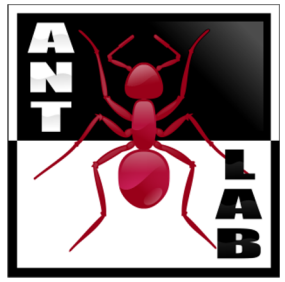
Politecnico di Milano

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



TinyOS SIMulator

Simulate a Wireless Sensor Network with TOSSIM

Motivations

- WSN require large scale deployment
- Located in inaccessible places
- Apps are deployed only once during network lifetime
- Little room to re-deploy on errors

System evaluation

- Check correctness of application behavior
- Sensors are hard to debug!
  - "… prepare to a painful experience" [Tinyos authors' own words]

Simulation Pros and Cons

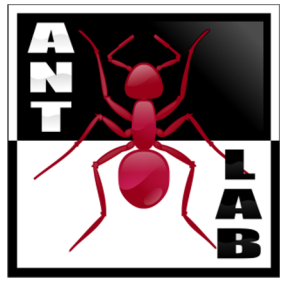
- Advantages
  - Study system in controlled environment
  - Observe interactions difficult to capture live
  - Helps in design improvement
  - Cost effective alternative
- Disadvantages
  - May not represent accurate real-world results
  - Depends on modeling assumptions

General concepts

- TOSSIM is a discrete event simulator
- It uses the same code that you use to program the sensors
- There are two programming interfaces supported: Python and C++

Key requirements

- Scalability
  - Large deployments (10^3 motes)
- Completeness
  - Cover as many interactions as possible
  - Simulate complete applications
- Fidelity
  - Capture real-world interactions
  - Reveal unexpected behavior
- Bridging
  - Between algorithm and implementation



Exercise

Temperature and humidity sensor

Fake temp/hum sensor

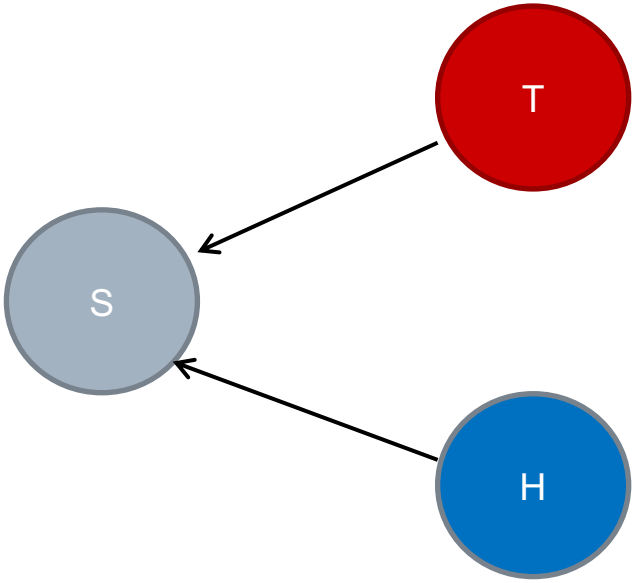
- Starting from the example in:

`IOT-examples/TinyOS/Supporting\ Files/TempHumSensor`

Simulate with TOSSIM a TinyOS application that:

- generate 3 motes (0, 1, 2)
- mote #0 is the sink
- mote #1 is the temperature sensor
- mote #2 is the humidity sensor

Topology



Fake temp/hum sensor

- Mote #0 only receive messages from #1 and #2
- Mote #1 sends periodic (every 1 second) messages to the sink (#0)
- Mote #2 sends periodic (every 2 seconds) messages to the sink (#0)

Messages

- Messages are composed like this:

- type: 0 or 1 (0 for temp and 1 for hum)

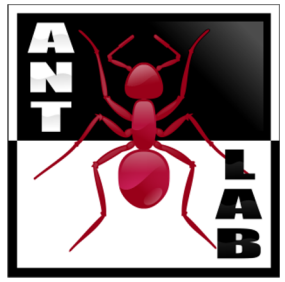
- data: the value from the fake sensor

Fake temp/hum sensor

- Let’s do it together!

Politecnico di Milano

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



Home Challenge #2

Simulate a Wireless Sensor Network with TOSSIM

Home project #2

- Develop a TinyOS application
- Simulate the application with TOSSIM

- Team: max 3 people
- Score: max 1 point

- Submit through beep «Homework/Activity 2» folder
- Don’t cheat ☺

What to deliver

- A zip folder per group named with:
  - A folder containing all the source code (TinyOS files, python file)
  - The log of your simulation
  - A short report

- Deadline: March 29 – 23:49

Send/ACK example

#1

REQUEST X

REQUEST X

REQUEST X

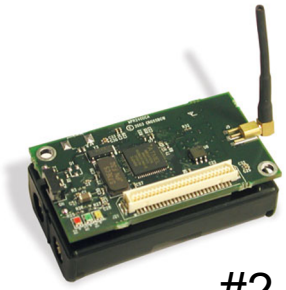
REQUEST

MOTE #2 BOOT

ACK

RESPONSE
(Sensor Read)

ACK



#2

What to do

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

What to do

- Only on receipt of a request, mote #2 sends back a reply (RESP) message with:
  - Message type: REQ
  - 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 ACK of the a REQ message:
  - Mote #1 stops to send requests
  - Mote #2 receive the RESP
  - 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

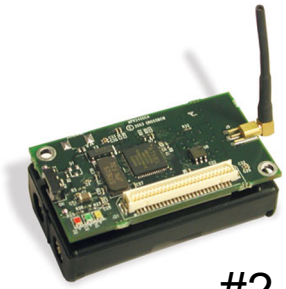
REQUEST →

MOTE #2 BOOT
(at time 5)

← ACK

← RESPONSE
(Sensor Read)

ACK →



#2

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

- Simulation is done in TOSSIM
  - Mote #1 at time 0
  - Mote #2 after 5 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 mote's code if you have modified it