|  |  |
| --- | --- |
| Activity 2  Wireless Sensor Network with TOSSIM |  |
|  | **Diego Costanzelli**  [diego.costanzelli@mail.polimi.it](mailto:diego.costanzelli@mail.polimi.it)  ID number: 10527966 - 928941  **Francesco Maffezzoli**  [francesco.maffezzoli@gmail.com](mailto:francesco.maffezzoli@gmail.com)  ID number: 10576556 - 944914  **Marco Passera**  [passera.marco@alice.it](mailto:passera.marco@alice.it)  ID number: 10531470 - 944947 |

# Goal

The goal of this project is to create a TOSSIM simulation, made by two TinyOS motes. The first mote constantly sends requests in broadcast. When the second mote wakes up, he receives and acknowledges the request. Upon receiving the Acknowledgment, the first mote stop sending further requests.

# Constraints

* Message contains a counter, the data and the type (1 for REQ, 2 for RSP)
* Requests are sent at a frequency of 1 Hz for mote 1.
* Mote 2 wakes up after 5 seconds.
* When mote 2 receives an ack, he stops sending requests.

# Implementation

The project is made from 6 files:

* sendAck.h
* sendAckAppC.nc
* sendAckC.nc
* RunSimulationScript.py
* Log file
* topology.txt

**sendAck.h** is the header file where message structure is defined. Here we have declared the message ﻿*my\_msg\_t* containing:

* A counter
* The data
* The value of the message (1 for REQ, 2 for RESP).

**sendAckAppC.nc** is the configuration file where components are defined and the relationship between them is given. This file is almost identical to the one we had as reference, with the addition of the sendAck component.

**sendAckC.nc** is the module component where we have implemented the interfaces. For each event, a function with actions to do is defined:

* For mote 1, a timer is set every 1000ms. Every time the timer fires, a new request is sent in broadcast, containing the sendAck.h message type.
* For every request sent, an Ack is requested using the function PacketAcknowledgements.requestAck.
* After 5 seconds mote 2 wakes up and starts receiving requests. When a new message is received, after verifying that it’s a request message, the motes 2 sends back to mote 1 a packet containing data from the fake sensor. Furthermore, since mote 1 asked for an Ack (using PacketAcknowledgements.requestAck) an Ack for the request message is sent.
* When the Ack is received in mote 1 the timer is stopped. In this way no more request will be sent.

**Topology.txt** is a very simple file containing the motes topology. It’s composed of only 2 motes.

**RunSimulationScript.py** is the python file. Is very similar to the given sumulation file, with some additions:

* Some new channels have been added.
* In this file is specified that Mote 2 will wake up after 5 seconds.

**Log.txt** file contains an example of the execution of the simulation.

# Achieved results

From the Log.txt file it is possible to note that the mote 1 will start sending request messages after about 1 seconds from the beginning of the simulation, until the mote 2 wakes up. When the mote 2 wakes up [DEBUG (2): Radio on!] it catches a request and immediately answer with a reply message containing the data of the sensor and sending the Ack right away. When the mote 1 receives the Ack, he stops sending further requests.

Oss: From the log it’s clear that, even after the mote 1 has received an ack, the mote 2 still receives (and answer) requests of mote 1. This might happen because probably, while the mote 2 is preparing and sending the ack, some new request could have been sent from mote 1 (since the timer has fired again). In this way the mote 2 will still receive some requests even if mote 1 have received an Ack. In any case, those requests are just a few (1 or 2) and in our opinion they does not represent a real problem.