

## Programming Assignment 2

ComS/CprE 454/554 Spring 2007

Total: 100 points

Due: Thursday, March 28, 11:59pm

### Problem Description

Suppose there are 3 motes whose addresses are 0, 1, and 2, respectively. Each mote maintains its local time.

Mote 1 and 2 each generate one sensor data item *every 10 seconds*. The content of each data item is as follows:

- Type – arbitrary one of four predefined types denoted as *type 0*, *type 1*, *type 2* and *type 3*
- Generator ID – ID of the mote that generates the data item
- Generation Time – the local time (unit: second) when the data item is generated
- Data – an array of 10 arbitrary integers

You will develop a *SimSensor* module to implement sensor data generation at each mote. Mote 0 simulates a sink that will query data stored in motes.

You are required to implement data management for motes 1 and 2 in two modes:

- **DCS mode** --- Mote 1 stores data of type 0 and type 2, and mote 2 stores data of type 1 and type 3. The data should be stored in their external flash memory (STM25p). If the memory is full when a new data item comes, the stored data item with the oldest generation time should be replaced with the new item (FIFO replacement rule). When the data of type *i* is queried, the query message should be forwarded to the storage node for the type (i.e., mote 1 for types 0 and 2, mote 2 for types 1 and 3), and the corresponding storage node services the query by sending back the queried data.
- **Indexed-DCS mode** --- Data items are stored locally at the motes that generated them. Mote 1 maintains the indexes for data of type 0 and type 2, and mote 2 maintains the indexes for data of type 1 and type 3. Again, both indexes and data items should be stored in the external flash memory. Among the 1MB available space, 256KB will be used for indexes, and the remainder for data items. If the memory (for either indexes or data) is full, the FIFO rule is applied for replacement. When data of type *i* is queried, the query message should be first forwarded to the mote maintaining the index for type *i*, and the mote further forwards the query to the corresponding mote storing the data; the mote storing the data will then return the queried data back to the sink.

The code for each data management mode (including the *SimSensor* module) should be compiled to one standalone program, named DCSmote and IDCSmote, respectively.

You are also required to implement mote 0 as a sink, which initiates data queries and receives responses to the queries. Every 1 minute, the sink should send out one randomly produced query belonging to the following types:

- Query for sensor data generated by mote 1 or mote 2 during a certain (past) time interval of length 30 seconds.
- Query for sensor data generated by both motes during a certain (past) time interval of length 20 seconds.
- Query for one certain type (i.e., type 0, 1, 2 or 3) sensor data generated by both motes during a certain (past) time interval of length 30 seconds.

For the purpose of testing, the sink should send its queries and received responses to the PC that it is connected to. The packet content should be able to be read and displayed by “java.net.tinyos.tools.Listen”.

### Submission

- You should zip the directory of your source code and submit it via WebCT.
- You should ensure that, once your submission is expanded under <tos>/apps directory, it can be compiled correctly. Otherwise, you may get no point.
- Your submission should be named as assignment-2-XXXXX-YYYYY.zip or assignment-2-XXXXX-YYYYY.tar.gz, where XXXXX and YYYYY are the last names of the students. One group only needs to submit one copy.

- The submission should include a readme file briefly explaining your solution; adequate comments between your codes are strongly encouraged to improve code readability.