

Keep your distance

INTERNET OF THINGS - PROJECT REPORT

Sergio Cuzzucoli 10532314

Daniele De Dominicis 10502843

JUNE, 2021

1 Data structures

1.1 Message

Each mote exchanges a message composed only of the sender ID. In particular the sender Id is a 16 bits unsigned integer; this allows us to have up to $2^{16} - 1$ IDs.

1.2 Contacts

Each mote keeps track of the other motes currently in its range through a structure that contains:

- **id**, id of the mote in range stored as 16 bits unsigned integer;
- **timeStamp**, indication of time of the last message sent by said mote stored as 32 bits unsigned integer;
- **counter**, with values in range $[1,10]$ stored as 8 bits unsigned integer.

2 Basic functioning of the mote

Each mote broadcasts a message every 500ms.

When two motes come in contact they exchange messages and the receiving mote starts to populate its contact list with all the information of the sender. If the two motes continue to exchange packets every 500ms then the counter of the respective contacts is increased by 1 every time. When the counter reaches the value of 10, the mote sends an "Alarm message" signaling that it has been in contact for too much time with the mote.

We assume that one mote will not communicate, at least at the same time, with all the other motes, thus the list of contact has a fix length which is lower than the number of possible motes. This is the reason behind the fact that the list is updated every firing of the timer to send the message.

Before sending a message every mote checks every entry in its contact list: if the difference between the current value of the timer and the timestamp of the last message received is grater than 500ms, then the mote has not received any keep-alive from the contact in 500ms. In this case we realize that the two motes went out of range and thus there is no need to keep the information about the contact.

3 From Cooja to NodeRed

Each mote prints a message that is shown in Cooja and forwarded to NodeRed.

4 NodeRed

In NodeRed, nodes stay always up listening for messages sent by motes. Such messages converge into a single node which will route them according to its type.

4.1 General log flow

Every message is sent to another node which will remove the impurity, constituted by random characters injected at the beginning of the message by cooja. A timestamp of the in the format [YY/MM/DD HH:MM:SS] is then added at the beginning. Finally, the resulting lines are appended to a txt file which will work as the log file of the application.

4.2 Alarm message flow

In addition to being stored in the log files, ALARM messages have also to be sent to the user via IFTTT.

First of all they undergo a process which removes the impurity at the beginning and extracts the motes ID, then these IDs are passed through a POST request via parameters of the URL at our IFTTT Applet.

5 IFTTT

The behavior of the Applet is quite simple:

- **IF-THIS side:** the Applet exposes a webhook, in order to receive the IDs from NodeRed.
- **THAN-THAT side:** once triggered by the reception of a message from NodeRed, the applet sends a message to the Telegram account of the IFTTT account owner.

Message text

Mote **Value1** entered in contact
with Mote **Value2** **OccurredAt**

6 Log

Logs file store five types of messages:

1. **[START:X]** Mote X has correctly started;
2. **[RANGE:X/Y]** Mote X has come in contact with mote Y;
3. **[UPDATE:X/Y-N]** Mote Y is still in range of mote X and it has sent the N-th message (with N which can have values between 2 and 10);
4. **[OUT:X/Y]** Mote Y is no longer in the range of mote Y.

7 Demonstration