FINAL PROJECT
Keep Your Distance
IoT 19-20

Name Surname	Person Code	Id Number
Andrea Crivellin	10491856	928320
Gabriele Guelfi	10491169	916207
Amin Soltanian	10711914	943340

Link repo git:
https://github.com/GabrieleGuelfi/IoT_assignments2020.git

IMPLEMENTATION

TinyOs

We started from *distance.h*, defining the message struct that includes:

- **ID**: the *TOS_NODE_ID* of the sender mote

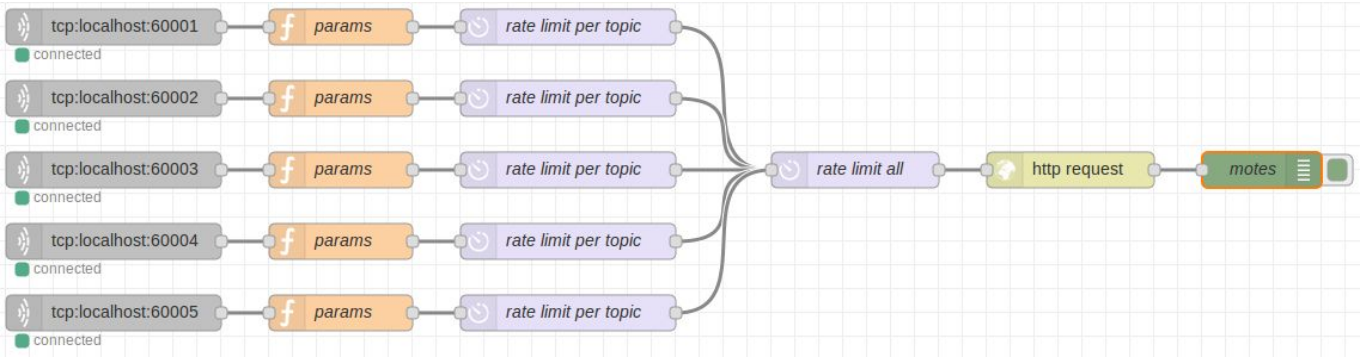
In the *distanceAppC.nc* file we defined and linked the components we needed:

- AMSenderC, AMReceiverC and ActiveMessageC for communication
- TimerMilliC to start the periodic timer of every mote
- SerialPrintfC to print strings through serial interface

Now we'll describe the *distanceC.nc* and the most relevant functions and events:

- **SplitControl.startDone**: every mote start their periodic timer(2Hz)
- **MilliTimer.fired**: every time the timer fires, each mote sends a broadcast message containing their own *TOS_NODE_ID*
- **Receive.receive**: whenever a mote receives a packet, it prints the received payload (the *TOS_NODE_ID* of the close mote) as *value1* and its own *TOS_NODE_ID* as *value2* through serial interface in a well formatted JSON string

Node-RED

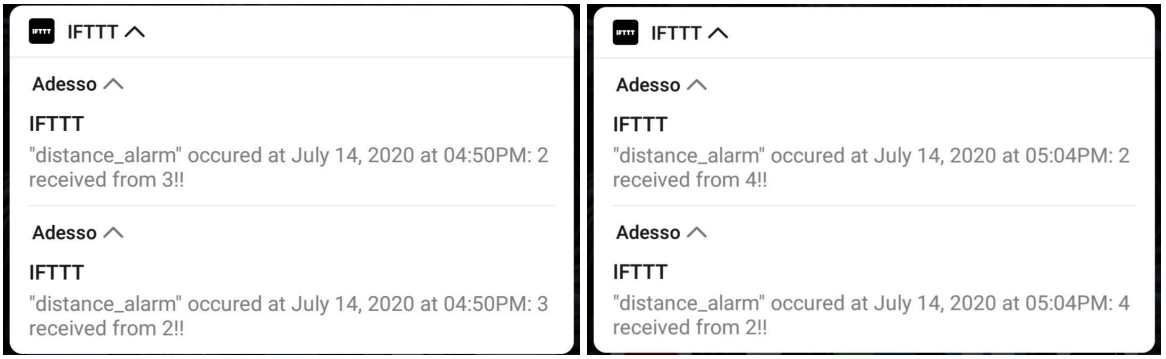


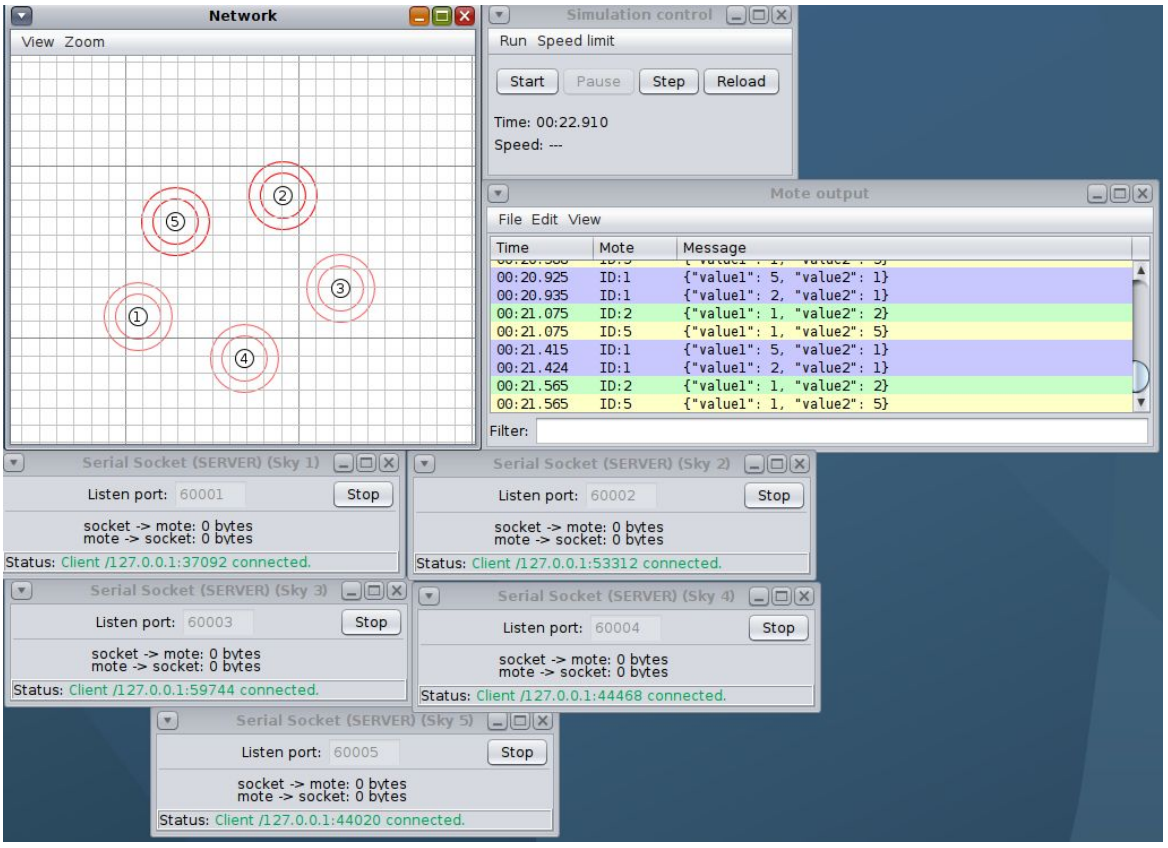
Each node chain represents one mote and they have the same functions, so below just one chain is described:

- **TCP_node:** it's connected to the serial socket of the mote and receives data from it (HOST: localhost, PORT: 60000 + mote_ID, delimiter: \n)
- **params:** it creates a message with *value1* as topic and both *value1* and *value2* as payload, ready to be sent to IFTTT
- **rate limit per topic**: for each topic, so for each different mote close to the current one, we store the first message arrived, dropping the following ones, and forward them all together every one minute; this node is useful to avoid spamming of many notifications per second
- **rate limit all:** this node limits the message rate to one each 2 seconds; we discovered that sending too many messages simultaneously was causing IFTTT to drop some of them
- **http request**: this node makes an HTTP POST to IFTTT service at the following URL: *https://maker.ifttt.com/trigger/distance_alarm/with/key/rLYZ_UpLa3UsOj6aPuBTO*

SIMULATION

The simulation has to be started in **Cooja**, adding five sky motes and starting a Serial Socket on each of them. Then the flows continues through **Node-RED** where we left debug nodes to check the correctness of data. The final output can be seen through **IFTTT** notification service.





IFTTT

Following the guide at https://wiki.instar.com/Advanced_User/Node-RED_and_IFTTT/, we created our own Event "*distance_alarm*" that shows a push notification with the following text:

"{{EventName}}" occured at {{OccurredAt}}: {{Value1}} received from {{Value2}}!!

A couple of examples are shown in the pictures above.

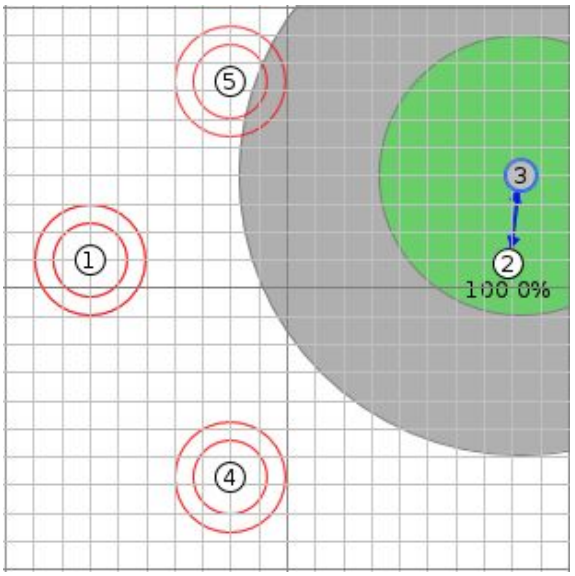
SCALABILITY

We put a special focus on scalability, we did the implementation with five motes because it is a significant pool for testing. Although we designed the solution in such a way that adding or removing motes is really easy, the only modification needed is to add/remove a block of nodes (*tcp, params, rate limit per topic*) and connect it to *rate limit all*. Of course if a simulation environment like Cooja is used, it is necessary to add/remove sky motes with the related Serial Port.

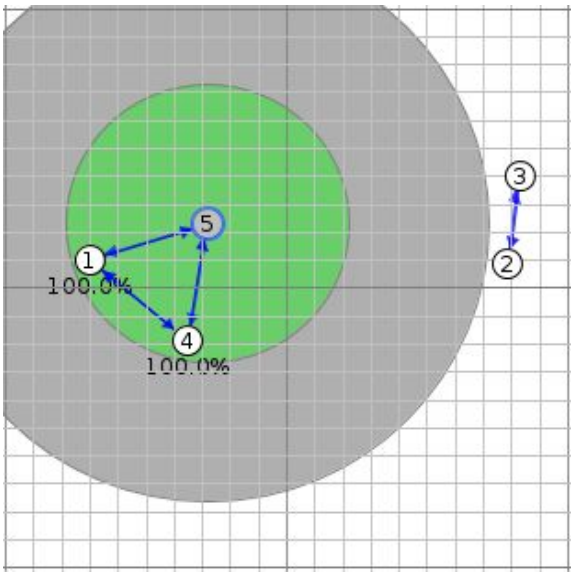
APPENDIX

The following images show the different scenarios of the log file:

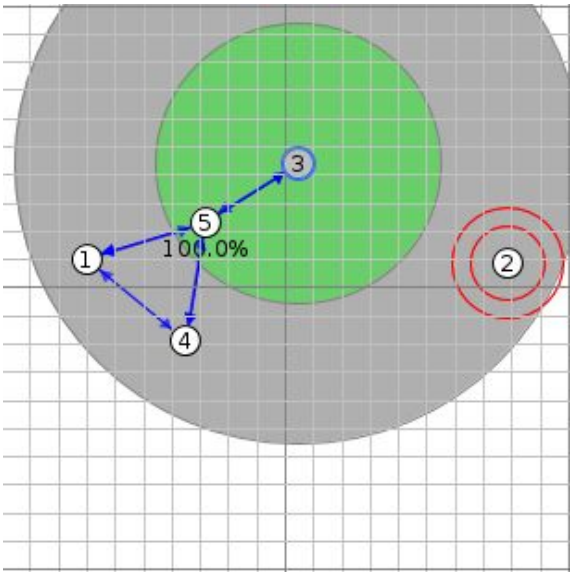
Scenario 1:



Scenario 2:



Scenario 3:



Scenario 4: