INFO-Y118 : Mobile and embedded computing
Project

Demeulenaere Emeric
Jongen Philemon
Mahmoudi Hamza

15/05/2022

Contents

1 Introduction

For this class of Mobile and Embedded Computing, we've been ask to write a low level routing protocol and to test it with the cooja simulator on contiki-ng OS.

This report aims to give more insight about how it works and the technical choice we've made to implement it.

Firstly, we'll talk about the different message that are sent within our protocol, from the construction of the tree like shape network, until the message regarding the sensor's value.

Secondly, we'll talk about the routing and how we've managed to make this work. This goes into detail for new nodes in the network, how they choose parent, how their ranking is chosen,...

Lastly, we'll talk about the computation nodes as it's the most complete one. getting the insight of that node will allow to understand how the whole protocol as one work.

2 Messages

2.1 Message Structure

When a message is sent, the sending node firstly create an instance of the structure Message. This structure take several information like the rank of the sender, the address source and destination of the message, the type of message, the sensor value and the order to open the valve. All of these information are not mandatory to send a message.

2.2 Message Type

The type of message describe the behavior of the receiving node. The behavior depend on the type message but also on the type of node (Border, Computation and Sensor).

2.2.1 Keep Alive

- **ID :** 0
- **Type :** Unicast
- **Purpose :** Send a keep-alive message to reset the time-to-live field in the destination routing table.

2.2.2 Parent Request

- **ID :** 1
- **Type :** Multicast
- **Purpose :** Send a parent-request to every nodes in range to ask for following information : rank and and address of potential parents.

2.2.3 Parent Proposal

- **ID :** 2
- **Type :** Unicast
- **Purpose :** Send rank and address as response to a `Parent Request` message to propose become its parent.

2.2.4 Sensor Value

- **ID :** 3
- **Type :** Unicast
- **Purpose :** Send value retrieved by a sensor. When incoming in a Computation node, this can compute the slope or forward it (explain in Computation nodes section).

2.2.5 Open Valve Order

- **ID :** 4
- **Type :** Unicast
- **Purpose :** Send from a Computation node or from the server, it command a sensor to open the valve for 600 secondes.

2.2.6 Init Msg

- **ID :** 10
- **Type :** Broadcast
- **Purpose :** Are send once, when the node arrive on the network to initialize the `nullnet_buf` `nullnet_len`

Most of these messages are proceed differently according to the type of node that receive the message. For example, a Computation node and a Sensor node won't proceed an incoming Sensor value (ID : 3) in the same way.

3 Routing

Concerning the routing, we were inspired by the `nullnet` example of `contiki-ng` repository. Routing starts by initialization of the local routing tables for each node type. They are designed in array of `routingRecord` where we can find the destination address, the next hop and a TTL.

3.1 New node on the network

First we have the Border Router Node, which is the root of the tree with a rank set to 1 and which will never change. Sensor nodes and Computation nodes start with no parent and with the maximum rank value, 99. If a node has no parent node, it is inactive on the network. That mean that it don't send outgoing messages and it ignores incoming messages except messages when the `msgType` specify that the incoming message is a `parentProposal`. That message type allow nodes to choose a parent.

3.2 Choosing a parent

When a node arrives, after initializing its variables, on the network, its first action is to broadcast a `parentRequest` to all nodes in its range. To choose a parent, there is 3 possible scenario's :

1. The node has no parent and take the first `parentProposal` he receive and save this node as parent.
2. The node receive a `parentProposal` and compares its rank with the rank of its actual parent. If the proposed parent's rank is lower, the node changes its parent for the received.
3. The node receive a `parentProposal` and compares the ranks but hold his actual parent because its rank is lower as the proposed parent.

A node cannot send a `parentProposal` when it don't have a parent yet. This ensure to avoid that a node accept a `parentProposal` from a node unconnected to the network. Note that every node send periodically `parentProposal` in broadcast to allow node that would have move their range to get an optimized parent.

3.3 Rank choosing

The Border node has constant rank = 1. Once the node get a parent, it retrieve its rank from the `parentProposal` and increment it by one to set its own rank. The closer the node is from the Border rank, the lowest is his rank. This ensure that, during the rank comparison step of choosing a parent, a node will chose as parent the closest node from the Border node.

3.4 Keep-alive

If a node has a parent and thus is part of the network, it periodically sends `keep-alive` messages to its parent to reset the time-to-live field in their routing table.

4 Computation nodes

The computation node have the ability to intercept values from Sensor nodes (for a limited number of Sensor nodes) and to compute the slope and decide if the Sensor node has to open the valve or not.

4.1 Timer

To act periodically, a node uses timer. It intialize a `SEND_INTERVAL` to 1 and set a timer that increases a variable `count` every second. Based on that 1 second-timer and `if(count%X == ...)` conditions, the node proceed to action every X seconds.

4.2 Receiving messages

The `input_callback` function allow the Computation node (and every other nodes) to have a listener on incoming messages. When a Computation node receive a message, it proceed to differents steps :

- Updates the `time-to-live` field in his routing table for the previous hop.
- Retrieves every information from the incoming message and especially `typeMsg` to determine the action to undertake.
- Determines if it has to proceed to an action or if it just have to forward the message. This decision is based on the `typeMsg` value.
  For example : an `OpenValveOrder` arriving at the Computation node will be directly forwarded to the next hop to reach the destination address.

4.3 Different behaviors for different types of message

The different types of messages that will impact the Computation node (other that forward the message) are the following :

- `keep-alive` [arriving in unicast](= ping) : The Computation node will update the time-to-live field from his routing table.
- `parentRequest` [arriving in broadcast] : The Computation node will answer to the request with an unicast, by informing his own address and rank.
- `parentProposal` [arriving in unicast]: The Computation node receive this after asking for a `parentRequest`
- `sensorValue` [arriving in unicast] : The Computation node 2 possibilities :
  - Handle the sending Sensor and compute his slope.
  - Forward the Sensor value if he already manage its maximum number of Sensor handled.

  The different steps to handle a Sensor are the following :

  - Looks if the Sensor is already in his handled list.
  - If not, it add it to its list and add the first received value.
  - If the Sensor is already in its list, the Computation node add the value to and array of value linked to that Sensor.
  - If that array has 30 values from that Sensor, it compute the slope and decide, based on a threshold to send an `openValveOrder` message or not.

- `openValveOrder` [arriving in unicast] : The Computation node just forward message to the next hop to reach the destination address.