Design and Implementation of Sensor Network Communications

Chuan Jiang Kaichun Mo Wenkang Yu

I. INTRODUCTION

This paper is the report for the course project of Computer Networking taught by Prof. Yanmin Zhu in Spring 2015. In this project, we develop a software support for a tiny sensor network. Communication protocol are designed and actual implementation are done in nesC programming language.

In this report, we will introduce our main protocol and algorithm first. Then, we implemented the algorithm detailedly and have run it on telosb motes successfully. Finally, as an additional function, GUI is implemented as a monitor to see the immediate results sensed by the nodes in the network.

II. PROTOCOL AND IMPLEMENTATION

In this section, we will briefly introduce you how we design the distance vector protocol and how we implement our algorithms using nesC on tinyos. Firstly, usage of program will be detailedly explained. Then, we will show the data structure that we used to implement packet switching. Next, our main algorithm and protocol details will be given. Finally, we will introduce some problems when we implemented the protocol and the corresponding solutions or advices to each of them.

A. Structure of Software

The whole program or software is developed in three separate parts: sensor part, sink part and UI part. In this section, I will mainly introduce the first two while leaving the last one to the next section.

Sensor part is the part of code to implement distance vector multi-hop protocol on all nodes except for a special node called sink, which is attached to a PC and acts as a final collector. All nodes except for sink will not only collect data in the surrounding but have to route and relay packets of other nodes as well.

Sink Part is the code to implement the protocol on sink node. Similar to sensor node, sink will also have two main functions. It needs to sense the data nearby as well as receive the data sensed by all other nodes in this sensor network and pass them to the PC to which it attaches.

We used simplified distance vector routing protocol to implement the routing and forwarding of packets. You can refer to Main Algorithm Subsection to see the design and details of the protocol. To facilitate our protocol, we design a specific data structure called myMessage to encapture the data and the control information. One can see Data Structure Subsection for details.

Next, we will introduce the directory and all the files used in our software. Our software comprises of two folders, which are sensor and sink. In the sensor folder, there are one nesC component SensorC and one figuration SensorP. They are going to be installed on the mote. One can also find a C header file myMessage.h in this folder, which contains the packet format and some necessary constants. While, in the sink folder, besides component SinkC, configuration SinkP and the header file, one can also find two Java files Sink.java and MFrame.java. These are the files used to display the CUI and the GUI of the final results.

Now, let’s see how to install and use our program firstly.

B. Usage of Program

To install our software, one need several telosb motes with all normal functions available and normally runnable. To install the sensor node, one can step into sensor folder and type the following demand in the terminal.

make clean
make telosb install,$1

where $1 is the node identity assigned to this node and it can be taken from the range \{1,2,\ldots,10\} for this sensor network of small size. To install the sink node, just direct into sink folder and type the exactly same code. Notice that it is your responsibility to guarantee every nodes have different identities and all node identities are chosen in the required range. Otherwise, the software may perform badly or stop working in the extreme case.

To see the result, direct into sink folder and type the following command.

java -cp lib/jcommon-1.0.23.jar:lib/jfreechart-1.0.19.jar:.$(CLASSPATH) Sink -comm serial0/dev/ttyUSB$1:telosb

where $1 is the USB serial number of your sink node when attached to PC. Then, if everything is right, you can see both CUI and GUI on the screen. GUI will show you the immediate changes of temperature, humidity and light detected near all the nodes that can be transmitted to the sink node directly or by other nodes. CUI will show the route for any packet to follow from the start node to the sink.

C. Data Structure

Look at the following code firstly.
typedef nx_struct myMessage{
    // type of message
    nx_uint8_t type;
    // destination nodeid
    nx_uint8_t dest;
    // data message
    nx_uint16_t nodeid;
    nx_uint16_t temperature;
    nx_uint16_t humidity;
    nx_uint16_t light;
    nx_uint16_t seq_num;
    // control message
    nx_uint8_t dv_dist;
    // only for showing the routes in CUI
    nx_uint8_t path[MAX_NODE_NUMBER];
    nx_uint8_t tot_node;
} myMessage;

Detailed explanations can be accessed in the following list.

- **type**: there are four types of messages, which are Control Message, Data Message, Sense Message and Reply Message. See the protocol for details.

- **dest**: the node identity of the receiver.

- **nodeid**: the node identity of the sender.

- **temperature**: the temperature message passing from sender to receiver.

- **humidity**: the humidity message passing from sender to receiver.

- **light**: the light message passing from sender to receiver.

- **seq_num**: the sequence number of the packet.

- **dv_dist**: used when the message type is Control Message to pass the minimal distance of the sender to the sink node.

- **path** and **tot_node**: used to store the route through which the packet is transmitted.

### D. Main Algorithm

We adopted and implemented a simplified version of the **distance vector routing algorithm**. Since every node only needs to transmit the packet to the sink, we can only record the optimal distances of all nodes to the final destination.

The sink node and sensor nodes have different actions due to their difference on functions.

1. **Sensor node**

   A sensor node periodically sends its current data and periodically senses neighbors using timeout.

   When a data timeout occurs, a sensor node reads the current data, packs the data and the path information, then sends the package to the neighbor which is in its shortest path to the sink.

   When a sense timeout occurs, a sensor node tries to send a sense-message to every node and starts a new timeout. Other sensors which reply within this timeout are considered the neighbors.

   When a sensor receives a data-message, it can simply relay the message according to its path information.

   When a sensor receives a sense-message, it should send back a reply-message which contains its shortest distance to the sink.

   When a sensor receives a reply-message, which indicates it sends sense-messages before, it should include the new neighbor and try to update the shortest distance. If the shortest distance is updated, a control-message should be sent to other nodes to inform this update.

   When a sensor receives a control-message, which indicates one of its neighbors updates the shortest distance, it should also try to update. If the update happens, the sensor should also send control-message to inform this update.

2. **Sink node**

   The sink node periodically sends its current data to PC using timeout. When a data timeout occurs, the sink node reads the current data, packs the data and sends the package to PC.

   When the sink node receives a data-message, it relays the message to PC.

   When the sink node receives a sense-message, it sends back a reply-message which contains the distance of 0.

   The sink node will not receive reply-message since it doesn’t need to sense neighbor. And it doesn’t need to respond to control-message since it doesn’t need to maintain shortest path.

### E. Problems and Solutions

1. **Count to Infinity**

   When a node fails, other nodes may update with lagged information, which leads to count-to-infinity problem. We set a maximum number for shortest distance to terminate this procedure. But it is not flexible enough and takes time to reach the maximum distance. Another solution would be poisoned reverse, which is not implemented in this project.

2. **Serial Number of Nodes**

   In this project, we assume that the serial number of each node is within a certain number. This is convenient for us to sense neighbors and compute shortest path but brings the potential problem when a serial number exceeds the maximum. A solution is to dynamically allocate an address to each node and manage accordingly, which is a bit complex to implement.

### F. Some Advices

#### III. UI DESIGN

We also design both CUI and GUI for this project to make this sensor network easy to use. The basic implementation is under the JAVA virtual machine by listening to the USB port used by Sink node of the network.

#### A. CUI

The motivation of our CUI is towards both debugging processing and correction checking. We lists all the packets the Sink node receives by chronological order. For each packet, we describe the following details:
1. The sequence number of the packet which stands for the relative sending order in different Sensor node.
2. The whole data sensed by the Sensor including temperature, humidity and light in this experiment.
3. The routing path from the particular Sensor to the Sink node.

B. GUI

In the designation of our GUI, we come up with an elaborate way to present the collected data considering both convenience and clarity. We use two JAVA toolkit (AWT[1] and JFreeChart[2]) to implement the GUI, and several features of our GUI are as follows:
1. We put three buttons on the top of the frame for user to choose which kind of sensed data is to present.
2. We put the data sensed by all the sensor on the same graph for the sake of course comparison between different sensors.
3. The data graph is updated dynamically according to the time in receiving packets by Sink node. Also, we automatically delete the ”outdated” packets and maintain the ”up-to-date” sensed data.

IV. CONCLUSION

In this project, we successfully developed a sensor network using distance vector protocol. It supports the data communication in the network and the data collection in the sink node. It also provides a simple UI to operate on.

V. ACKNOWLEDGEMENT

The authors would like to thank Professor Zhu and the teaching assistant for the help and guidance during this project.

REFERENCES
http://docs.oracle.com/javase/7/docs/technotes/guides/awt/index.html