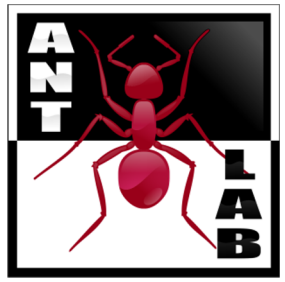
Politecnico di Milano

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



Home Challenge #4

Simulate a Wireless Sensor Network with TOSSIM

Home project #4

- Develop a TinyOS application
- Simulate the application with TOSSIM

- Team: max 2 people
- Score: max 1 point
- **Deadline**: May 16th – 23:69

- Submit through webeep «Challenge 4» folder

- File name:

<personal_code1>_<personal_code2>.zip

Challenge deliverables

- Form: https://forms.office.com/r/1CXDaMAvbP
  - Only one entry for group
- Zip Content:
  - All the source code (TinyOS files, python file, topology, noise, etc…)
  - Complete log of the simulation
  - Short report (max 1 page)
  - Your names + ID number on top of the report
  - Repository link (if used)

Send/ACK example

#1

REQUEST X

REQUEST X

REQUEST X

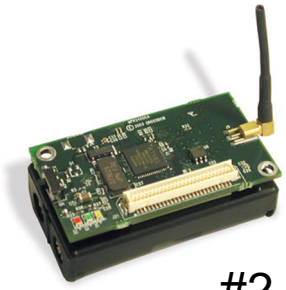
REQUEST

MOTE #2 BOOT

ACK

RESPONSE
(Sensor Read)

ACK



#2

What to do

- Simulate 2 motes talking between each others
- Mote #1 sends periodic request (REQ) messages to mote #2 containing:
  - Message type: REQ
  - An incremental counter
- The request has periodicity 1000ms

What to do

- Only on receipt of a request, mote #2 sends back a reply (RESP) message with:
  - Message type: RESP
  - The counter sent by mote #1
  - A value read from the fake sensor
- Fake sensor is just a module which return a random number, you don't need to modify it

What to do

- Each message, REQ and RESP, must be acknowledged using the TinyOS built in ACK module
- Upon receipt of the X^{th} REQ-ACK message:
  - Mote #1 stops to send requests
  - The exercise is done!
- Use the module `PacketAcknowledgements` to send the ACK, don't reimplement it!

Send/ACK example

#1

REQUEST → X

REQUEST → X

REQUEST → X

…

REQUEST →

← 1st REQ-ACK

…

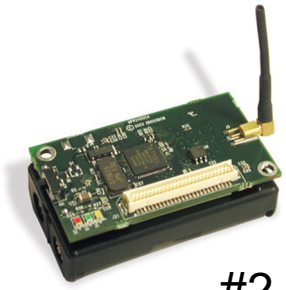
REQUEST →

← Xth REQ-ACK

← RESPONSE (Sensor Read)

RESP-ACK →

MOTE #2 BOOT (at time Y)



#2

DONE!

Parameters

- X = [last digit of your (team 1) person code] + 1
- Y = person code without:
  - last three digits
  - first three digits

- 10410164
  - X = (4+1) = 5
  - Y = 10

PacketAcknowledgements

- Read the documentation
- Available at `/home/user/Desktop/tinyos-main/doc/nesdoc/telosb/index.html`

Commands
command error_t noAck(message_t *msg) Tell a protocol that when it sends this packet, it should not use synchronous acknowledgments.
command error_t requestAck(message_t *msg) Tell a protocol that when it sends this packet, it should use synchronous acknowledgments.
command bool wasAcked(message_t *msg) Tell a caller whether or not a transmitted packet was acknowledged.

Template

- In the folder SendACK_template there's a draft of the code to fill in
- Try to use as much as possible that draft
- Files to modify:
  - SendAck.h
  - SendAckAppC.nc
  - SendAckC.nc

```
//********************* AMSend interface ****************//
event void AMSend.sendDone(message_t* buf,error_t err) {
  /*  This event is triggered when a message is sent... */
  /*
   * STEPS:
   * 1. Check if the packet is sent
   * 2. Check if ack is received (Read the docs!)
   * 2a. If yes stop the timer, the program is done
   * 2b. Otherwise: send again a request
   * Always: use debug statements
   */

}
```

Simulation

- Simulate it with TOSSIM
  - Mote #1 at time 0
  - Mote #2 after Y seconds

Message structure

- Only one message type containing:
  - `msg_type`: REQ/RESP
  - `msg_counter`: incremental integer
  - `value`: value from the fake sensor

Hints

1) Create the message structure
2) Choose which modules you'll use
3) Write the modules in the …AppC.n and in the …C.nc files
4) Wire the modules in the …AppC.nc file
5) Implement the logic in the …C.nc file

Hints

- Run the code often to check syntax or compiler errors, not only at the end!
- Use a lot of debug statements (DBG/DBG_CLEAR)
- Read the docs
- Use the examples seen before as scheme
- Think!

Commands

- Compile the mote's code
  - `make micaz sim`
- Run the simulation
  - `python RunSimulationScript.py`

- Before run a simulation, recompile the code of the mote if you have modified it