Keep your Distance

Andrea Calici, C. Persona 10490117, Matricola 944717

Each mote has one timer for each other mote and two arrays of counters to keep track of the messages received by them. I decided to implement the project resetting the counter of a mote when the interval between two messages received by the same sender is greater than 3s. Indeed, when a mote receives a message from the same sender it increments a counter; if the receiver doesn't receive new messages for 3s, it resets the counter of the related sender. **A slim version of the project can be done removing all the timers and avoiding to reset the counter.**

TinyOS The TinyOS part of the project is composed by three main compontents that are described in the following list:
-**KeepYourDistance.h**: this element defines the format of the messages sent by the motes that contain a single field that is the id of the sender of the message (`sender_id`).
-**KeepYourDistanceAppC.h**: this elements allows to define the components and link them to the interfaces.
-**KeepYourDistanceC.h**: this elements represents the logic of the application. It is composed by:

- `MilliTimer`: a timer to send messages every 500ms;
- `ResetTimer_i`: a timer for each motes, where 'i' represents the number of the sender mote;
- `counters[]`: an array of counters to keep track of the number of messages received from each motes. The position 0 represents mote 1, the position 1 represents mote 2 and so on;
- `reset_counters[]`: an array of counters to keep track of the time past from a message and another form the same mote. The position 0 represents mote 1, the position 1 represents mote 2 and so on.

The application proceeds as follows:

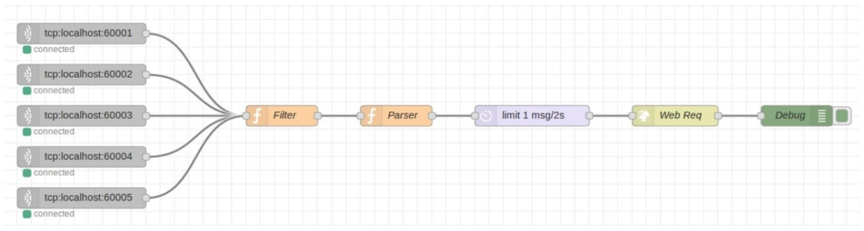
1. each motes starts and runs a timer that fires every 500ms to sent a broadcast message that contains the sender id;
2. if a mote receives a message from another, it increments the related counter of the array `counters[]` and `reset_counters[]`. Then it runs the specific timer calling the `startResetTimer` function that choose the correct timer and starts it.

This timer allows to know if the messages received by the same sender are arrived consecutively or there was an interval greater than 3s after which the counter is reset;

3. if the counter of the messages from the same sender is 10, the receiver prints an alarm in JSON format containing its id and the id of the sender that will be handled by Node Red.

Cooja The simulation of the application was launched using Cooja, using 5 Sky Motes. Each motes enables his socket port to talk with Node Red and send the alarms received by the motes.

Node Red The flow of Node Red was implemented starting from the following tutorial: `https://wiki.instar.com/Advanced_User/Node-RED_and_IFTTT/`.



It contains:

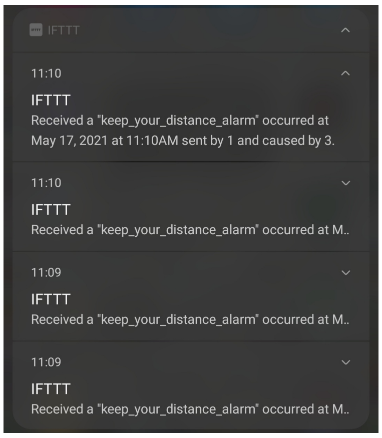
- `tcp:localhost:6000i`: a *tcp in* node that receives all the messages from Cooja;
- `Filter`: a *function* node to discard all the debug messages (that contains 'DEBUG') sent by the notes that aren't alarms;
- `Parser`: a *function* node that removes all the wrong characters derived from the *printf* function, parses the alarms and sends them to the following node;
- `limit 1 msg/2s`: a *limit* node that limit to 1 message/2s;
- `Web Req`: an *http request* node that sends the messages to the IFTTT applet through a POST request with the url `https://maker.ifttt.com/trigger/event/with/key/API_KEY`;
- `Debug`: a *debug* node that prints all the alarms.

IFTTT Following the tutorial above, I created an applet that receives request from Node Red and sends notifications to the application already installed on my smartphone. The text of each notification is the following:

Received a "keep_your_distance_alarm" occurred at *TIMESTAMP* sent by *MOTE_A* and caused by *MOTE_B*.

where:

- *TIMESTAMP*: the timestamp of the alarm.
- *MOTE_A*: the sender;
- *MOTE_B* the mote that didn't keep the distance from the sender.



IFTTT

11:10
IFTTT
Received a "keep_your_distance_alarm" occurred at May 17, 2021 at 11:10AM sent by 1 and caused by 3.

11:10
IFTTT
Received a "keep_your_distance_alarm" occurred at M..

11:09
IFTTT
Received a "keep_your_distance_alarm" occurred at M..

11:09
IFTTT
Received a "keep_your_distance_alarm" occurred at M..

Obviously in most of the cases, two alarms will be sent simultaneously by the two nodes involved in the violation of distances. The only case where a single alarm is sent occurs in the following situation:

- node A and node B are sending messages and incremented their counters;
- obviously the time in which they send the messages is not equal but can change in milliseconds;
- node A goes away from B and moves close to B in a time that is greater then 3s from the last message sent by him, but lesser than the interval that occurs from to messages sent by B;
- the counter of A is reset, but the one of B isn't.

Log The `log.txt` file contains 6 scenarios that can help to understand the application. It containts 2 kinds of messages:

- debug: contains the mote **i** that sent the debug message, the node **j** from which it received the message and the counter **k** of the messages received by j

  [DEBUGi] Current counter for node j: k

- alarm: contains the sender of the alarm **i** and the motes that doesn't keep the distance **j**

  {"value1": i, "value2": j}