University of Maryland Baltimore County
Department of Computer Science and Electrical Engineering

CMPE 684, Wireless Sensor Networks

Assignment #3 Due: Thu 11/18/2013 in the class

Objective:

To experience the development environment for sensor network applications and gain hands-on experience by simulating the network operation. In this homework you are to get familiar with TinyOS and its programming language, nesC, and its simulation environment TOSSIM.

Statement:

This assignment requires developing an application using TinyOS and using TOSSIM to conduct simulation of a network of sensor nodes. Please refer to "TinyOS and TOSSIM Development Environment" document that has been provided by TA for setting up the TinyOS development environment. Also further information about TinyOS, nesC and TOSSIM is available via http://www.tinyos.net/.

The TA has prepared a simulation for a wireless sensor network consisting of 27 nodes using TOSSIM. You should download this simulation environment by following the instruction provided below. In the simulation, Node with ID 0(zero) is programmed to be the sink (Base-station; BS) for the data generated by sensors. Each sensor periodically generates a packet that gets relayed over a multi-hop route towards the BS (node with ID 0). The BS reports statistics about the network periodically; e.g., number of received packets, and average delay of packet delivery.

The provided implementation reflects a normal network operation scenario without considering the possibility of having malicious nodes in the network. Your task is mainly to extend the implementation so that any node could be configured to act maliciously by undertaking attacks to degrade the quality of the service (QoS) in the network. Packet delay, drop and injection are the types of QoS attacks that you need to implement.

How to get and run TA provided code:

TA provided development environment and tutorial is available from following link:

https://www.dropbox.com/sh/auoarijtw75e5hu/utJxWq-WnC

Please download:

1- "TinyOS and TOSSIM Development Environment" document
2- Ubuntu_TinyOS.vdi
3- VirtualBox-4.2.18-88781-Win.exe
4- HW3_WSN_Provided.zip

Follow the instructions in "TinyOS and TOSSIM Development Environment" document to get yourself familiar with the basics and setup the development environment. Pay specially attention to TOSSIM section. Run the code by issuing following commands:

```
make micaz sim
./Simulate.py
```

The following files are included in the "HW3_WSN_Provided.zip" file:

1- *HW3C.nc*: main file containing HW3C module and its implementation. (You will do your changes here)
2- *HW3APPC.nc*: configuration module to wire HW3C.nc to TinyOS components. (no need to change)
3- *Makefile*: to compile main modules and required packet components for the project (no need to change)
4- *Simulate.py*: a python program to run the TOSSIM simulation
   i. You might add, remove, comment dbg (print) statement channels defined in this file
   ii. You might need to increase/decrease simulation time for your own tests
5- *topology.txt*: defines communication links among sensor nodes. For simplicity we have defined same gain value for all connections. Note: Routing can only use a valid connection. There is no need to change this file.
6- *Noise_short.txt*: defines a noise model for radio layer. There is no need to change this file.
7- *Malicious_list.txt*: Sample list of malicious nodes and their QoS attack type. You will change this to test your implementation. Note: this file will be changed when grading your code; in other words do not think this is a fixed file and hardcode anything.
8- QOS_Attack function in HW33C.nc reads the malicious_list.txt file and returns a value indicating the QOS attack assigned to a node.
9- Static routing algorithm has been applied; Function "uint8_t GetMyParent(uint8_t)" from file HW3C.nc does the static routing

What you will do: You will code to provide following features:

1- Changes to packet structure so it will record the route a packet takes until it reaches the BS.
2- Implement packet drop, delay and injection attacks as follows:
   i. Add following enum to the header file

```
enum {
      QOS_NORMAL = 0,//node acts normal
      QOS_DROP = 1,
      QOS_DELAY = 2,
      QOS_INJECT = 3
};
```

   ii. *QOS_DROP*: node only will drop packets of other nodes relayed to it
   iii. *QOS_DELAY*: node will delay forwarding packets that has been relayed to it. Amount of delay will be half of duty cycle of packet generation (TIMER_PERIOD_MILLI = 250)
   iv. *QOS_INJECT*: node will inject another packet for each relayed packet to increase the traffic
   v. *QOS_NORMAL*: node will not do any malicious activity and will act normal

3- Enhance the report BS prints by adding the packet throughput of the network
4- Add malicious node detection code
   i. Based on duty cycle of each node, and the route each packets takes to reach to BS, BS can have a good estimate of the number of packets it will receive in a normal network. Also delay of the packets will be estimated.

ii. If a node is acting up (applying QoS Attack) BS will be able to guess the route that a malicious node is resides on.
iii. Knowing duty cycle, expected number of packets and the route a malicious node resides on, BS will identify a node as malicious.
iv. BS will print the state of network as following:
    a. Normal
    b. Malicious node detected; Route:x,y,z,d,e (id of nodes on the route that malicious node is part of); ID: id_of_malicious_node

Note: While the network may have multiple malicious nodes, a node may exhibit ONLY one QoS attack behavior; in other words a node cannot mix Packet Drop, Delay and/or Inject behavior.

Submission Instruction:

In order to submit your work:

1- Make sure your project compiles and runs without any problem in Ubuntu version that we have provided you with. Error while compiling or executing your project will be interpreted as no submission.
2- Run "make clean" in your project directory to get rid of all the temp and executable.
3- Rename your project folder to your UMBC email address (e.g. if you email is yousef2@umbc.edu you will name your folder yousef2).
4- Make sure there are no extra files/directories inside your project folder.
5- ZIP the folder with any program that you have access to. "rar", "tar" and other extensions are not accepted. ONLY standard zip. (now you will have yousef2.zip)
6- Submit your zip file (yousef2.zip) via blackboard. Make sure your upload has worked fine by double checking that you can download and manipulate your submitted zip file (this makes sure there is no data corruption).
7- DO NOT email your homework.

Grading:

This assignment contributes 5% to the final class grade. Completed and successfully executing programs will be allotted 100%. Partial credit will be based on the level of efforts and the portion of the program that is working correctly.