IoT Projects (MILANO) - 2017/2018

Daniele Moro
daniele.moro@polimi.it

May 7, 2018

1 General Rules, Grading and Deadlines

Grade composition of the entire course:

- 26 out of 30 points are assigned based on the written exams
- up to 8α out of 30 points are assigned based on projects (parameter α depends on project delivery date, see below).

General rules:

- Projects are NOT mandatory. One student can decide not to take any project; in this case, that maximum grade he/she can get will be 26/30.
- Projects can be developed in groups of maximum 2 people.

Projects delivery deadlines and grading:

- September 10, 2018: $\alpha = 1$, this means that if you deliver your project by September you can get the full 8 points
- December 31, 2018: $\alpha = 0.5$, this means that if your deliver the project after September 10, 2018, but before December 31, 2018 you can get up to 4 points
- after December 31, 2018: $\alpha = 0$, this means that you don't get any additional points after this deadline.

The students willing to take the project assignment must choose among the project proposals listed in the following section or propose an original project, using the online form available at `https://goo.gl/forms/07vyIYJTDlqzeZNY2` by **May 20, 2018**.

IMPORTANT 1: ONLY THE PROJECTS REGISTERED ON THE ONLINE FORM WILL BE CONSIDERED. LATE REGISTRATION WILL NOT BE ACCEPTED.

IMPORTANT 2: REGISTRATION IS NOT BINDING. IF YOU REGISTER FOR A PROJECT AND THEN DECIDE AFTERWARDS YOU DON'T WANT TO DELIVER IT, THAT'S OK.

For original projects, write to `daniele.moro@polimi.it` describing the project you intend to implement.

2 Proposed Projects

The following projects proposal are thought for being implemented with the tools seen during the hands-on lectures (TinyOS, Contiki, Moterunner, RIoT, Node-Red, etc..). For projects related to WSN software programming, you are free to choose which O.S. to use, although TinyOS is probably simplest way.

You have to deliver the following items:

- Complete source code of the project
- Self-explanatory log file, showing that your project works. Try to be as detailed as possible when preparing the log file (i.e., use debug/print statements in all the crucial phases of your project)
- Project report (max. 3 pages), which summarizes your approach in solving the problem, including figures when needed. Don't include source code in the project report.

Projects will be evaluated based on the rationale and technical depth of the design choices, correctness of the source code and organization and clarity of the project report.

2.1 Project 1. Ad Hoc On-Demand Distance Vector routing protocol - up to 8 points

You are requested to design and implement a protocol similar to AODV (`https://tools.ietf.org/html/rfc3561`) and test it with simulations on the topology represented in Figure 1

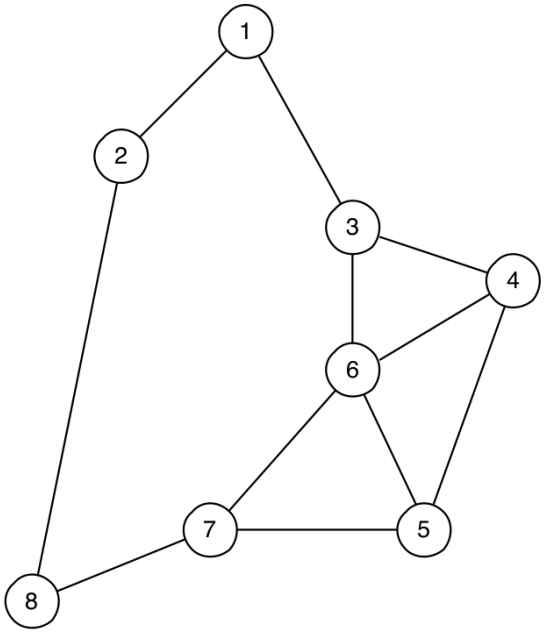

Figure 1: Network topology.

The following features need to be implemented:

1. Each node generates a DATA message for a random destination every 30 seconds. The payload of each DATA message must contain the final destination address and some random data

2. Before DATA message transmission, each node checks its routing table to see if a route is present for the selected destination. If the destination is present, the message is forwarded to the next hop indicated in the routing table. Otherwise a ROUTE REQ message is broadcast, containing the selected destination.

3. A node receiving a ROUTE REQ message should

    - broadcast it if the ROUTE REQ is a new one (i.e. coming from a node for a new destination)
    - discard it if it is a duplicated ROUTE REQ. A node can discover duplicates by checking the ROUTE REQ ID (which should be contained in the message), its source node and its destination and comparing such information with the one stored on board
    - reply with a ROUTE REPLY, if it is the destination of the ROUTE REQ. The ROUTE REPLY should be transmitted to all nodes who forwarded the ROUTE REQ to the destination and in turn back to the original destination.

4. after transmitting the ROUTE REQ message, the source node waits for 1 second ROUTE REPLY messages coming from all routes available. Each ROUTE REPLY must carry the accumulated number of nodes that it passes going back, and the source node must select the route with the minimum number of hops. ROUTE REPLY messages received after 1 second from the transmission of the ROUTE REQ must be discarded.

5. After that, the source node finally transmits a DATA message, which contains random data.

6. Each node needs to have a routing table which stores for each possible destination what is the next hop to which messages should be forwarded. A timer also controls the validity of each entry which becomes invalid after 90 seconds from their creation.

2.2 Project 2. Smart Lights WSN - up to 6 points

Implement a network of smart lights, then program the network to visualize luminous pattern inside the predefined topology. Assign a fixed address to each node (e.g., use TOS_NODE_ID when using TinyOS), the routing will be defined a priori as described below.

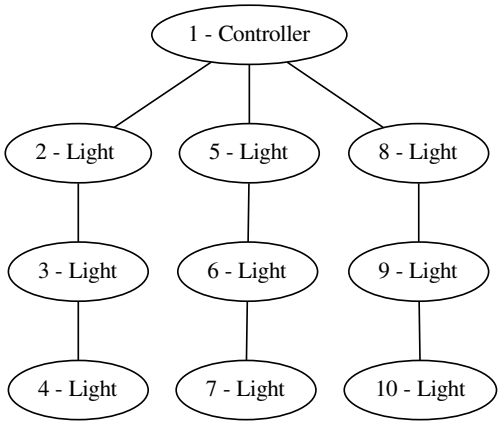

Figure 2: Network topology.

The requirements of this project are:

1. Create a topology with at least 10 nodes (1 controller node and 9 light nodes), as illustrated in Figure 2.

2. Addressing: use the same addressing idea as illustrated in Figure 2, for example controller will have the address 1 etc. You can modify addressing but they must always have a hierarchy to simplify the routing phase.

3. Routing: each node will have its own pre-defined routing table. Multihop communication has to be considered (only nodes 2, 5, 8 are in the transmission range of the controller). The controller node knows that messages addressed to node from 2 to 4 have to be sent toward node 2, messages to node from 5 to 7 toward node 5 etc. Light nodes instead

have to forward packets not addressed to themselves. For example when node with address N receives messages addressed to nodes with address greater than its own it has to forward them to node N+1, while messages addressed to nodes with address lower than its own have to be forwarded to node N-1. Remember to take care of not re-forward messages to the same node from which the message is received.

4. The controller node is programmed with some pre-defined pattern to switch on the smart lights, it periodically sends unicast commands to the lights to switch them on and off.

5. Show at least 3 patterns (e.g. a cross switching on node 2,4,6,8,10; a triangle switching on node 6,4,7,10) that cyclically change. Remember that from one time to the other all the nodes have to be switched off before switching them on (it is ok to send a switch off message to every node instead of using broadcast packets).

You are free to use a different topology or different patterns from the proposed ones. Simulate the program in Cooja showing the LEDs on the nodes.

2.3 Project 3. Data collection with Thingspeak - up to 4 points

In this project, you are required to implement a system for data collection using TinyOS, Node-RED and Thingspeak.

The requirements of this project are:

1. Create TWO simulated WSN, each one with a sink node and two sensor nodes (one equipped with a temperature sensor and the other with a humidity sensor, you can use the provided `TempHumSensorC.nc` component, whose operation is demonstrated in the `SensorTestAppC.nc` application). Simulate the two WSNs in Cooja attaching EACH sink node to a Node-RED socket (use the SERVER serial socket tool in Cooja). Data should now be available in Node-RED.

2. Use the functionalities provided by Node-RED to transmit the data from the simulation to ThingSpeak. Each WSN should be linked to a different channel, and each sensor node in each WSN should be logged on a different Field.

3. Use Node-Red functions to read the data from ThingSpeak and write a function to send an alert email when the average value of the Temperature from the two WSN exceed a predefined threshold.