TASK
(Tiny Application Sensor Kit)

By
Suchitha Koneru
WSN(CS 663) Winter 2004
Professor Dr.Ajay Gupta

Task Components

Purpose of task:
Tiny Application Sensor kit (TASK) aims to break down the barrier to entry for non-sophisticated users to develop and deploy their own sensor network applications.

Targeted towards actual users of Sensor Networks

TASK is a suite of Java utilities that allows you to manage a network of motes deployed for remote data collection.

Task consists of the following:
- Task DB: provides the mote side functionality for the task, the core part of TASK
- Task DBMS: a relational data base management system which holds sensor readings, sensor network health statistics, currently PostGreSql is used
- Task Server: a server process running on the sensor network gateway, acts as a proxy server for the sensor network on the internet

Task Components contd.....

- Task Client Side tools:
  - Task deployment: record mote placement
  - Task Configuration: Query Designing, filtering and aggregating
  - Task Visualization: Monitor network health and Sensor Readings
  - Task Field Tool: can run on a PDA and helps users to diagnose and resolve problems in the field
Running Task

- Tinydb.conf file should be changed with respect to the AM group
  - AM group ID should be the same in TinyOS and in Tiny DB
  - TOS_AM_GROUP(AM.H) and am-group-id(tinydb.conf) should have the same value

- The following statement needs to be added in the bash.bashrc file found in the etc directory under cygwin
  - export CLASSPATH=$CLASSPATH:tinyos-1.x/tools/java/jars/pgjdbc2.jar
    // for getting an access to the JDBC ODBC drivers

- The PostGreSql database should be initialized before it could be used
  - so the setup-task.sh should be run from the directory
  - C:\tinyos-1.x\tools\java\net\tinyos\task\tasksvr

Contd…. Programming motes with TASKApp

- Mode 0 should act like a base station
- Running TASKApp using Tossim
- Starting the Task Server
  - C:\tinyos-1.x\tools\java\net\tinyos\task\tasksvr\TASKServer -sim
    - Starts on port number 5431
- Starting the Task Visualization tool (GUI)
  - C:\tinyos-1.x\tools\java\net\tinyos\task\taskviz\TASKVisualizer localhost
Task Visualizer

- Supports visual deployment, configuration and visualization of a sensor network.
- The following interface is formed.
Running commands

For debugging TASK-based sensor networks
- Walk up to a task mote and modify its state by issuing commands
- The field tool only operates on the motes that are in direct radio range
- Runs on hand held devices

Field tool configuration
- Serial or IP configuration, usually serial is selected
- Host or serial port name for communication
- Port/ Baud rate, baud rate for serial communications is 19200 baud for mica and 57600 baud for mica2, port name for IP communication
- Group ID of the TASK motes that is an IP group ID
- Local ID for the field tool should be different from Task mote ID
- Msg Size - mote radio message size, must be 56
- Command period - commands selected by the user are sent multiple times
Contd...

- **Command count**: the number of times a command is sent.
- **Wake up period**: the interval between the messages the field tool sends to advertise its presence.
- **Mote timeout**: after this period of time, motes which have not been heard by the field tool are removed from the mote list.

**Field Tool Commands**

- Commands are of debugging nature.
- Green toggles the green led, can be used for checking the tool/mote communication, motes return done as result.
- Beep sounds the mote's sounder.
- Reset resets the mote.
- Ping returns the vital statistics regarding the operation of the TASK-mote.
  - Result is of the form "X,V,Parent N,RAM R,q1n Q,mhq M,dpth D, qual L q1 Q1, q2 Q2".
  - X is the current voltage of the mote's batteries.
  - N is the parent mote in the routing tree.
  - R is the amount of RAM available for Queries.
  - Q is the radio send Queue length.
  - D is the depth of the mote in multi hop routing tree.
  - M is the forward queue length of the multi hop routing layer.

**Field tool**

- The field tool contains options for green, beep, reset, ping, and a result pane displaying the vital statistics.
Code

- TASKApp contains the following files
  - TASKApp.nc
  - FieldApp.nc
  - Field.nc
  - Ping.nc
  - TASKApp.cc
  - FieldApp.cc
  - Field.cc
  - Field.h
  - Ping.cc

- Field App uses the components Field, TimerC, GENERICCOMM/PROMISCUOUS as Comm., LedsC, Command, Ping, Attr, RandomLFSR, ServiceSchedulerC.

Contd..

- 5 types of commands: setLedr, setLedy, setLedG, Reset, Pot

- Command component implements 3 interfaces namely Command Register, StdControl and CommandUse:
  - Command Register interface has 3 methods
    - command result_t registerCommand(char *name, TOSType retType, uint8_t retLen, ParamList *paramList);
    - event result_t commandFunc(char *commandName, char *resultBuf, SchemaErrorNo errorNo, ParamVals *params); called by Command.td when a specific command is invoked by the user
    - command result_t commandDone(char *commandName, char *resultBuf, SchemaErrorNo errorNo);

- CommandUse interface has 3 methods
  - command CommandDescPtr getCommand(char *name);
  - command uint8_t numCommands();
  - command CommandDescsPtr getCommands();

Contd..

- GENERICCOMM/PROMISCUOUS.cc
  - Similar to the Generic Comm. but uses on checksum and promiscuous mode
  - RandomLFSR
    - Implements the interface Random, pseudo random number generator
    - Basing on the local address of the mote 16 bit random number is generated

- ServiceSchedulerC
  - QueuedSend
    - Sends the message reply's in a FIFO manner

- QueuedSendM (conf file)