radio communication and start Timer1, otherwise retry. The timer - after five seconds - triggers the Node 1 to try to send a DATA message to node 7.

TRIGGERING THE PROCESS

If the destination was present in the routing table of node 1, the node would send a DATA message hop-by-hop towards node 7; if not it generates a broadcast ROUTE_REQ to reach and find node 7. In this way the node 7 will generate a ROUTE_REPLY which will update the routing tables of all nodes.

In this case study the routing table of node 1 is initially empty, so it will send a ROUTE_REQ triggers the full communication process among the nodes.



HANDLING RECEIVED MESSAGES

Leds' routine:

Every time a node receives a message of any type (unless it is corrupted), a digit of the person code is extracted with a round robin cycle to compute the index of the led to toggle. The process prints leds' status dividing each digit of the bitmask by its most significant value.

Leds' status history of node 6: 000 , 010 , 110 , 111 , 110 , 111

ROUTE REQUEST message received:

- If the node requested is NOT in the node's routing table, it broadcasts the ROUTE_REQ. This is what happens after node 1 sends the first ROUTE_REQUEST. Indeed, all nodes have empty routing tables.
- If the node requested is in the node's routing table, reply in broadcast with a ROUTE_REPLY, setting the ROUTE_REPLY cost to the cost in my routing table + 1. In our process this case never happens.
- If the node requested is itself, reply in broadcast with a ROUTE_REPLY and setting the ROUTE_REPLY cost to 1. This is what happens when the node 7 receives the ROUTE_REQUEST from its neighbours originally generated by node 1.

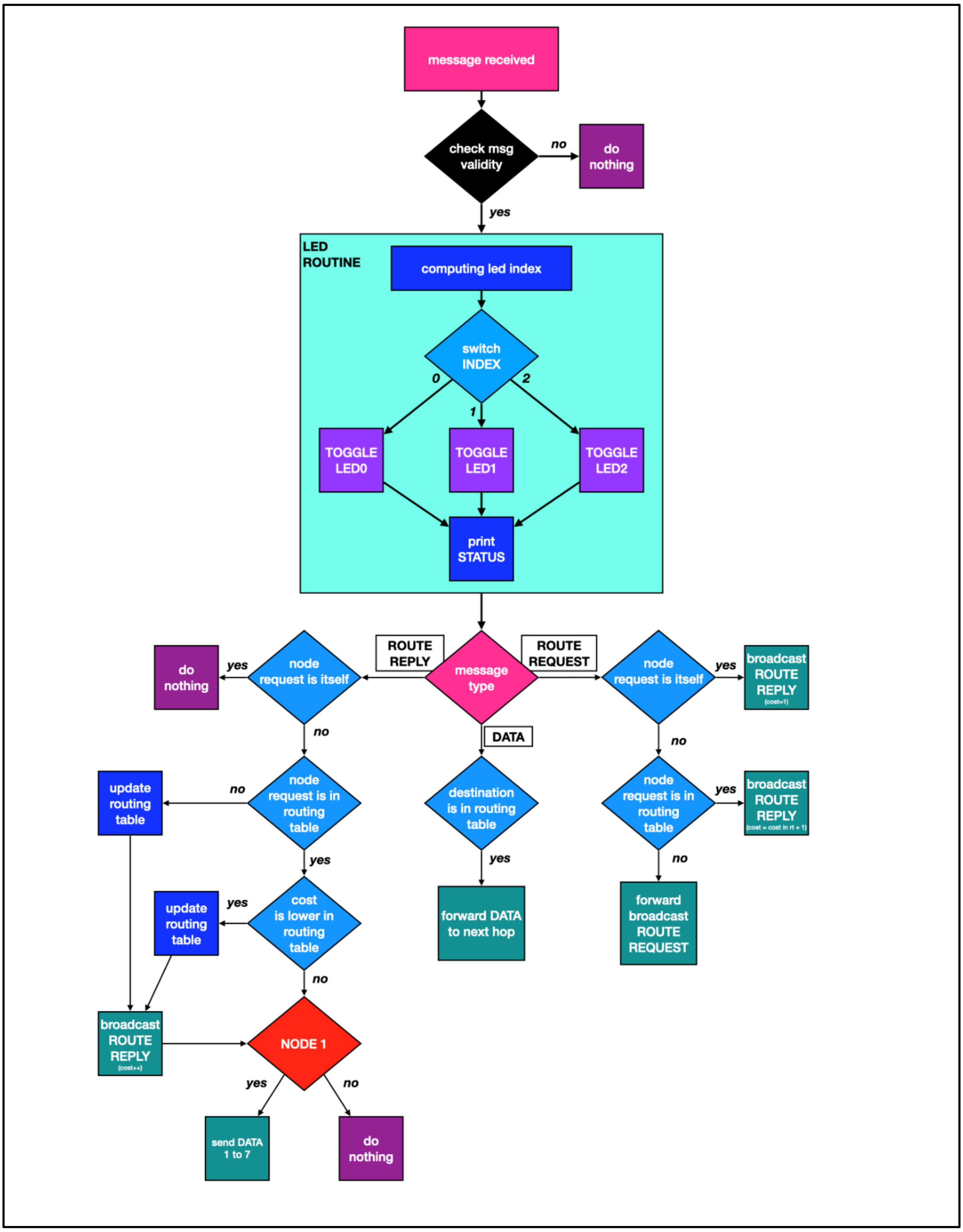
ROUTE REPLY message received:

- If the current node is the requested one in the REPLY, it doesn't have to do nothing. For example, this happens when the neighbours' nodes of 7 send the ROUTE_REPLY back to it.
- If the current node is not the node requested and if in its routing table there isn't the requested destination OR if the new cost is lower than the current cost, the node updates its routing table and forwards the ROUTE_REPLY in broadcast by incrementing its cost by 1. The nodes - forwarding the ROUTE_REPLY generated by node 7 - update their routing tables finding the shortest way to reach the requested node (which is node 7).
- Otherwise, if the routing table in not empty and the new cost is not lower than the current one, the nodes doesn't have to do nothing.

In any of these cases, as soon as the node 1 receives the REPLY_REQUEST from node 7 it can now send the DATA message towards node 7 hop-by-hop.

DATA message received:

- Check in the routing table if the destination is present. if it is present generate the same DATA message and send it to the next hop. In our case the path that starts from 1 and ends in node 7 passes through the nodes 1 – 3 – 5 – 7 . Node 3 receives the DATA from 1 and forwards it to 5, which in turn sends it to 7.
- Otherwise do nothing.



HANDLING GENERATE AND ACTUAL SEND

Every time we send a message, *generate_send* function is called that handles messages' sending by putting them in a queue and sending them through the *actual_send* function timing the process. Also, *generate_send* limits the number of ROUTE_REQ and ROUTE_REPLY that a node sends to 1.