单汇点无线传感网络的实现

贾枭, 李诗剑, 王孟晖

January 5, 2012

1 简介

本次实验中，我们使用一些无线传感节点，搭建了一个单汇点无线传感网络。传感器节点是一个由电池供电的小型设备，它可以采集附近的温度和湿度信息，并且可以和其他节点进行无线通信。我们设计了节点之间的路由算法和通信协议，实现了：

1. 实时地显示所有传感器节点当前的温度和湿度信息；
2. 自适应动态路由：无论何时调整节点的位置或者添加移除一些节点后，它们依然能够正常工作。并且路由算法能够保证每个节点选择的传输路径一定是最短的（跳数最少）；
3. 错误检测：节点能够正确检测数据在传输过程中产生的错误；
4. 图像输出：计算机通过采集到的数据，为每个节点绘制了温度、湿度变化曲线。

此外，我们的所有协议都具有可扩展性，支持传输任意数据而不仅仅是温度信息。路由算法亦可应用到较多节点。

本文第 2 节将从较高的抽象层面介绍整个系统，第 3 节详细讨论路由协议和算法，第 4.1 节介绍数据传输协议，第 5 节将展示我们实验结果，最后第 6 节总结我们的设计。

2 系统概览

在所有传感器节点中，有一个节点被称为 *Sink*。它的主要责任有：

- 采集附近的温度和湿度数据；
- 收集所有传感器节点发送过来的温度和湿度数据；
- 把收集到的数据通过 USB 接口发送给计算机。

除了 Sink 之外的所有其他节点，需要做的是：

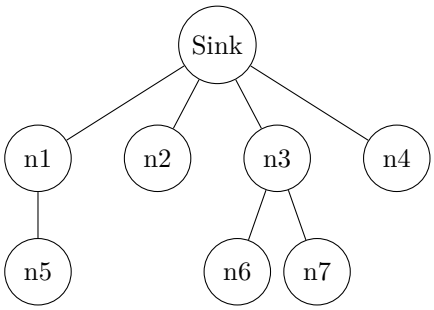

Figure 1: 节点路由示意图

- 采集附近的温度和湿度数据;
- 把采集到的数据发送给 Sink。

由于并非所有节点都能够直接和 Sink 通信，所以向 Sink 发送的数据包必须经由其他节点转发。为了确定每个节点在发送数据时先发给谁，我们需要一个路由协议。

我们的路由协议采用了 Dijkstra 算法。每个节点将会选择一个其他的节点，称为父节点，作为自己发送数据包的对象。每个节点收到数据包后，也会把它转发到自己父节点。经过几次转发后，数据包最终将到达 Sink。

为了应对新增或者离开的节点，以及节点的位置变化，路由协议将会周期性地执行。

每个节点也会周期性地读取温度和湿度的测试值。然后，它们把测试值发送给自己的父节点。父节点将会转发数据包给父节点的父节点，最终数据包会发到 Sink。Sink 得到的数据包都会给计算机。

3 路由协议

每隔一段时间，Sink 会广播一个控制包。控制包包含了节点号 `node_id`、距离 Sink 的跳数 `hop_num`、序列号 `seq_num`、以及校验和 `checksum`。Sink 发出的控制包中，`hop_num`等于 0。对于收到控制包的节点，它会查看包的序列号是否和当前保存的序列号相同。如果不同，说明新一批的路由信息来了，需要更新自己的父节点，并广播转发这个控制包。或者，如果控制包里跳数加一比当前的跳数更小，说明有一个更优选择，这时也需要更新并转发这个控制包。

3.1 控制包定义

```
typedef nx_struct ctrl_param_t {
    nx_uint16_t magic_num;
    nx_uint16_t node_id;
```

```
    nx_uint16_t hop_num;
    nx_uint16_t seq_num;
    nx_uint16_t checksum;
} ctrl_param_t;
```

这里 magic_num是为了区分其他数据包。 node_id表示发出这个包的节点的编号。 hop_num表示这个节点距离 Sink 有几跳。 seq_num是为了区分同一批控制包。 checksum把以上所有位加起来，用来检测传输过程中的比特错误。

3.2 Sink 节点路由算法

Sink 节点周期性地广播一个控制包。包的 hop_num置为 0。每次广播的seq_num递增。

3.3 普通节点路由算法

Figure 2: 初始化

procedure INIT
    hop_num = ∞
    parent = -1
    hop_seq = -1
end procedure

Figure 3: 收到一个控制包时

procedure ONRECEIVE(ctrl_param_t packet)
    if hop_seq $\neq$ packet.seq_num or packet.hop_num + 1<hop_num **then**
        hop_seq $\leftarrow$ packet.seq_num
        hop_num $\leftarrow$ packet.hop_num + 1
        parent $\leftarrow$ packet.node_id
        BroadcastControlPacket(my_node_id, hop_num, hop_seq)
    end if
end procedure

收到数据包时，只有产生更新或者新一批的数据包到来时，才会转发。这样可以避免广播风暴的产生。

4 数据传输协议

当采集到温度和湿度的数据后，只需要向 parent节点发送数据。 parent节点在收到数据之后，会负责把你的包通过其他人最终转发到 Sink。

4.1 数据包定义

```
typedef nx_struct data_param_t {
    nx_uint16_t magic_num;
    nx_uint16_t node_id;
    nx_uint16_t seq_num;
    nx_uint16_t hop_num;
    nx_uint16_t temperature;
    nx_uint16_t humidity;
    nx_uint16_t checksum;
} data_param_t;
```

这里加入了 `seq_num`是为了避免数据包不按顺序到达，导致收到的温度、湿度信息并不是按照时间递增的顺序。

5 实验结果

我们把上面的设计用 nesC 语言编程实现。实验中，我们每 10 秒执行一遍路由协议，每 0.6 秒采集一次数据点。我们用 Java 把采集到的温度和湿度信息绘制到了曲线上，如图 5 所示。

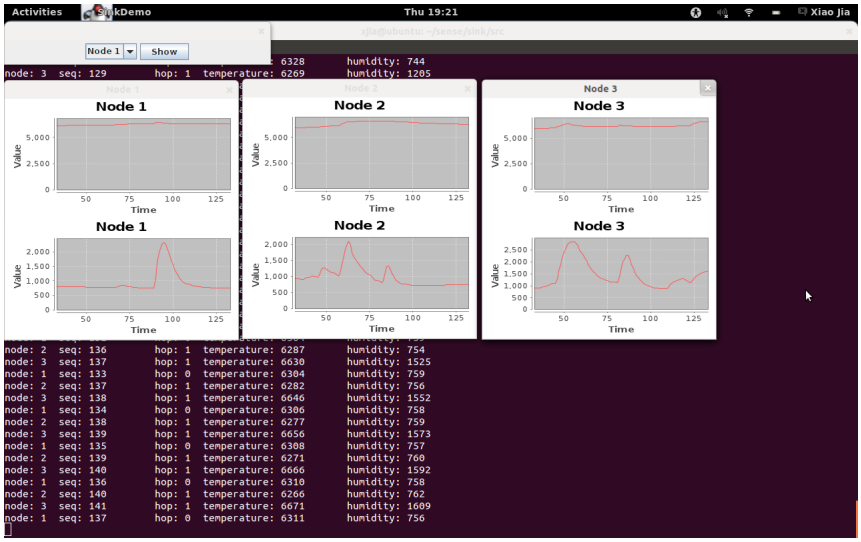

Figure 4: 程序运行截图

6 总结

本次实验中，我们为单汇点传感网络设计了路由协议和数据传输协议，使得所有传感器节点能够实时地向计算机返回温度和湿度信息。

路由协议具有较大的容错性，允许中途新增节点或者一些节点退出。在节点位置发生变化的时候，也能够重新选择一条更短的路径进行数据传输。

数据传输和路由都进行了数据校验，保证数据包正确地被接收。通过为数据包添加序列号，可以有效防止由于数据包不按顺序到达导致采集到的信息失真。

除了基本功能的实现，我们还将温度和湿度的信息绘制成曲线输出，从而可以更直观地看出各个节点处温度和湿度的变化。

References

[1] *TinyOS Tutorials*. http://docs.tinyos.net/tinywiki/index.php/TinyOS_Tutorials