

**CSE 591 - Spring 2009**  
**Lab IV: Sensor Network Projects**  
Status Report Due: April 20<sup>th</sup> (by e-mail)  
Due Date: May 6<sup>th</sup> (in class)

**Implement one of the following projects (START EARLY):**

**Project 1- Development of a shell to bridge sensor network and user**

In this project, you are required to develop an application to receive and inject packet into a sensor network. You should design an interface program for the user with necessary instructions, eg., receiving packet, injecting packet, etc. There are several kinds of command messages you could inject into the sensor network, which is defined in detail later. This may look like a shell in some operating system.

For example, when the user starts the program

```
senseapp> sense (user input)
```

The basestation will ask the network to do some sensing job.

```
senseapp> rd (report data)
```

The mote who has collected data should send the message to the basestation. (You should consider multi-hop scenario here. )

On receiving the reported data, the data are to be displayed on the screen.

The minimum set of commands you need to implement are:

1: `sense <node ID>`. Report light data from all motes or a particular mote.

2: `rd`

3: `inject <packet type>`

4: `off/on <node id>`. Use this command to control on and off of the motes

5: `ft` (file transfer).

Usually in sensor network, some configuration files (containing sampling frequency) are stored in the external flash on the mote. Here you need to learn how to use the external flash on the mote. The basestation is asked to send a small configuration file into the network. The file is divided into several packets; you should use some sequence number to make sure they are in the correct order. You also need some method to find the end of file: you could give the information as the header or add a different symbol (EOF) in the last packet payload. On completing receiving the packet, the mote stores it on its external flash. To simplify the task, there is only one configuration file on each mote, which means newer configuration file will flush the old one.

To get started, take a look at the code under `/apps/tests/storage` and read *tinyos tutorial 7 – storage*.

6: `rf` (retrieve the configuration file in the network)

Ask the mote in the network to send back their configuration file.

Groups of three should create three additional commands to control the network.

**Demo and submission**

You will be asked to demo your program on Weds May 6<sup>th</sup> during lecture. You are required to submit

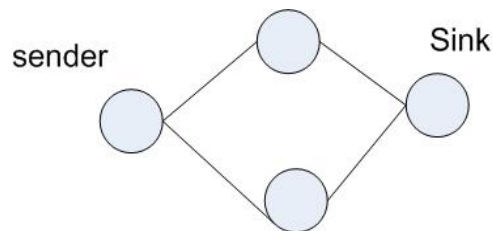
your source code right after the demo. (We will detect plagiarism using some software) A README/User guide should be provided which has the detail instructions to run your code.

## Project 2 - Performance evaluation of Sensor Network MAC protocols

Use [MLA](http://www.cse.wustl.edu/wsn/index.php?title=MLA) (MAC Layer Architecture, <http://www.cse.wustl.edu/wsn/index.php?title=MLA>) in TinyOS to evaluate the performance of three different sensor MAC protocols: Pure-TDMA, B-MAC, and S-MAC (for a group of three you should also include the performance of Z-MAC).

First, read carefully the basic papers on these MAC protocols. Download the MLA from the website, compile the code, and upload the notes. Since MLA has already provided the different protocol component, you don't need to implement these MAC protocols. But you may need to change their code to accommodate your experiment.

- You need to consider three scenarios:
  1. 4 node single-hop network. For the single-hop network, there are 3 senders and one receiver.
  2. 4 node multihop network. For the multihop network, the topology is shown below.
  3. A network topology of your choice with 6-8 nodes.



To construct this multihop network, you need to shrink the transmission range of the nodes by adding this line into the Makefile of your application, `CFLAGS += -DCC2420_DEF_RFPOWER=X` (try X from 1 - 30 to get the proper value for your setting). Another way to enable the multihop network is to enforce the topology in your nesC code. For example, in the MAC layer, when the node receives a packet it checks whether the sender is its neighbor; if not it simply drop the packet, instead of passing up to the network layer. Both methods have some disadvantages. By changing the transmission power, it is very hard for you to find an exact power level that guarantees two non-neighboring nodes can not receive packet from each other. However if you enforce the topology, chances are that you can not capture the real behavior of that MAC protocol and you need to touch the low-level TinyOS code. In the report, you should explicitly explain which method you choose. As long as the reason is acceptable and you construct the topology, you don't lose any point.

- The minimum performances you need to evaluate include throughput and packet loss rate. You need to change the transmission rate on the sender side to see how the throughput and packet loss rate varies. (Note you can expect higher packet loss in the multihop network if you set the power level. Why?)
- You may also consider enabling the packet acknowledgement in the data communication to see if it affects the performance.

The technical report should be no longer than 6 pages following the format of a standard conference paper (single space and double column, e.g., IEEE or ACM). The content of the report should contain brief descriptions of each protocol you choose to evaluate and their differences, the topology or

scenario you use, performance data and plotted figures, and all references. When you finish reading the papers describing those MAC protocols, you should get some idea how your report looks like.

### **Demo and submission**

You will be asked to demo an example of how you performed the network analysis on Weds May 6<sup>th</sup> during lecture. You are required to submit your TinyOS nesc code with an README to explain how to run your code & reproduce your data, and the technical report after your demo

### **Status Report (due April 20, by e-mail):**

Send an e-mail with the project you selected, the current status of your implementation and your project plan for the remaining 2 weeks. Details should be included on the command implementation (project 1) or the topology construction method and evaluation metrics chosen (project 2).

### **Handing in your code (for both projects):**

Send the Link to a tarball/zip of the files. Your tarfile should be named "Group<#>\_Lab4" which includes your \*.nc files and Makefiles, and README. Also include any files which you used for testing your application (TOSSIM instructions, etc)

Your code should be documented. Doesn't have to be extensive, but give clear information about what each component does and what is happening (if it is not obvious).

### **Grading (200 points):**

Each group will be graded on their demo (up to 50 pts) and the documentation of your code and README (up to 50 pts). The remaining points 100, will be based on the document, analysis, and/or efficiency of your code. Points will be taken off for not following the above directions.

**NO EXTENSIONS!!! NO EXCEPTIONS!!!**