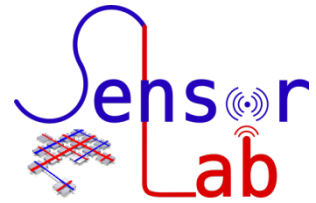




GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN



Georg-August-Universität Göttingen
Faculty of Mathematics and Computer Science
Institute of Computer Science
Telematics Group
Sensorlab

Lab 3 –Simulation in TinyOS with TOSSIM

Chencheng Liang – University of Goettingen – 11603960

Introduction

I did this lab in my own computer using virtual machine with xubuntuOS .

TOSSIM (TinyOS SIMulator) can simulate the sensor nodes, and can be controlled by Python or C++, in this lab we use Python because it is simple to use. Different from previous compile, if we use TOSSIM, we use the command "make iris sim" to compile it. In python, with dir(object) command we can see what functions we can use.

This time we want to know the debug information from the node, so we need to create channels in the simulation to get the information be printed on the screen, so we can see our debug information in the node. For example, in sender node, it can send the information about simulation time, sensor id, and these information can be print on screen from the channel that we defined.

In assignment 3.1, I create 3 nodes, and they broadcast information(simulation time and sensor id), and receive information for each other. The simulator will gather these information from the channels which defined in Python and the program in the nodes as well.

In assignment 3.2, I use different noise trace to interfere the transmission, if the interfere is strong then there is no package received. If the interfere is not so string, then some packages received, and some packages are lost, which will be verified by the counter.

Assignment 3.3 explains the receive code in RadioCountToLedsC.nc.

Assignment 3.1

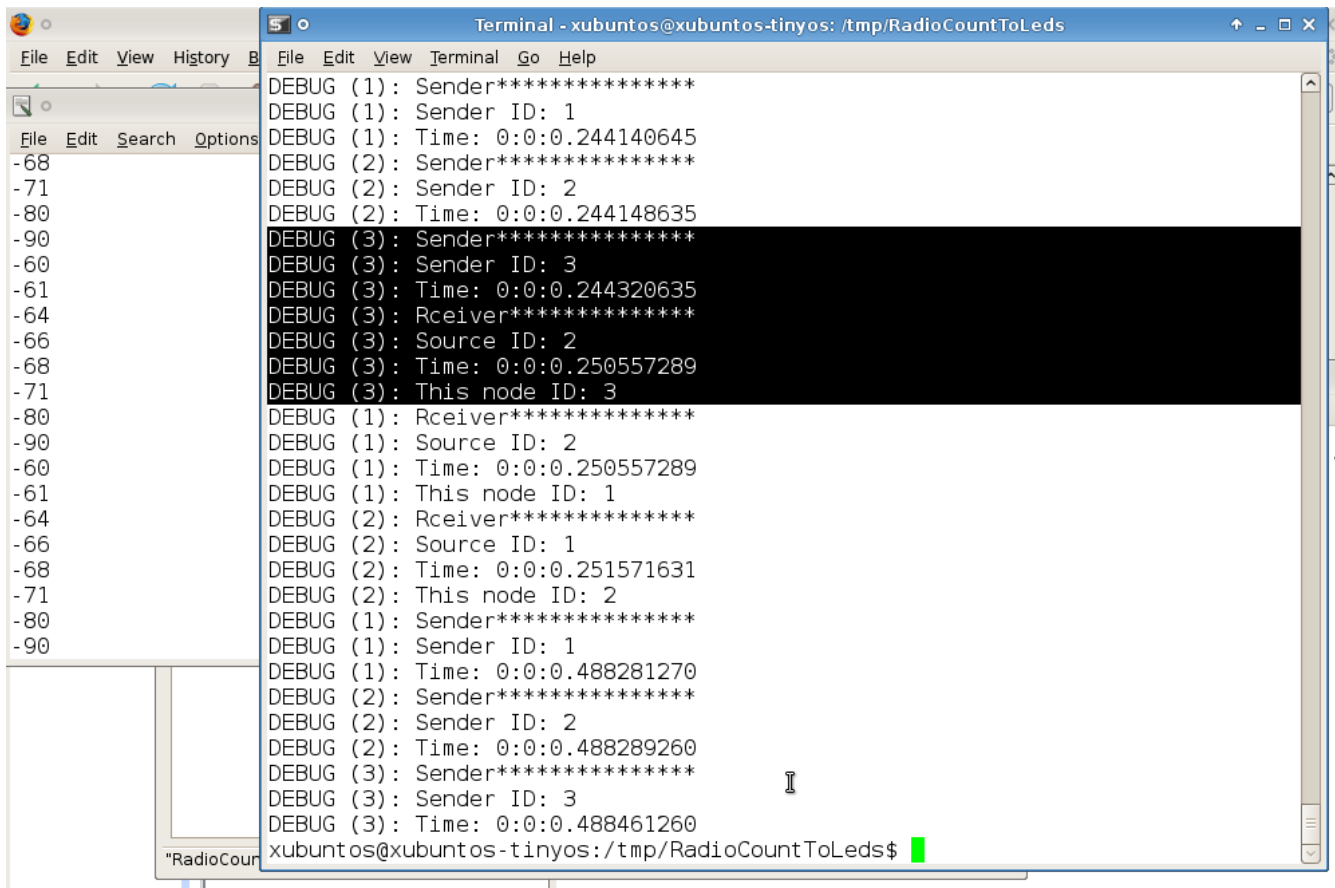
In RadioCountToLedsAppC.nc, add wires(App.Packet -> AMSenderC and App.AMPacket -> AMSenderC;) to load the package and get the address of current node and the source node.

In RadioCountToLedsC.nc, under MilliTimer.fired() add "uint16_t Sender_ID = call AMPacket.address() " to get the sender node id, and print it on RadioCountToLedsC channel. we can use sim_time_string() to get the simulation time.

Under the Receive.receive(), we receive the message. we call AMPacket.source() to get the source node id, and we call AMPacket.address() to get the receiver node id. Then print them on the screen by dbg() in RadioCountToLedsC channel. Also get the simulation time by sim_time_string().

Use " radio_count_msg_t* rcm = (radio_count_msg_t*)payload" to get the message out of the payload, then we can get the counter by "rcm->counter" which is defined in the head file (RadioCountToLeds.h).

The result is as following:



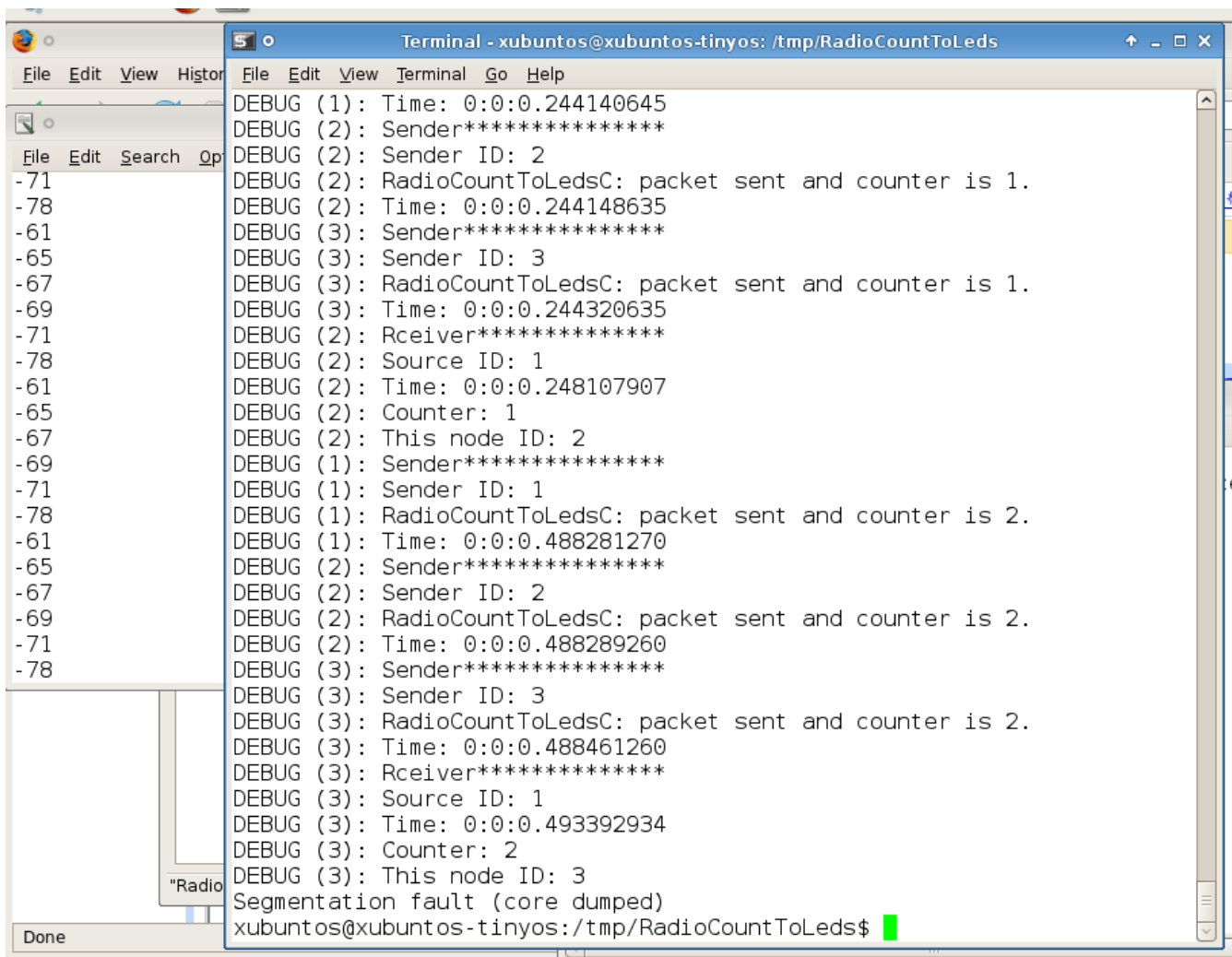
```
Terminal - xubuntos@xubuntos-tinyos: /tmp/RadioCountToLeds
DEBUG (1): Sender*****
DEBUG (1): Sender ID: 1
DEBUG (1): Time: 0:0:0.244140645
DEBUG (2): Sender*****
DEBUG (2): Sender ID: 2
DEBUG (2): Time: 0:0:0.244148635
DEBUG (3): Sender*****
DEBUG (3): Sender ID: 3
DEBUG (3): Time: 0:0:0.244320635
DEBUG (3): Rceiver*****
DEBUG (3): Source ID: 2
DEBUG (3): Time: 0:0:0.250557289
DEBUG (3): This node ID: 3
DEBUG (1): Rceiver*****
DEBUG (1): Source ID: 2
DEBUG (1): Time: 0:0:0.250557289
DEBUG (1): This node ID: 1
DEBUG (2): Rceiver*****
DEBUG (2): Source ID: 1
DEBUG (2): Time: 0:0:0.251571631
DEBUG (2): This node ID: 2
DEBUG (1): Sender*****
DEBUG (1): Sender ID: 1
DEBUG (1): Time: 0:0:0.488281270
DEBUG (2): Sender*****
DEBUG (2): Sender ID: 2
DEBUG (2): Time: 0:0:0.488289260
DEBUG (3): Sender*****
DEBUG (3): Sender ID: 3
DEBUG (3): Time: 0:0:0.488461260
xubuntos@xubuntos-tinyos: /tmp/RadioCountToLeds$
```

Assignment 3.2

By changing the content in file meyer-heavy.txt to over -200, we can see that there is no message received:

```
File Edit View History B File Edit View Terminal Go Help
3 1 -60.0
2 3 -64.0
3 2 -64.0
-200 Creating noise model for 1
-201 Creating noise model for 2
-204 Creating noise model for 3
-207 DEBUG (1): Application booted.
-200 DEBUG (1): Application booted.
-201 DEBUG (1): Application booted (second message).
-204 DEBUG (1): Application booted (third message).
-207 DEBUG (2): Application booted.
-200 DEBUG (2): Application booted.
-201 DEBUG (2): Application booted (second message).
-204 DEBUG (2): Application booted (third message).
-207 DEBUG (3): Application booted.
-200 DEBUG (3): Application booted.
-201 DEBUG (3): Application booted (second message).
-204 DEBUG (3): Application booted (third message).
-207 DEBUG (1): Sender*****
-200 DEBUG (1): Sender ID: 1
-201 DEBUG (1): RadioCountToLedsC: packet sent and counter is 1.
-204 DEBUG (1): Time: 0:0:0.244140645
-207 DEBUG (2): Sender*****
DEBUG (2): Sender ID: 2
DEBUG (2): RadioCountToLedsC: packet sent and counter is 1.
DEBUG (2): Time: 0:0:0.244148635
DEBUG (3): Sender*****
DEBUG (3): Sender ID: 3
DEBUG (3): RadioCountToLedsC: packet sent and counter is 1.
DEBUG (3): Time: 0:0:0.244320635
xubuntos@xubuntos-tinyos:/tmp/RadioCountToLeds$
```

By changing that around -60, we can see the counter shows some packages are lost. For example there are 3 senders, the counter is 2, and there are only two receiver shows that its counter is 2.



```
Terminal - xubuntos@xubuntos-tinyos: /tmp/RadioCountToLeds
DEBUG (1): Time: 0:0:0.244140645
DEBUG (2): Sender*****
DEBUG (2): Sender ID: 2
DEBUG (2): RadioCountToLedsC: packet sent and counter is 1.
DEBUG (2): Time: 0:0:0.244148635
DEBUG (3): Sender*****
DEBUG (3): Sender ID: 3
DEBUG (3): RadioCountToLedsC: packet sent and counter is 1.
DEBUG (3): Time: 0:0:0.244320635
DEBUG (2): Rceiver*****
DEBUG (2): Source ID: 1
DEBUG (2): Time: 0:0:0.248107907
DEBUG (2): Counter: 1
DEBUG (2): This node ID: 2
DEBUG (1): Sender*****
DEBUG (1): Sender ID: 1
DEBUG (1): RadioCountToLedsC: packet sent and counter is 2.
DEBUG (1): Time: 0:0:0.488281270
DEBUG (2): Sender*****
DEBUG (2): Sender ID: 2
DEBUG (2): RadioCountToLedsC: packet sent and counter is 2.
DEBUG (2): Time: 0:0:0.488289260
DEBUG (3): Sender*****
DEBUG (3): Sender ID: 3
DEBUG (3): RadioCountToLedsC: packet sent and counter is 2.
DEBUG (3): Time: 0:0:0.488461260
DEBUG (3): Rceiver*****
DEBUG (3): Source ID: 1
DEBUG (3): Time: 0:0:0.493392934
DEBUG (3): Counter: 2
DEBUG (3): This node ID: 3
Segmentation fault (core dumped)
xubuntos@xubuntos-tinyos: /tmp/RadioCountToLeds$
```

Assignment 3.3

The main modify is add receive event to make the node to receive the message by adding " event message_t* Receive.receive(message_t* bufPtr, void* payload, uint8_t len) " to the code. And use AMPacket.address() to get current node id, use AMPacket.source() to get the source node id from the message.

Other modification has been explained in Assignment 3.1.

Lessons Learned & Conclusion

In this lesson, I learned what is TOSSIM (TinyOS Simulator), and I can use it to simulate the devices. by doing so, I can test the programming in the simulator first, then to install it to the node, which is super time-saving. And it can simulate many nodes at one time.

To use TOSSIM, first need to use command "make iris sim " to compile it. Then use python to control it to start. It is very simple to use python control the simulation, first we import TOSSIM by command " from TOSSIM import *", then use command " t = Tossim([])" to create a TOSSIM object. We can simulate one node by get the node from command " m = t.getNode(32)" and command " m.bootAtTime(45654)" to boot that node at time 45654. And call command " t.runNextEvent()" to run the simulation.

For getting debug information, we need to create some channels in the python code, and import the sys package in Python. And in the nesC code use dbg function to print the information, and the channel must be matched with the python code. The debug information will show from which node this information comes.

I also learned in radio propagation model, TOSSIM also simulates the RF noise and interference a node hears, both from other nodes as well as outside sources by Closest Pattern Matching (CPM) algorithm.. We can get the noise trace from a file, then use addNoiseTraceReading() to read it and generate the noise model by createNoiseModel(). For the noise trace, if it is too long, it consume more node memory, if it is too short it reduce the fidelity of simulation.

Even we can add the nodes to the topology one by one, but it is wiser to create a script to specify the topology in a file, such as topo.txt, then use python to read it and create the topology.

We can store all the python command in a batch file, then execute them in once.

The greatest thing for me is I can use TOSSIM in my own xubuntuOS in virtual machine, by to do so, I do not need to go to lab then. That is very convenient to me!

References