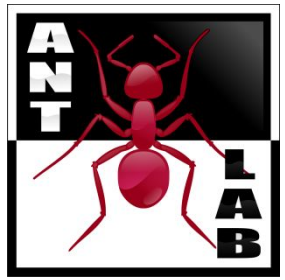
Politecnico di Milano

Advanced **N**etwork **T**echnologies **Lab**oratory



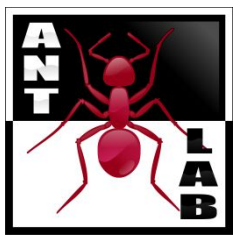
Internet of Things

Projects

General Rules

- Grading
  - 26/30 are assigned based on the written exam
  - 8α/30 are assigned based on the project
- Projects delivery deadlines
  - Before *Sept. 10, 2018*→**α=1**
  - After *Sept. 10*, but before *Dec. 31, 2018*→**α=0.5**
  - After *Dec. 31, 2018*→**α=0**
- Which project to implement, which OS?
  - Your decision (TinyOs is recommended)

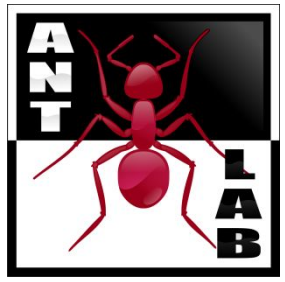
What to deliver



- Complete source code of the project
- Self-explanatory log file
  - It should show that your project works
  - Try to be as detailed as possible!
- Project Report
  - Max. 3 pages, summarizing your approach
  - Include figures, if needed
  - Don’t include source code!

Project registration

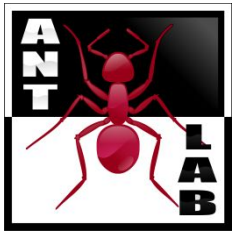
- Register to this google form: https://goo.gl/forms/07vyIYJTDlqzeZNY2
- Registrations are open until
  May 20, 2018
- **IMPORTANT: Only the projects registered on the online form will be considered. Late registration will not be accepted.**
- Registration is not binding, if you register for a project and then decide afterwards you don't want to deliver it, that's OK



1. Ad Hoc On-Demand Distance Vector routing protocol

Up to 8 points

Distance Vector



- Ad-Hoc On-demand Distance Vector: build distance vector routes on-demand

- Implement something similar to ZigBee AODV routing algorithm

Protocol specs

- Each node generates **DATA** message for a random destination.
  Payload: final destination + random data.
- Before DATA transmission: check ***routing table*** to see if a route is present for the selected destination.
  If no route present: **ROUTE_REQ**, broadcast, it contains the selected destination

ROUTE_REQ

- When receiving a ROUTE_REQ:
  - broadcast if it is a new one (e.g., new destination)
  - duplicate ROUTE_REQ are discarded (check
  - ROUTE_REQ_ID, source node, destination node).
  - reply with **ROUTE_REPLY** if it is the destination of the ROUTE_REQ.
    ROUTE_REPLY should be transmitted to all nodes who forwarded the ROUTE_REQ to the destination and in turn, back to the original destination (as *unicast*)

Protocol specs - cont.

- After transmitting ROUTE_REQ:
  - Wait 1 second for the ROUTE_REPLY.
  - You can receive ***multiple*** ROUTE_REPLY.
  - ROUTE_REPLY must carry the ***number of nodes*** that it passes going back.
  - The node has to select the *route with the **minimum number** of hops*.
  - ROUTE_REPLY received after 1 second from the transmission of the ROUTE_REQ must be discarded.

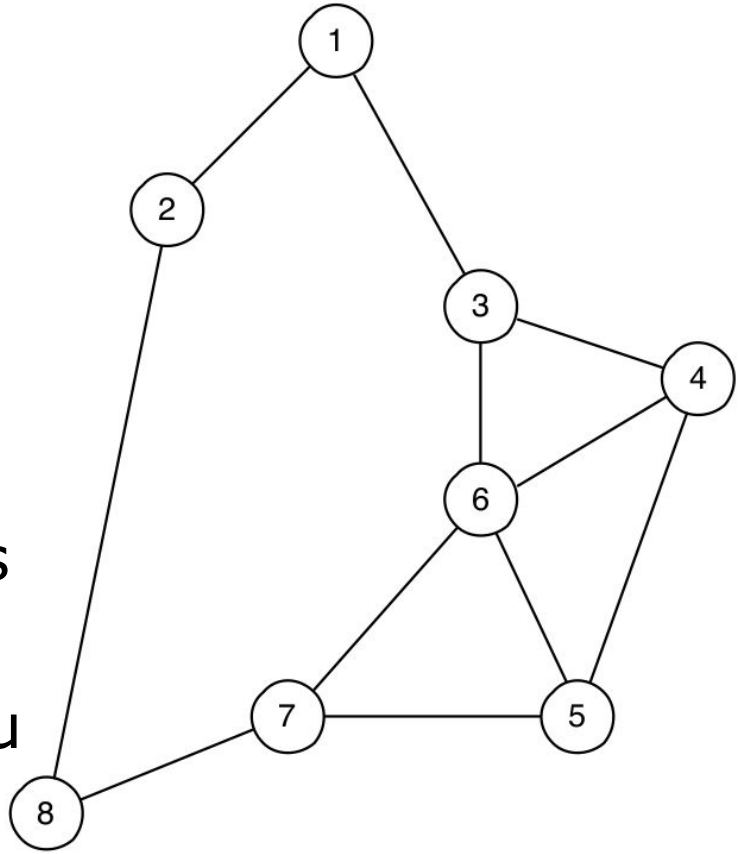
Protocol specs - cont.

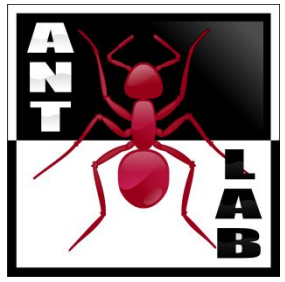
- After that the nodes can finally ***transmit*** the DATA message
- Each node stores its ***routing table***.
  Routing table stores, for each possible destination, what is the *next hop* to which messages should be forwarded.
  Each entry of the routing table *becomes invalid after* ***90 seconds*** from their creation.

Destination	Next Hop
0	1
3	2

Test Topology

- Test the proposed algorithm with the topology presented here.
- Test should be performed in TOSSIM or Cooja.
- Remember that with cooja you'll need to move the nodes to define a topology as in the image, while with TOSSIM you need to define the topology file





2. Smart Lights WSN

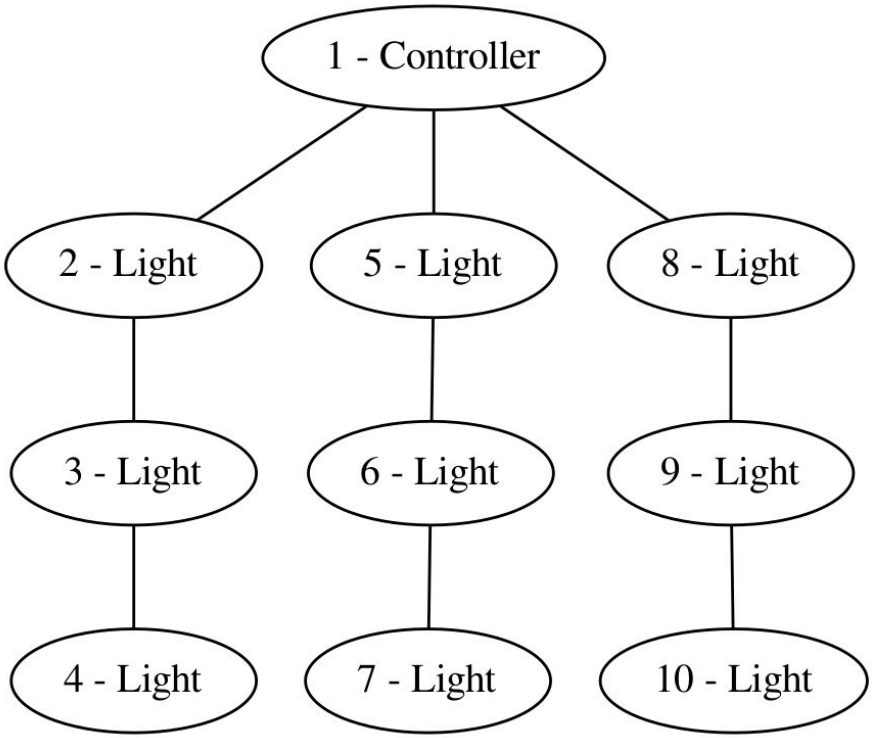
Up to 6 points

Specification

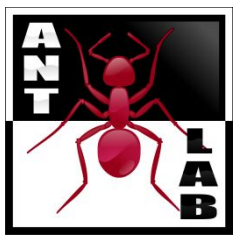
- Network of smart lights, at least:
  - 1 lights controller
  - 9 lights node
- Visualize luminous patterns
- Messages that trigger the switching ON and OFF of the LEDs sent as unicast message from the *controller* node.
- Messages must contain the action to be performed (switch ON/OFF) and destination
- Patterns are pre-configured into the controller node

Topology

- Use the proposed topology
- You are free to use a different topology
- Multi-hop communication must be used



Addressing



- Based on a hierarchy to simplify routing
- Controller has lowest node ID
- Each node has a fixed address (e.g. TOS_NODE_ID)

- See topology to have an example of addressing

Routing - 1

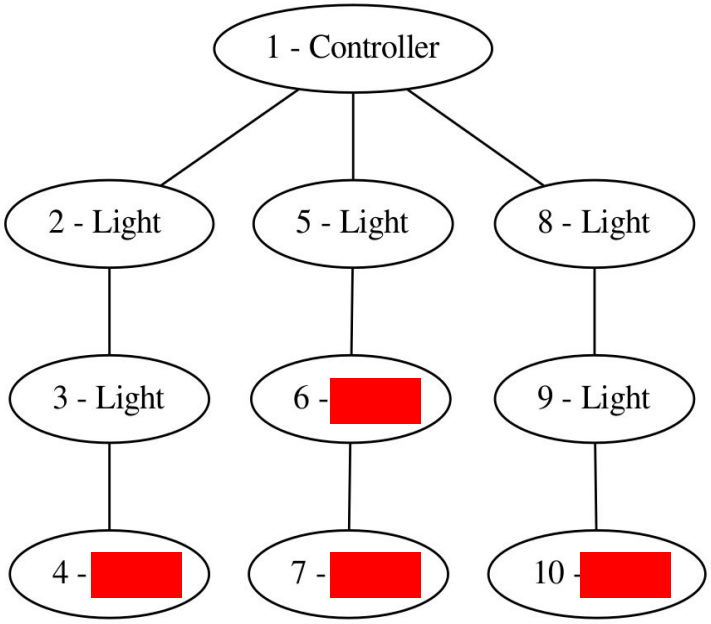
- Fixed routing table defined a priori
- Based on the proposed topology:
  - Controller:
    - messages to nodes 2-4:
      forward messages to node 2
    - messages to nodes 5-7:
      forward messages to node 5
    - messages to nodes 8-10
      forward messages to node 8

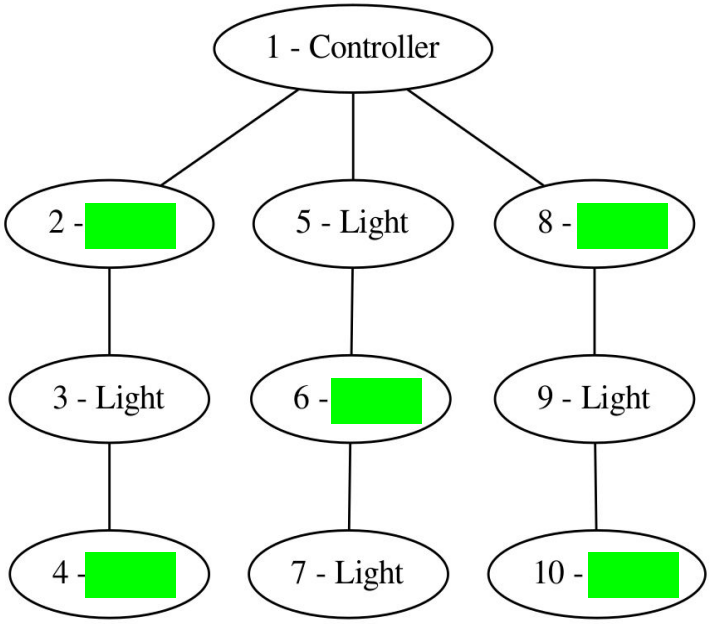
Routing - 2

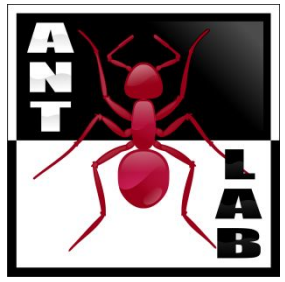
- Light Nodes:
  - forward messages not addressed to themselves.
  - Node N receives a message with destination K:
    - If K=N: read message and act on LEDs
    - If K>N: forward message to node N+1
    - If K<N: forward message to node N-1

Pattern

- Feel free to use different patterns
- Show at least **3 patterns** that *cyclically change*.
- Remember to switch off the nodes from one pattern to the other



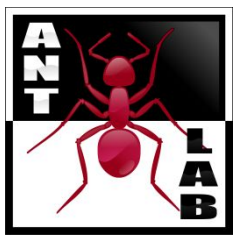




3. Data collection with Thingspeak

Up to 4 points

Specification



- Implement a system for data collection using TinyOS, Node-RED and Thingspeak

Requirements - 1

- Implement TWO simulated WSN
  Each one with a sink node and two sensor nodes (one with a temperature sensor and the other with a humidity sensor, use the provided TempHumSensorC.nc component).
- Simulate the two WSNs in Cooja attaching EACH sink node to a Node-RED socket (server socket tool of Cooja).

Requirements - 2

- From Node-RED upload data to ThingSpeak
- Each WSN linked to a different channel
- Each sensor logged in a different field

Requirements - 3

- Finally 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

Report

- Please include screens of ThingSpeak and Node-RED into the technical report to show that everything is working correctly.