实验 3　静态路由转发实验

实验目的

本实验的目的是实现节点和节点间的无线通讯和路由转发，可以使用串口发送消息到基站（指定编号为 1 的节点为基站），然后转发到指定节点，或者从某个节点定时发送数据，经过固定路径可以转发到基站节点，通过简单的静态路由实验了解无线传感网络的数据路由过程和传感器数据采集过程。

实验要求

根据修改实验二的程序 RadioAndSerial，添加静态路由转发功能，具体要求：

1.节点之间通过单播的方式收发数据；

2.节点路由方式为：目标节点编号大于自身节点编号的，将数据转发到编号为自身节点编号+1 的节点，目标节点编号小于自身节点编号的，转发到编号自身节点编号-1 的节点，节点只能直接与相邻编号的节点通讯，即数据只能在相邻节点之间转发。

３．中继节点的 LED 显示目标节点的编号，停留时间为 1 s，目标节点 LED 显示计数值，停留时间为 3s，数据包只转发一次，不进行重复发送，数据到达目标节点后计数值改为原来的计数值加上该节点的节点编号，沿原路径返回到基站；

４．指定编号为 1 的节点为基站，基站进行串口读写为广播方式,如果基站节点收到数据，目标节点为自身时将数据发送到串口，否则发送到无线模块；

实验内容

1、静态路由转发

例子程序 BlinkToRadio 中包含了简单的发送和接收数据包功能，只需要在接收到数据后进行处理，确认目标是自身还是需要进行转发．并根据相应情况设置 LED 灯的显示以及定时器的时间，由定时器触发数据转发事件；

简单的路由转发逻辑可以按如下方式实现：

```
uint16_t  aim_node = 1;
if(nodeid == TOS_NODE_ID){
   counter = counter + TOS_NODE_ID ;
   nodeid = 1;
 }
 if(nodeid > TOS_NODE_ID){
    aim_node = TOS_NODE_ID + 1;
 } else if(nodeid < TOS_NODE_ID) {
    aim_node = TOS_NODE_ID - 1;
}
```

在该代码中，如果当前节点为目标节点，将会将计数值加上该节点的编号返回到基站节点(假设基站节点设置为 1)；

Timer 定时设置：由于只需要一次定时，所有如果使用函数 startPeriodic()则需要在定时器触发之后调用 stop()，或者使用函数 startOneShot()，只使用一次定时器；

固定编号为 1 的节点充当基站，通过 USB 接口连接在电脑上，可以用来监控整个数据的转发过程，基站会将所有从无线模块收到的数据返回到串口显示；

2．基站与节点之间的通信

通过修改 RadioAndSerial 代码实现数据的转发和简单路由，除了基站节点，其余节点只通过无线模块收发数据，通过 LED 显示部分结果；基站节点可以控获取最终的数据包；

基站的实现是基于实验 2，只需要实现简单的基站功能即可，通过 mig 工具构建 BllinkToRadioMsg.java 类，可以查看通过基站收到的数据包：

```
Java BlinkToRadioMsg -comm serial@/dev/ttyUSB0:telos
```

或者通过 Listen 程序查看二进制的数据包：

```
Java net.tinyos.tools.Listen -comm serial@/dev/ttyUSB0:telos
```

3.通过串口读写数据到指定节点

简单的写串口可以使用 net.tinyos.tool.Send 类，使用方法在实验 2 中已经介绍，但是使用该类不能在写的同时接收，可以使用 SerialForwarder 工具实现写串口和读串口的同时进行；

另外还可以通过编写实验 2 中的 BlinkToRadio.java 类实现数据的发送与接收，数据发送的函数为:

```
MoteIF.send(2, payload);
```

本实验中设置基站只能发送数据到其中一个节点，与固定的一个节点通信，以实现静态路由（这里假设基站的节点编号为 1，固定通信节点为 2），如果要使用广播的方式发送数据，则可以使用地址 MoteIF.TOS_BCAST_ADDR;

从 PC 端通过串口发送的数据包含 nodeid 和 counter ，需要使用 java 程序从终端中获取这两个数据，然后发送到基站，基站会根据发送目标(即 send() 函数中的第一地址参数)将数据发送到指定的节点；

Java 代码中需要导入包：

```
import java.util.Scanner;
```

```

```

读取参数：

```
Scanner sc = new Scanner(System.in);
nodeid = sc.nextInt();
counter = sc.nextInt();
```

添加 BlinkToRadio.java 后需要修改 Makefile 文件，添加编译依赖：

```
BUILD_EXTRA_DEPS += BlinkToRadio.class
```

添加清除项

```
CLEAN_EXTRA = *.class BlinkToRadioMsg.java
```

文件依赖：

```
BlinkToRadio.class: $(wildcard *.java) BlinkToRadioMsg.java
    javac  *.java
```

注意第二行的开头是 tab
清除之前的生成（默认重新编译不会覆盖.class 文件）

```
Make clean
```

重新编译

```
Make telosb
```

将基站部分烧录到节点 1，BlinkToRadio 部分烧录到三个节点分别 2,3,4

```
Make telosb install,1
```

(不同节点使用不同的编号)

然后 PC 连接基站节点，使用命令

```
java BlinkToRadio -comm serial@/dev/ttyUSB0:telos
```

进行数据的收发，典型的例子如下，向节点 2 发送 counter=3

```
Input the nodeid and counter(like 1  2): 2 3
Sending to 2 number is : 3
Received packet to: 65535 nodeid: 1 counter: 6
```

在这个例子中，从 PC 向节点 2 发送数据 3（数据包中目标节点为 2，计数值为 3），过程为：基站收到数据后显示目标节点 2，并在 1 秒后将数据包发送到节点 2，节点 2（目标节点）收到后 LED 显示 3（计数值）然后在 3 秒后将 counter 值设为 3+2 ，目标节点改为 1，然后转发到节点 1，节点 1 显示 5，并在 3 秒后将计数值 5 ＋ 1 发送到串口，PC 通过串口接收到节点 1 的计数值 6；

PC 端写串口（同实验 2）

通过编写 BlinkToRadio.java 可以控制从 PC 端发送数据，将数据发送的指定的目标节点，从而更好的控制数据的转发和路由过程.

通过使用包 net.tinyos.message.* 中的 MoteIF 类,实现 PC 端自定义的数据收发，

```
public class BlinkToRadio implements MessageListener {

private MoteIF moteIF;

public BlinkToRadio(MoteIF moteIF) {
  this.moteIF = moteIF;
  this.moteIF.registerListener(new BlinkToRadioMsg(), this);
}
```

使用方法 registerListener()方法注册自动生成的类 BlinkToRadioMsg()；

```
public void messageReceived(int to, Message message) {
    BlinkToRadioMsg msg = (BlinkToRadioMsg)message;
    System.out.println("\nReceived packet to: " + to + " nodeid: "+
msg.get_nodeid() +" counter: " + msg.get_counter());
    System.out.print("Input the nodeid and counter(like 1  2): ");
  }
```

需要实现 messageReceived()函数，其中参数 to 是消息的目的节点编号（单播或广播地址），默认会接收所有能监听到的数据包；

```
BlinkToRadioMsg payload = new BlinkToRadioMsg();
payload.set_nodeid(nodeid);
payload.set_counter(counter);
moteIF.send(2 , payload);
```

Payload 可以使用 get 和 set 方法获取或使用其中的数据，注意和 BlinkToRadio.h 头文件的定义的一致，send() 函数第一个参数为发送的目的节点（可以指定单播地址）；

完成 BlinkToRadio.java 类之后还需要修改 Makefile 文件，添加编译依赖:

```
BUILD_EXTRA_DEPS += BlinkToRadio.class
BlinkToRadio.class: $(wildcard *.java) BlinkToRadioMsg.java
      javac  *.java
```

之后运行 make clean ， make telosb 重新编译，即可生成 BlinkToRadio.class 文件，可以使用这个类来进行串口的读写操作，命令如下：

```
Java BlinkToRadio -comm serial@/dev/ttyUSB0:telos
```

获取用户输入和格式化输出的逻辑需要在 BlinkToRadio.java 中实现，参考 TestSerial.java 即可；

实验过程说明

首先需要修改节点通信的相关信道，选择 26 信道，在 Makefile 文件中进行修改， PFLAGS+=-DCC2420_DEF_CHANNEL=26。对 BlinkToRadio 程序中的 Makefile 文件进行修改，使用 mig 创建 BlinkToRadioMsg 的 java 对象 ：

```
BlinkToRadioMsg.class: BlinkToRadioMsg.java
      javac BlinkToRadioMsg.java
BlinkToRadioMsg.java:
mig java -target=null $(CFLAGS) -java-classname=BlinkToRadioMsg
BlinkToRadio.h BlinkToRadioMsg -o $@
```

1 号节点作为基站节点， 2 、 3 号为普通节点，可以通过串口和 LED 清楚的看到整个通信转发过程。

提示步骤：

1.可以在实验 2 的基础上修改 BlinkToRadio 路由转发逻辑。

2.首先实现基本的节点数据转发（至少三个节点）,然后再实现从 PC 端进行任意的指定转发。