华中科技大学

课程实验报告

课程名称： 物联网通信技术

专业班级： 物联网工程 1601

学　　号： U201614898

姓　　名： 潘　　翔

指导教师： 徐 海 银

报告日期： 2018.12.28

计算机科学与技术学院

目　录

1　节点-节点单播无线通信

1.1 实验目的与要求

本实验介绍了如何在 TinyOS 上进行节点与节点之间的无线通信。通过这个实验，熟悉通信相关的组件及接口以及如何以单播的方式发送和接收消息。

根据提供的例子程序，详细了解程序结构，并尝试进行程序的修改运行

1) 熟悉 TinyOS 无线通信的接口和通信流程；
2) 修改例子程序并测试运行。

1.2 实验内容

1) 消息发送
2) 消息接收

实验要求说明：

实现类似跑马灯的效果，两个节点都维护计数值 counter，初始值为 1。节 点 1 每隔 1 秒发送计数值到节点 2 ,节点 2 每隔 1 秒发送自身计数值+ 1 到节点 1.。节点 1 和节点 2 均接收对方发送的计数值，收到后更新 counter，用 LED 灯显示 counter 的末三位。

效果：

节点 1,节点 2 都开着的时候，节点 1 和节点 2 的 LED 灯显示的值每隔 1 秒通增 1。此时按住其中一个节点的 RESET，另外一个节点的显示将不会停住不再变化，放开之后，又重新从 1 开始计数。提示：需要分辨节点的编号以发送不同的计数值。

1.3 实验过程与结果

1.3.1 开发环境

OS：	Manjaro 18.0.2 Illyria(Arch-Based Distribution)
Kernel:	x86_64 Linux 4.20.0-1-MANJARO
TinyOS：	TinyOS(Make Version 3.0.0)
GCC:	gcc (GCC) 7.4.1
nescc:	1.3.6
Text Editor:	VisualStudio Code(1.30.1 x64)

1.3.2 BlinkToRadio 源程序

```
hover@wings ~/tinyos-main/apps/tutorials/BlinkToRadio master pwd
/home/hover/tinyos-main/apps/tutorials/BlinkToRadio
```

图 1-1 BlinkToRadio 源程序

TinyOS 官方提供了系列源程序与 tutorials 目录下供学习。包括基本的串口程序，无线程序和传感器程序。

```
hover@wings ~/tinyos-main/apps/tutorials/BlinkToRadio master make telosb docs
[INFO] making documentation for BlinkToRadioAppC on telosb
nescc -fsyntax-only -fnesc-dump=interfacedefs -fnesc-dump='components(wiring)' -fnesc-dump=interfaces -fnesc-dump='functions(!global())' -fnesc-dump='referenced(interfaces
,components,functions)' -gcc=msp430-gcc -Wnesc-all -fnesc-include=tos -fnesc-scheduler=TinySchedulerC,TinySchedulerC,TaskBasic,TaskBasic,TaskBasic,runTask,postTask -fnesc-
cfile=build/telosb/app.c -fnesc-separator=__ -I../../../tos/platforms/telosb -I../../../tos/platforms/telosb/chips/stm25p -I../../../tos/platforms/telosa -I../../../tos/pl
atforms/telosa/chips/cc2420 -I../../../tos/platforms/telosa/chips/s1087 -I../../../tos/platforms/telosa/chips/s10871 -I../../../tos/platforms/telosa/chips/sht11 -I../../..
/tos/platforms/epic/chips/ds2411 -I../../../tos/chips/cc2420 -I../../../tos/chips/cc2420/alarm -I../../../tos/chips/cc2420/control -I../../../tos/chips/cc2420/csma -I../..
/../tos/chips/cc2420/interfaces -I../../../tos/chips/cc2420/link -I../../../tos/chips/cc2420/lowpan -I../../../tos/chips/cc2420/lpl -I../../../tos/chips/cc2420/packet -I..
/../../tos/chips/cc2420/receive -I../../../tos/chips/cc2420/spi -I../../../tos/chips/cc2420/transmit -I../../../tos/chips/cc2420/unique -I../../../tos/chips/cc2420/securit
y -I../../../tos/chips/msp430 -I../../../tos/chips/msp430/adc12 -I../../../tos/chips/msp430/dma -I../../../tos/chips/msp430/pins -I../../../tos/chips/msp430/timer -I../../
../tos/chips/msp430/usart -I../../../tos/chips/msp430/sensors -I../../../tos/chips/stm25p -I../../../tos/chips/sht11 -I../../../tos/chips/ds2411 -I../../../tos/lib/timer -
I../../../tos/lib/serial -I../../../tos/lib/power -I../../../tos/lib/onewire -mmcu=msp430f1611 -fnesc-target=msp430 -fnesc-no-debug -DPLATFORM_TELOSB -Wall -Wshadow -I../../
../tos/system -I../../../tos/types -I../../../tos/interfaces -DIDENT_APPNAME=\"BlinkToRadioApp\" -DIDENT_USERNAME=\"hover\" -DIDENT_HOSTNAME=\"wings\" -DIDENT_USERHASH=0
xdd15bc99L -DIDENT_TIMESTAMP=0x5c26c6a8L -DIDENT_UIDHASH=0x546c80b1L BlinkToRadioAppC.nc > ../../../doc/nesdoc/telosb/doc.xml
../../../tos/chips/cc2420/lpl/DummyLplC.nc:39:2: warning: #warning "*** LOW POWER COMMUNICATIONS DISABLED ***"
nesdoc-archive --topdir ../../..   ../../../doc/nesdoc/telosb < ../../../doc/nesdoc/telosb/doc.xml
interface CC2420Config (tos.chips.cc2420.interfaces.CC2420Config)
interface ResourceRequested (tos.interfaces.ResourceRequested)
interface HplMsp430I2CInterrupts (tos.chips.msp430.usart.HplMsp430I2CInterrupts)
interface BusyWait (tos.lib.timer.BusyWait)
interface SplitControl (tos.interfaces.SplitControl)
interface ResourceConfigure (tos.interfaces.ResourceConfigure)
interface Msp430Capture (tos.chips.msp430.timer.Msp430Capture)
interface LocalIeeeEui64 (tos.interfaces.LocalIeeeEui64)
interface Msp430Compare (tos.chips.msp430.timer.Msp430Compare)
interface CC2420Register (tos.chips.cc2420.interfaces.CC2420Register)
interface Msp430TimerEvent (tos.chips.msp430.timer.Msp430TimerEvent)
interface PacketTimeStamp (tos.interfaces.PacketTimeStamp)
interface RadioBackoff (tos.chips.cc2420.interfaces.RadioBackoff)
interface GpioCapture (tos.interfaces.GpioCapture)
interface Boot (tos.interfaces.Boot)
interface SendNotifier (tos.interfaces.SendNotifier)
```

图 1-2 生成文档图

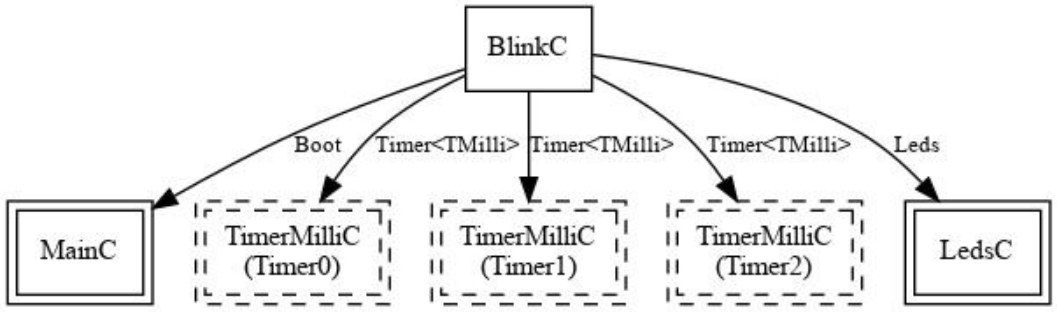


图 1-3 代码结构图

整个 tiny-os 架构采用组件式，故相应的代码结构利于分析，在原程序中，利用 Timer 进行 Task 定时，利用 Led 进行相应的任务响应显示。

1.3.3 项目修改

1) Makefile 文件

```
1 Makefile
3 COMPONENT=BlinkToRadioAppC
2 TINYOS_ROOT_DIR?=../../..
1 include $(TINYOS_ROOT_DIR)/Makefile.include
4
```

图 1-4 源程序 Makefile 文件

由于采用最新的 Make Version 3.0，而不是采用原有 2.1.2 中的 Make 系统，没有了 TINYOSROOT，TINYOSHOME 等环境变量，故在每一个项目中需要修改相应的 TINYOS_ROOT_DIR。为了方便，可以采用系统编写环境变量然后统一使用该环境变量的方法，此处，为了不污染系统环境变量，直接使用绝对路径

赋值，当发生源码拷贝时，注意修改。

为了实现单播，需要指定信道，否则默认为广播模式。

```
hover@wings ~/Desktop/Labs/HUST_IOTCommunicationTechnology_Labs/HUST_IOTCommuication_Labs/Lab1 master ● ccat Makefile
COMPONENT=BlinkToRadioAppC
TINYOS_ROOT_DIR?=/home/hover/tinyos-main

PFLAGS += -DCC2420_DEF_CHANNEL=7

include $(TINYOS_ROOT_DIR)/Makefile.include
```

图 1-5 修改后 Makefile 文件

2) 项目文件

```
enum
{
  AM_BLINKTORADIO = 6,
  TIMER_PERIOD_MILLI = 1000,
  NODE_ID_1 = 1,
  NODE_ID_2 = 2
};
```

图 1-6 指定节点编号及时延

```
event message_t* Receive.receive(message_t* msg, void* payload, uint8_t len)
{
    if (len == sizeof(BlinkToRadioMsg))
    {
        BlinkToRadioMsg* btrpkt = (BlinkToRadioMsg*)payload;
        if(btrpkt->nodeid==NODE_ID_2)
        {
            setLeds(btrpkt->counter);
            counter = btrpkt->counter;
        }
    }
    return msg;
}
```

图 1-7 指定 NODE1 仅仅接受的 NODE2 报文

3) 节点烧入

程序开发完成后将程序烧录进两台不同的节点，一台 ID 为 1，另一台为 2，而后观察实验现象。

1.4 实验结果分析

节点 1,节点 2 都开着的时候，节点 1 和节点 2 的 LED 灯显示的值每隔一秒递增 1，节点 1 和节点 2 的计数值交替显示。按住其中一个节点的 RESERT，另外一个节点的显示将会停住不再变化，放开后又重新从 1 开始计数，表明两个节点根据对方节点的 counter 值维护自身的 counter 值。

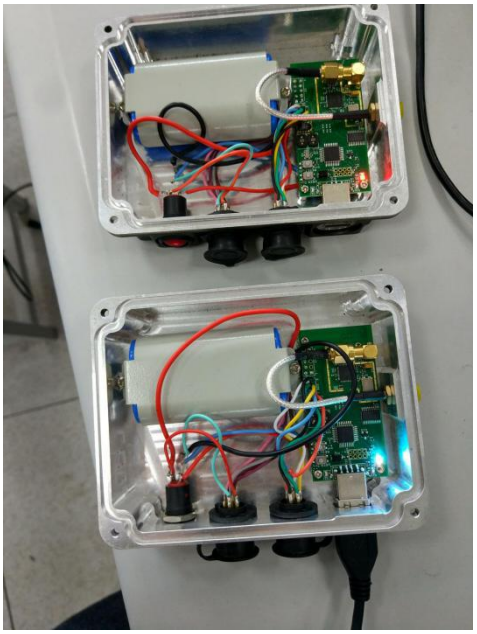


图 1-8 无线单播测试图

1.5 心得体会与总结

因为去年的时候在进行 TinyOS 实验时，已经尝试了 BaseStation 等实验，故实验较为简单，此次实验采用 Make Version 3 编译系统，其中可以清晰的看到 TinyOS 各代码段数据段所占用的大小，故能够更加的体会到 TinyOS 为静态系统，在拷入节点的时候已经完成完全确定，这样的特性，在进行大规模拷贝和在线更新的时候具有一定的优势。

2 节点-PC 串口通信实验(简单基站版本)

2.1 实验目的与要求

本实验的目的是实现节点和 PC 间的串口双向通讯，通过串口连接，从网络收集其他节点的数据，也可以发送数据或者命令到节点，因此，串口通信编程是无线传感器网络中的重要内容。

根据例子提供的例子程序，详细了解程序结构，并尝试进行程序的修改运行；具体实验要求如下：

1) 学会使用串口通信相关的接口函数，实现串口通信。
2) 修改 BlinkToRadio 程序，添加串口收发，实现一个简单的基站功能。
3) 了解串口双向通信的方法，学会使用 mig 工具以及 SerialForwarder。

2.2 实验内容

2.2.1 实验涉及内容

1) TeatSerial 例子程序
2) 基站程序示例
3) MIG 及数据包对象
4) SerialForwarder 和其他数据包源
5) mote 节点向串口发送数据包
6) PC 写串口

2.2.2 实验内容要求

1) 修改 BlinkToRadio 程序实现简单的基站程序 RadioAndSerial．使得当 RadioAndSerial 接收到无线数据包时 Led2 闪烁，并将数据包转发到串口；当串口接收到数据包时，Led0 闪烁，并将数据包转发到无线模块；
2) 使用 mig 创建 BlinkToRadioMsg 的 java 对象，BlinkToRadio 发送的消息由 RadioAndSerial 接收并转发到串口，然后使用 MsgReader 读取 BlinkToRadioMsg 对象，展示接收到的数据。
3) 扩展要求：实现 BlinkToRadio.java，进行 PC 和串口之间的双向通信，从而实现利用 PC 发送消息控制 BlinkToRadio 节点 Led 灯的闪烁。

1．修改 BlinkToRadio 程序实现简单的基站程序 RadioAndSerial．使得当 RadioAndSerial 接收到无线数据包时 Led2 闪烁，并将数据包转发到串口；当串口接收到数据包时，Led0 闪烁，并将数据包转发到无线模块；

2．使用 mig 创建 BlinkToRadioMsg 的 java 对象，BlinkToRadio 发送的消息由 RadioAndSerial 接收并转发到串口，然后使用 MsgReader 读取 BlinkToRadioMsg 对象，展示接收到的数据。

提示步骤：

1.mote 节点的串口读写与无线模块的读写基本一致，可以参考 testSerial 源码

2.本次实验中不需要使用定时器，当收到数据包时即进行转发，不用考虑数据包队列，延迟等.

3.按照 TestSerial 例子修改 makefile。注意，javac 后面应该接一个 tab 字符而不是空格。

4．设置环境变量 MOTECOM 可以减少参数的输入

使用 Mig 工具时如果遇见以下类似信息

```
warning: Cannot determine AM type for BlinkToRadioMsg
 (Looking for definition of AM_BLINKTORADIOMSG)
```

就需要将 AM 类型从 AM_BLINKTORADIO 改为 AM_BLINKTORADIOMSG，这是 mig 命名规则的需要。然后再次编译烧录。

3.BaseStation 节点与修改后的 BlinkToRadio 节点都通电。然后在 BlinkToRadio 目录输入：

```
java net.tinyos.tools.MsgReader -comm serial@/dev/ttyUSB0:telos
BlinkToRadioMsg
```

结果应该类似以下：

```
1152232617609: Message
  [nodeid=0x2]
  [counter=0x1049]

1152232617609: Message
  [nodeid=0x2]
  [counter=0x104a]

1152232617609: Message
  [nodeid=0x2]
  [counter=0x104b]
BlinkToRadio.class: $(wildcard *.java) BlinkToRadioMsg.java
        javac  *.java
```

```
1152232617621: Message
  [nodeid=0x2]
  [counter=0x104c]
```

4．（扩展部分）生成 BlinkToRadio.class 文件，PC 端通过该类收发串口的数据．除了正常收到串口的数据外，还可以从 PC 主动发送数据到基站，由基站转发到节点，可以观察到基站有发送数据的显示，节点收到数据后会亮灯；
类似如下：

```
serial@/dev/ttyUSB2:115200: resynchronising
Input the nodeid and counter(like 1  2): 3 1
Sending to 3 number is : 1
Received packet to: 3 nodeid: 3 counter: 1
Input the nodeid and counter(like 1  2): 2 1
Sending to 2 number is : 1
Received packet to: 1 nodeid: 1 counter: 3
Input the nodeid and counter(like 1  2): 4 1
Sending to 4 number is : 1
Received packet to: 3 nodeid: 4 counter: 1
```

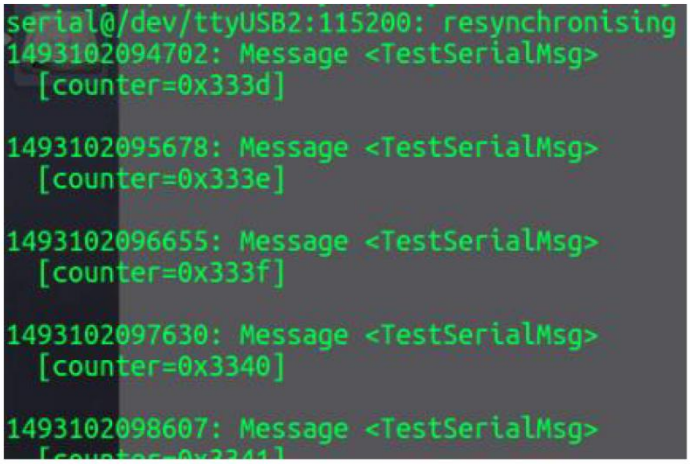
实验过程说明

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 $@
```

申请节点，自动分配为 3 号和 4 号节点，将 3 号节点作为普通节点，烧录 BlinkToRadio 程序，4 号节点作为基站节点，烧录 RadioAndSerial 程序，可以看到 4 号节点的 LED1 灯在不停的闪烁，说明基站节点不停的收到网络包并将其发送至串口。

使用 MsgReader 命令，结果如下图所示：



图中每两行代表基站节点收到的一个网络包，后面显示的是计数值。

2.3 实验过程与结果

2.3.1 开发环境

OS： Manjaro 18.0.2 Illyria(Arch-Based Distribution)
Kernel: x86_64 Linux 4.20.0-1-MANJARO
TinyOS： TinyOS(Make Version 3.0.0)
GCC: gcc (GCC) 7.4.1
nescc: 1.3.6
Text Editor: VisualStudio Code(1.30.1 x64)

2.3.2 TestSerial 例程

1) 例程分析

在系统 Test 目录下面有 TinyOS 的测试程序，并提供了接口测试，此处进行 TestSerial

图 2-1 TestSerial 目录

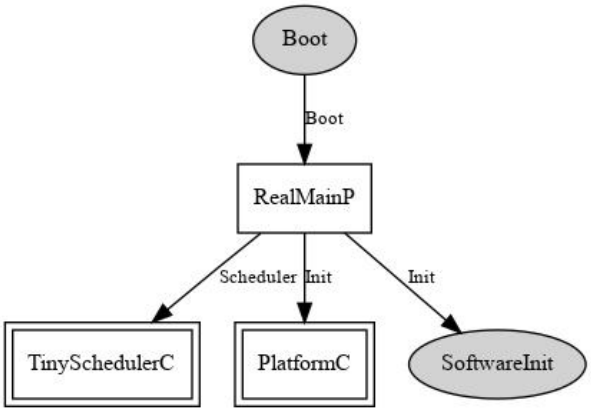


图 2-2 TinyOS 程序结构

系统启动调用 Boot 进行启动，然后初始化 PlatformC 进行配置初始化，然后启动 TinySchedulerC 进行任务监听

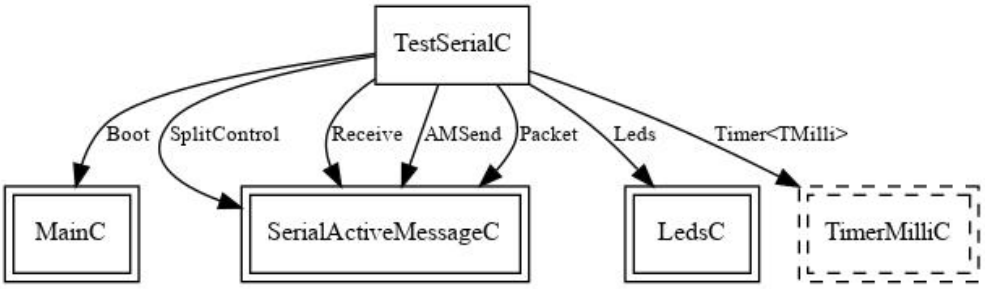


图 2-3 TestSerial 项目结构

TestSerial 接受到消息(可以来自于定时器或者无线消息接受)之后调用 SerialActiveMessageC 组件进行 Task 制定，从而完成类似中断机制的消息传送

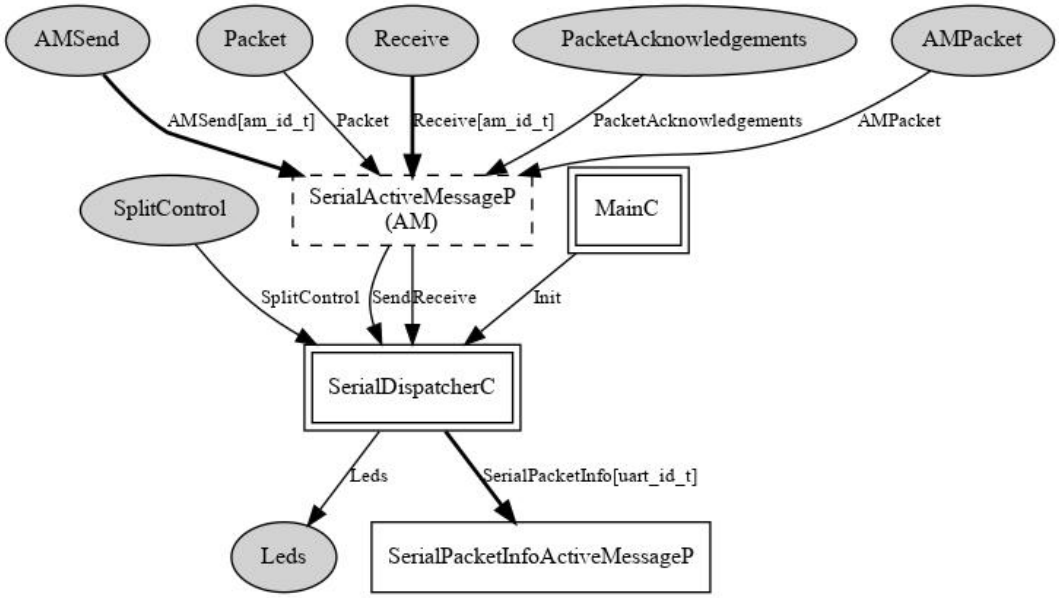


图 2-4 SerialActiveMessageC 组件配置

利用 SerialDispatcherC 进行报文发送和 LED 灯设置

2) 例程烧录

a) 编译烧录程序：

make telosb install /dev/ttyUSB0

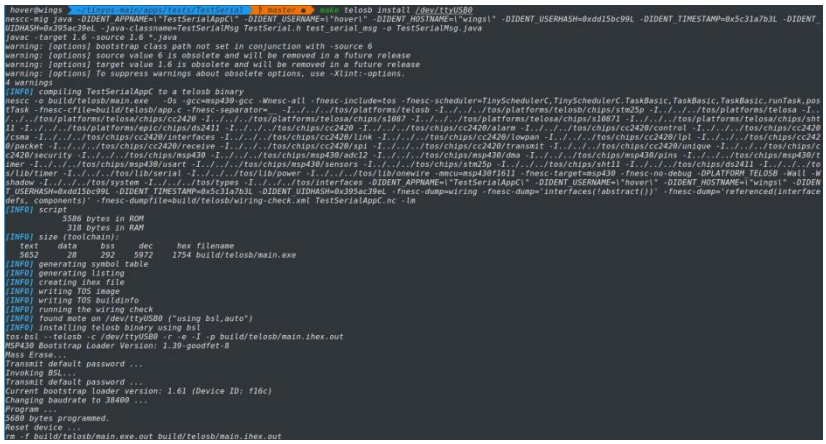


图 2-5 TestSerial 拷贝

b) PC 端串口信息接受程序(Java Verision)：

java TestSerial -comm serial@/dev/ttyUSB0:telosb

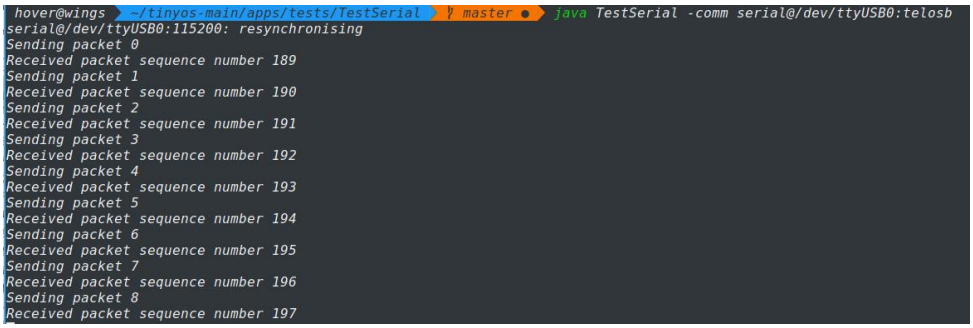
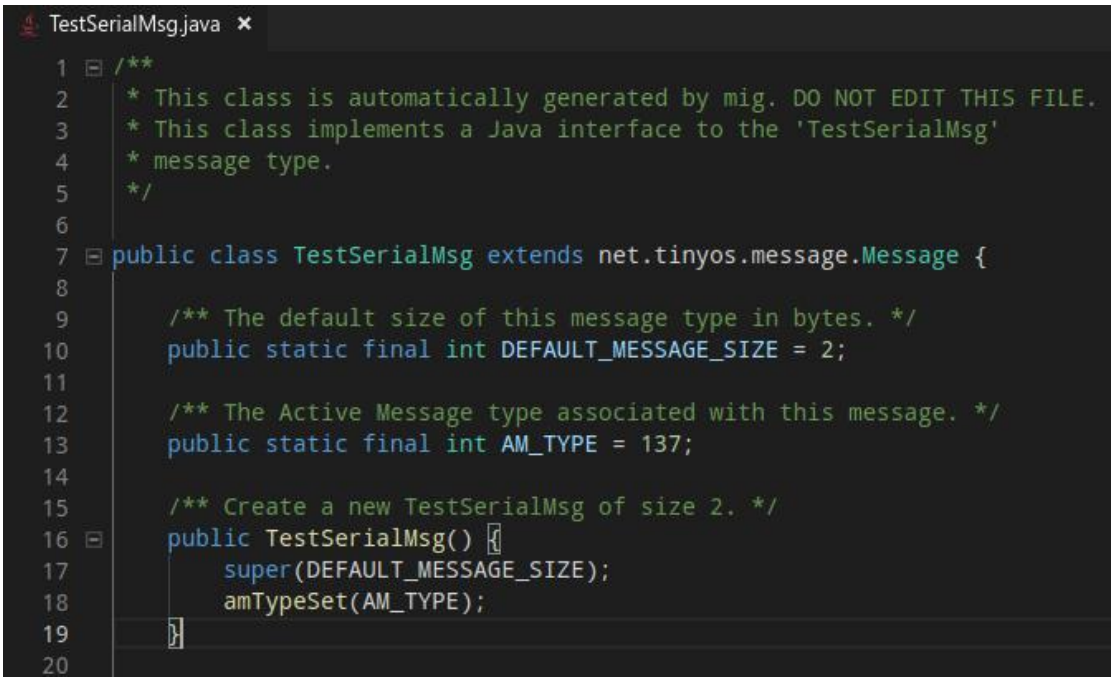


图 2-6 TestSerial PC 端串口程序输出

可以看到输出 seq 号和相应的报文解析



```
TestSerialMsg.java ×

1  /**
2   * This class is automatically generated by mig. DO NOT EDIT THIS FILE.
3   * This class implements a Java interface to the 'TestSerialMsg'
4   * message type.
5   */
6
7  public class TestSerialMsg extends net.tinyos.message.Message {
8
9      /** The default size of this message type in bytes. */
10     public static final int DEFAULT_MESSAGE_SIZE = 2;
11
12     /** The Active Message type associated with this message. */
13     public static final int AM_TYPE = 137;
14
15     /** Create a new TestSerialMsg of size 2. */
16     public TestSerialMsg() {
17         super(DEFAULT_MESSAGE_SIZE);
18         amTypeSet(AM_TYPE);
19     }
20
```

图 2-7 MIG 生成的报文 java 文件

此时 mig 工具会生成对应的报文格式的 java 描述

```
public class TestSerial implements MessageListener {

    private MoteIF moteIF;

    public TestSerial(MoteIF moteIF) {
        this.moteIF = moteIF;
        this.moteIF.registerListener(new TestSerialMsg(), this);
    }

    public void sendPackets() {
        int counter = 0;
        TestSerialMsg payload = new TestSerialMsg();

        try {
            while (true) {
    System.out.println("Sending packet " + counter);
    payload.set_counter(counter);
    moteIF.send(0, payload);
    counter++;
    try {Thread.sleep(1000);}
    catch (InterruptedException exception) {}
            }
        }
        catch (IOException exception) {
            System.err.println("Exception thrown when sending packets. Exiting.");
            System.err.println(exception);
        }
    }
```

图 2-8 报文解析 java 文件

利用报文解析的 java 文件进行报文解析并输出，之后可以对报文解析程序进行相应的修改。

TinyOS 提供了多套报文解析工具，其中有 Java Version 和 Python Version，只要解析程序能够按照特定的格式进行解析就可以了获取数据，这和通常网络中的协议报文规定语法语义类似。

2.3.3 BaseStation 例程

1) 例程分析

```
hover@wings ~/tinyos-main/apps/BaseStation master ● pwd
/home/hover/tinyos-main/apps/BaseStation
```

图 2-9 BaseStation 目录

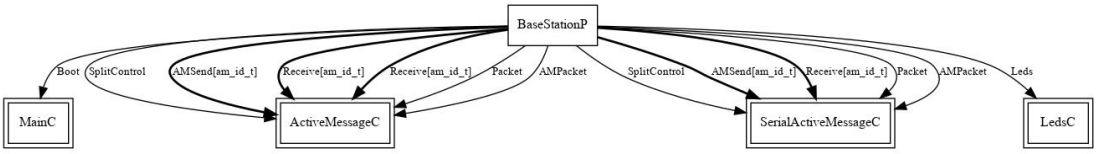


图 2-10 BaseStation 项目结构

同样录用 MainC 进行 Boot 然后调用 ActiveMessageC 进行相应的消息处理，无线消息的发送和接受，利用 SerialActiveMessageC 进行相应的有线(串口)信息处理，利用 LED 进行状态的反馈。

完成如下功能：

a) 在串行链路上，BaseStation 发送和接收简单的活动消息（不是特定的无线数据包）：

b) 在无线电链路上，它发送无线电活动消息，其格式取决于所使用的

网络堆栈。 BaseStation 会将其编译后的组 ID 复制到从串行链路发送到无线电的消息，并过滤掉不包含该组 ID 的传入无线电消息。

c) BaseStation 包括两个方向的队列，保证一旦消息进入队列，它将最终离开另一个接口。 队列允许 BaseStation 处理负载峰值。仅当该消息成功排入以传送到无线电链路时，BaseStation 才会确认通过串行链路到达的消息。

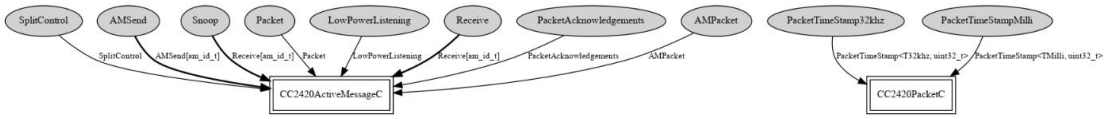


图 2-11 ActiveMessageC 组件配置图

此处调用相应的接口进行请问消息的处理分组和发送，调用 TimeStamp 进行报文的时间戳处理。

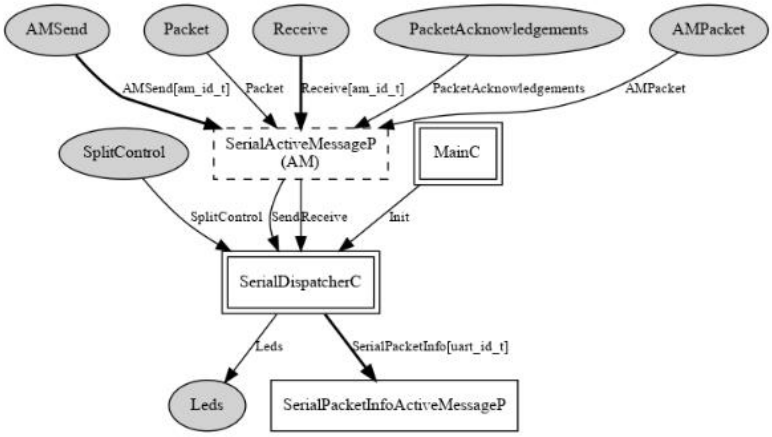


图 2-12 SerialActiveMessageC 组件配置

SerialActiveMessageP 向无线模块提供发送和接受接口，向串口模块提供报文的发送和接受状态，以及 Task 唤醒。

2) 编译烧录

如图左为 BaseStation 节点，节点 LED 不断闪烁，表示接受到报文

右为 BlinkToRadio 节点

图 2-13 BaseStation 烧录结果

```
 hover@wings  ~/tinyos-main/apps/BaseStation  master ●  java net.tinyos.tools.Listen -comm serial@/dev/ttyUSB1:telosb
serial@/dev/ttyUSB1:115200: resynchronising
00 FF FF 00 01 04 22 06 00 01 00 02
00 FF FF 00 01 04 22 06 00 01 00 03
00 FF FF 00 01 04 22 06 00 01 00 04
00 FF FF 00 01 04 22 06 00 01 00 05
00 FF FF 00 01 04 22 06 00 01 00 06
00 FF FF 00 01 04 22 06 00 01 00 07
00 FF FF 00 01 04 22 06 00 01 00 08
00 FF FF 00 01 04 22 06 00 01 00 09
00 FF FF 00 01 04 22 06 00 01 00 0A
00 FF FF 00 01 04 22 06 00 01 00 0B
00 FF FF 00 01 04 22 06 00 01 00 0C
```

图 2-14 PC 串口报文解析

2.3.4 RadioAndSerial 程序

修改 BlinkToRadio 程序

```
module RadioAndSerialC
{
    uses interface Boot;
    uses interface Leds;
    uses interface Timer<TMilli> as Timer0;
    uses interface Timer<TMilli> as Timer1;
    uses interface Packet;
    uses interface AMSend;
    uses interface Receive;

    uses interface Packet as SMPacket;
    uses interface Packet as SMPacket1;
    uses interface AMSend as SMSend;
    //uses interface AMSend as SMSend1;
    uses interface Receive as SMReceive;

    uses interface SplitControl as AMControl; //AMControl
    uses interface SplitControl as SMControl; //SerialControl
}
```

图 2-15 RadioAndSerial 组件配置

```
event message_t* Receive.receive(message_t* msg, void* payload, uint8_t len)
{
    if (busy_s == TRUE) return msg;
    if (len == sizeof(RadioAndSerialMsg))
    {
        RadioAndSerialMsg* btrpkt = (RadioAndSerialMsg*)payload;
        nodeid = btrpkt->nodeid;
        counter = btrpkt->counter;
        //call Leds.led2On();
        call Leds.led2Toggle();
        busy_s = TRUE;
        call Timer0.startOneShot(0);
    }
    return msg;
}

event message_t* SMReceive.receive(message_t* smsg, void* payload, uint8_t len)
{
    if (busy_a == TRUE) return smsg;

    if (len == sizeof(RadioAndSerialMsg))
    {
        RadioAndSerialMsg* btrpkt = (RadioAndSerialMsg*)payload;
        nodeid = btrpkt->nodeid;
        counter = btrpkt->counter;
        call Leds.led0Toggle();
        busy_a = TRUE;
        call Timer1.startOneShot(0);
    }
    return smsg;
}
```

图 2-16 串口和无线模块接受消息处理函数

2.4 实验结果分析

实验结果：烧录终端节点和基站节点，在基站节点程序目录下输入命令 java BlinkToRadio –comm serial@/dev/ttyUSB0:telos，按照提示输入一组数据，如图 2-4 所示，即数据包的目的节点为 3，counter 值为 1，然后计算机通过串口收到来自节点 3 的消息，可知基站节点实现了双向通信。

```
✗ © hover@wings ~/Desktop/Labs/HUST_IOTCommunicationTechnology_Labs/HUST_IOTCommuication_Labs/Lab2/RadioAndSerial ᚠ master ● java BlinkToRadio -comm serial@/dev/ttyUSB3:telosb
serial@/dev/ttyUSB3:115200: resynchronising
Input the nodeid and counter(like 1 2):
1
50
Sending to 1 number is : 50
Input the nodeid and counter(like 1 2):
1 51
Sending to 1 number is : 51
Input the nodeid and counter(like 1 2):
1 52
Sending to 1 number is : 52
Input the nodeid and counter(like 1 2):
1 102
Sending to 1 number is : 102
Input the nodeid and counter(like 1 2):
1 103
Sending to 1 number is : 103
Input the nodeid and counter(like 1 2):
1 55
Sending to 1 number is : 55
Input the nodeid and counter(like 1 2):
```

图 2-17 PC 端口发送消息

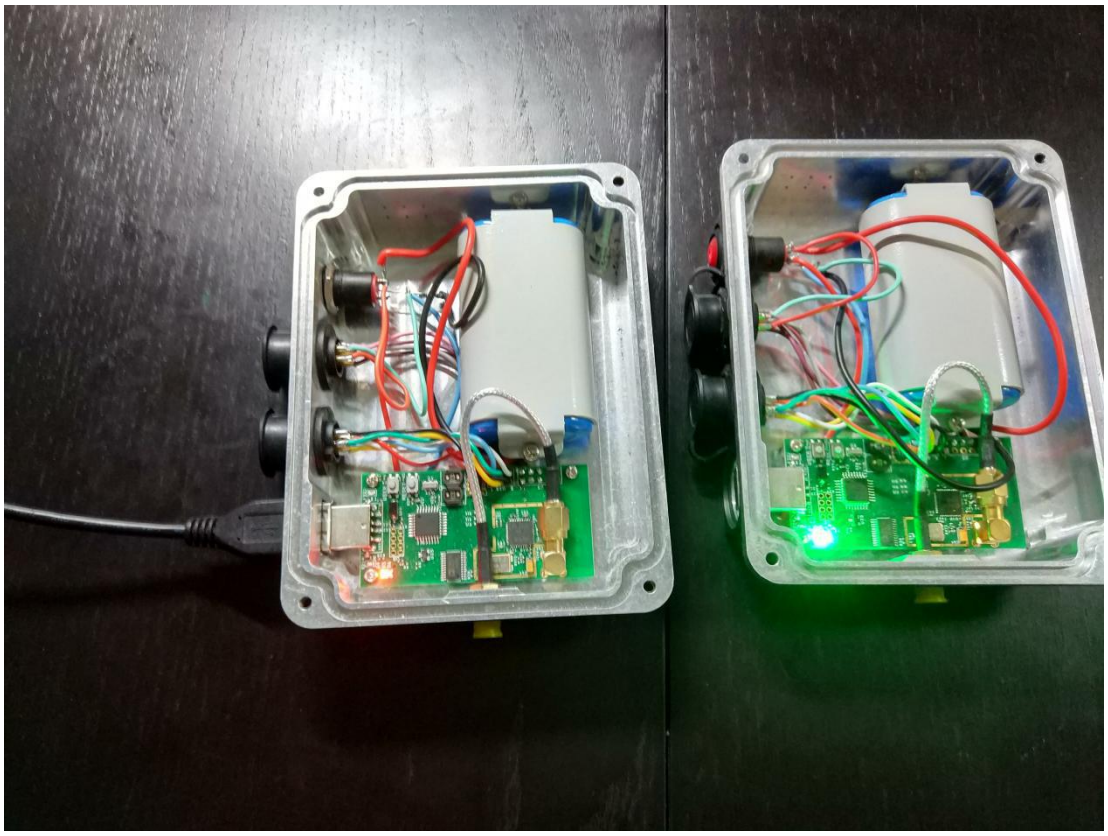


图 2-18 PC 端口发送消息

红灯为基站发送消息，绿灯为节点接受消息。

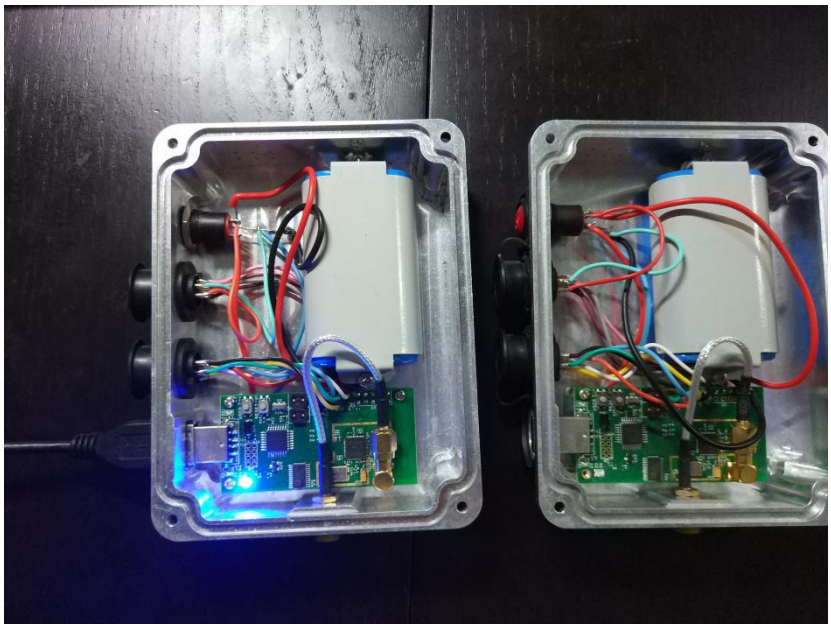


图 2-19 接站接受消息

```
hover@wings ~/Desktop/Labs/HUST_IOTCommunicationTechnology_Labs/HUST_IOTCommunication_Labs/Lab2/RadioAndSerial  master  java net.tinyos.tools.MsgReader -comm ser
ial@/dev/ttyUSB2:telosb BlinkToRadioMsg
serial@/dev/ttyUSB2:115200: resynchronising
1546763225044: Message <BlinkToRadioMsg>
  [nodeid=0x1]
  [counter=0x431]

1546763226032: Message <BlinkToRadioMsg>
  [nodeid=0x1]
  [counter=0x432]

1546763227006: Message <BlinkToRadioMsg>
  [nodeid=0x1]
  [counter=0x433]

1546763227980: Message <BlinkToRadioMsg>
  [nodeid=0x1]
  [counter=0x434]
```

图 2-20 PC 端口监听消息

由实验现象可以看出基站节点实现了从无线转发到串口以及从串口转发到无线的功能，实现了节点-串口双向通信。

2.5 心得体会与总结

1) 实验过程中对 TinyOS 无线组件进行了熟悉
2) 尝试采用两个基站互发消息，此时基站的处理程序直接将消息传给串口，从而可以实现两个 PC 端的无线通信
3) 在烧录节点时记得查看设备号，避免设备弄混，每一次插拔设备，tty 值会加 1
4) 因为未提供 Debug 工具，故只能靠信号灯状态进行判断，实验过程中，出现节点电量较低而无法正常通信的状况，可以将两个节点靠近从而实现通信，在 TinyOS 组件中提供了天线，也可以增加通信距离。

3 路由转发

3.1 实验目的与要求

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。

3.2 实验内容

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 1s, 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;

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。

3.3 实验过程与结果

3.3.1 开发环境

OS： Manjaro 18.0.2 Illyria(Arch-Based Distribution)
Kernel: x86_64 Linux 4.20.0-1-MANJARO
TinyOS： TinyOS(Make Version 3.0.0)
GCC: gcc (GCC) 7.4.1
nescc: 1.3.6
Text Editor: VisualStudio Code(1.30.1 x64)

3.3.2 RadioAndSerial 项目

1) 项目分析

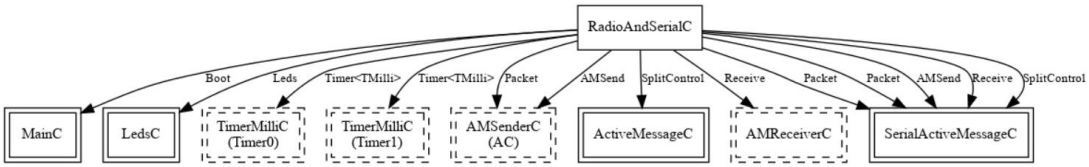


图 3-1 RadioAndSerial 项目架构

利用 Lab2 中的 RadioAndSerial 程序，进行报文的发送和接受，利用指示灯进行状态制定，利用串口通讯工具进行报文的目的地制定，此时地址等同于节点 IP。

在 TinyOS 中提供了一个映射工具，可以将计算机上的端口映射到 TinyOS 串口，从而可以实现不同网络之间的互联(internet 和 zigbee 网络),互联网发送的报文直接发送到 pc 相对应的端口，工具进行报文拆解转换成 TinyOS 通过串口转发给 TinyOS 结点。

3.3.3 BlinkToRadio 项目

1) 项目架构

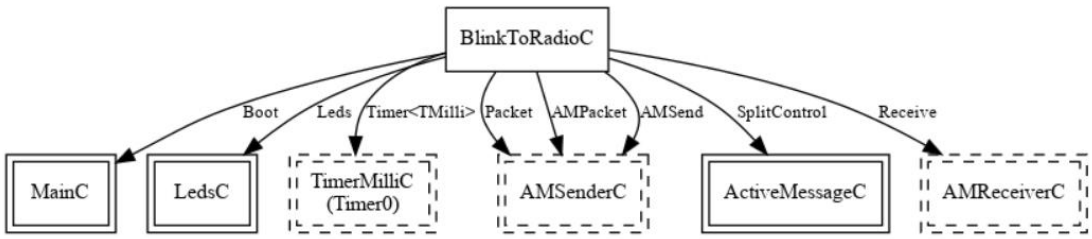


图 3-2 BlinkToRadio 项目架构

整个项目的逻辑并未改变，只不过对于接受报文的 Task 进行了改变不再仅仅是触发 LED，而是修改下一跳结点的 ID,从而实现静态路由，在这一步我们可以看到需要进行报文解析和报文封装过程，同有线网络的路由层次从物理层到网络层再到物理层的过程类似，不过 TinyOS 报文较为简短，且无多层次，故解析报文并不十分耗能。

2) 路由转发逻辑

```
event message_t* Receive.receive(message_t* msg, void* payload, uint8_t len)
{
    if (len == sizeof(BlinkToRadioMsg))
    {
        BlinkToRadioMsg* btrpkt = (BlinkToRadioMsg*)payload;
        BlinkToRadioMsg* trpkt = (BlinkToRadioMsg*)(call Packet.getPayload(&pkt, sizeof(BlinkToRadioMsg)));

        if(TOS_NODE_ID == btrpkt->nodeid)
        {
            trpkt->counter = btrpkt->counter + TOS_NODE_ID;
            trpkt->nodeid = 1;
            setLeds(btrpkt->counter);
            aim_node = TOS_NODE_ID - 1;
            call Timer0.startOneShot(3000);
        }
        else if(btrpkt->nodeid > TOS_NODE_ID)
        {
            trpkt->nodeid = btrpkt->nodeid;
            trpkt->counter = btrpkt->counter;
            setLeds(btrpkt->nodeid);
            aim_node = TOS_NODE_ID + 1;
            call Timer0.startOneShot(1000);
        }
        else if(btrpkt->nodeid < TOS_NODE_ID)
        {
            trpkt->nodeid = btrpkt->nodeid;
            trpkt->counter = btrpkt->counter;
            setLeds(btrpkt->nodeid);
            aim_node = TOS_NODE_ID - 1;
            call Timer0.startOneShot(1000);
        }
    }
    return msg;
```

图 3-3 接受报文并转发逻辑

基站节点 nodeid 为 1，如果接受节点为目标节点，则发送目标节点为基站的反馈报文，如果目标节点大于自身节点编号，下一跳节点编号加 1

如果目标节点小于自身节点编号，下一跳节点编号减少 1

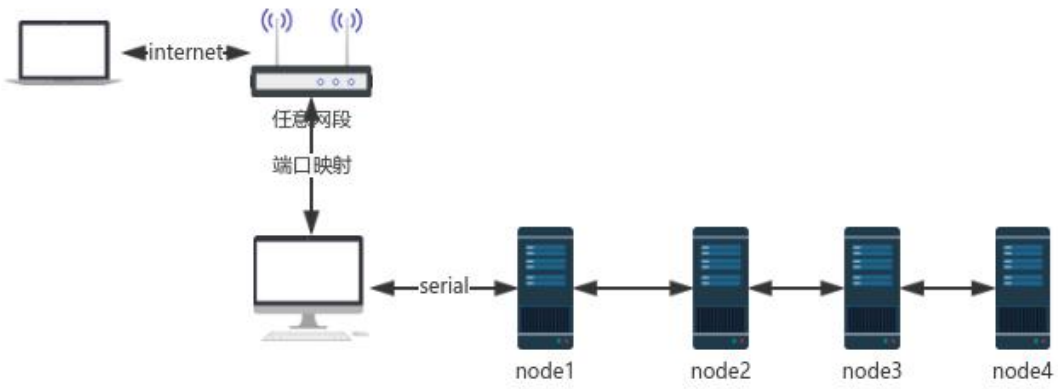


图 3-4 TinyOS 网络架构图

```
event message_t* Receive.receive(message_t* msg, void* payload, uint8_t len)
{
    if (busy_s == TRUE) return msg;
    if (len == sizeof(BlinkToRadioMsg))
    {
        BlinkToRadioMsg* btrpkt = (BlinkToRadioMsg*)payload;
        BlinkToRadioMsg* trpkt = (BlinkToRadioMsg*)(call SMPacket.getPayload(&pkt, sizeof(BlinkToRadioMsg)));
        if(btrpkt->nodeid == TOS_NODE_ID)
        {
            trpkt->counter = btrpkt->counter + TOS_NODE_ID;
            trpkt->nodeid = btrpkt->nodeid;
            busy_s = TRUE;
            setLeds(btrpkt->counter);
            call SMSend.send(trpkt->nodeid, &pkt, sizeof(BlinkToRadioMsg));
            call Timer0.startOneShot(3000);
        }
    }
    return msg;
}

event message_t* SMReceive.receive(message_t* smsg, void* payload, uint8_t len)
{
    if (busy_a == TRUE) return smsg;
    if (len == sizeof(BlinkToRadioMsg))
    {
        BlinkToRadioMsg* btrpkt = (BlinkToRadioMsg*)payload;
        BlinkToRadioMsg* trpkt = (BlinkToRadioMsg*)(call SMPacket1.getPayload(&spkt, sizeof(BlinkToRadioMsg)));
        trpkt->nodeid = btrpkt->nodeid;
        trpkt->counter = btrpkt->counter;
        setLeds(trpkt->nodeid);
        busy_a = TRUE;
        call AMSend.send(trpkt->nodeid, &spkt, sizeof(BlinkToRadioMsg));
        call Timer1.startOneShot(1000);
    }
    return smsg;
}
```

图 3-5 基站转发逻辑

3.4 实验结果分析

从左到右依次为 4,3,2,1(基站)节点

图 3-6 发送给 2 号节点

图 3-7 发送给 3 号节点

图 3-8 4 号节点返回报文 2 号节点中继

我们可以看到在 3 号节点和 2 号结点处进行了一次中继，距离较近时可以直接考虑进行多跳操作，并于其中进行判断，而不是一跳跳传输，在进行静态路由的过程中，可以对于发送的跳数进行设定以满足实际的需求，在编程实现中，还可以对报文进行进一步扩充，记录沿路的路由路径，以达到全局路由的目的，但考虑 TinyOS 的收发能力和路由规定，通常只采用局部路由。

3.5 心得体会与总结

1) 由于实验的要求，此处定时器仅仅作为灯的延时而并未作为任务的延时，在接收到消息之后，对于报文进行解析打包然后发送，而不是在一个灯定时之后进行发送。TinyOS 的 TASK 可以定时执行也可以延时执行
2) 可以看到，在进行长距离通信的时候 TinyOS 在无天线的情况下较弱，必须距离较近，且节点电量对于通信能力有较大影响。
3) 实验过程中，因为插拔导致 ttyUSBx 中 x 值不断改变，且拷贝后也会改变，十分麻烦，不利于拷贝，编写 shell 脚本，先查询 ttyUSBx，然后进行自动编号，从而实现快速拷贝。

4) 本次实验加深了对路由的理解，路由需要拆解报文然后封装报文，完成网络层一下的全部功能，有一个网络层次先上升再下降的过程。
5) 整个实验再一次熟悉了 TinyOS 的编程，将网络知识和操作系统知识结合，对于一个开源的教学系统，能够学会分析协议栈的相关 API 和部分实现原理，能够强化学习新系统的能力，最后，感谢老师的帮助以及所参阅资料的提供者。

参考文献

[1] Shahin Farahani. ZigBee 无线网络与收发器，北京航空航天大学出版社

[2] 潘浩、董齐芬、张贵军.无线传感器网络操作系统 TinyOS，清华大学出版社(2011-08)

[3] TinyOS GitHub 文档.URL: https://github.com/tinyos/tinyos-main

[4] TinyOS 官方 Wiki.URL: http://webs.cs.berkeley.edu/tos/

附录

因为开源代码协议声明，故以下保留协议描述。

对于所参阅的代码基于 Tutorial 则未附上，参见实验过程。

对于无差异部分，参见实验过程。

项目结构如下

```
├── Lab1
│   ├── BlinkToRadioAppC.nc
│   ├── BlinkToRadioC.nc
│   ├── BlinkToRadio.h
│   ├── Makefile
│   └── README.txt
├── Lab2
│   ├── BaseStation
│   │   ├── BaseStationC.nc
│   │   ├── BaseStationP.nc
│   │   ├── BlinkToRadio.java
│   │   ├── Makefile
│   │   └── README.txt
│   ├── BlinkToRadio
│   │   ├── BlinkToRadioAppC.nc
│   │   ├── BlinkToRadioC.nc
│   │   ├── BlinkToRadio.h
│   │   ├── Makefile
│   │   └── README.txt
│   └── RadioAndSerial
│       ├── BlinkToRadio.java
│       ├── Makefile
│       ├── RadioAndSerialAppC.nc
│       ├── RadioAndSerialC.nc
│       ├── RadioAndSerial.h
│       ├── RadioAndSerialMsg.java
│       └── README.txt
├── Lab3
│   ├── BlinkToRadio
│   │   ├── BlinkToRadioAppC.nc
│   │   ├── BlinkToRadioC.nc
│   │   ├── BlinkToRadio.h
│   │   ├── Makefile
│   │   └── README.txt
│   └── RadioAndSerial
│       ├── BlinkToRadioAppC.nc
│       ├── BlinkToRadioC.nc
│       ├── BlinkToRadio.h
│       ├── BlinkToRadio.java
│       ├── Makefile
│       └── README.txt
├── README.md
```

Lab1

BlinkToRadio.h

```
/* FileName:       BlinkToRadio.h
  * Author:          Hover
  * E-Mail:          hover@hust.edu.cn
  * GitHub:          HoverWings
  * Description: BlinkToRadio Program, the message head
  */
// $Id: BlinkToRadio.h,v 1.4 2006-12-12 18:22:52 vlahan Exp $

#ifndef BLINKTORADIO_H
#define BLINKTORADIO_H

enum {
   AM_BLINKTORADIOMSG = 6,
   TIMER_PERIOD_MILLI = 1000
};

typedef nx_struct BlinkToRadioMsg {
   nx_uint16_t nodeid;
   nx_uint16_t counter;
} BlinkToRadioMsg;

#endif
```

BlinkToRadioAppC.nc

```
/* FileName:       BlinkToRadioAppC.nc
  * Author:          Hover
  * E-Mail:          hover@hust.edu.cn
  * GitHub:          HoverWings
  * Description: BlinkToRadio Program, the app configuration and wiring
  */

// $Id: BlinkToRadioAppC.nc,v 1.5 2010-06-29 22:07:40 scipio Exp $

/*
  * Copyright (c) 2000-2006 The Regents of the University    of California.
  * All rights reserved.
  *
```

```
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * - Redistributions of source code must retain the above copyright
 *     notice, this list of conditions and the following disclaimer.
 * - Redistributions in binary form must reproduce the above copyright
 *     notice, this list of conditions and the following disclaimer in the
 *     documentation and/or other materials provided with the
 *     distribution.
 * - Neither the name of the University of California nor the names of
 *     its contributors may be used to endorse or promote products derived
 *     from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 * FOR A PARTICULAR PURPOSE ARE DISCLAIMED.    IN NO EVENT SHALL
 * THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 */
```

```
/**
 * Application file for the BlinkToRadio application.  A counter is
 * incremented and a radio message is sent whenever a timer fires.
 * Whenever a radio message is received, the three least significant
 * bits of the counter in the message payload are displayed on the
 * LEDs.  Program two motes with this application.  As long as they
 * are both within range of each other, the LEDs on both will keep
 * changing.  If the LEDs on one (or both) of the nodes stops changing
 * and hold steady, then that node is no longer receiving any messages
 * from the other node.
 *
 * @author Prabal Dutta
 * @date   Feb 1, 2006
 */
#include <Timer.h>
#include "BlinkToRadio.h"

configuration BlinkToRadioAppC
{

}
implementation
{
    components MainC;
    components LedsC;
    components BlinkToRadioC as App;
    components new TimerMilliC() as Timer0;
    components ActiveMessageC;
    components new AMSenderC(AM_BLINKTORADIO);
    components new AMReceiverC(AM_BLINKTORADIO);

    App.Boot -> MainC;
    App.Leds -> LedsC;
    App.Timer0 -> Timer0;
    App.Packet -> AMSenderC;
    App.AMPacket -> AMSenderC;
    App.AMControl -> ActiveMessageC;
    App.AMSend -> AMSenderC;
    App.Receive -> AMReceiverC;
}
```

BlinkToRadioC.nc

```
/* FileName:       BlinkToRadioC.nc
  * Author:         Hover
  * E-Mail:         hover@hust.edu.cn
  * GitHub:         HoverWings
  * Description:    BlinkToRadio Program, the module configuration and
implementation
  */

// $Id: BlinkToRadioC.nc,v 1.6 2010-06-29 22:07:40 scipio Exp $

/*
  * Copyright (c) 2000-2006 The Regents of the University    of California.
  * All rights reserved.
  *
  * Redistribution and use in source and binary forms, with or without
  * modification, are permitted provided that the following conditions
  * are met:
  *
  * - Redistributions of source code must retain the above copyright
  *      notice, this list of conditions and the following disclaimer.
  * - Redistributions in binary form must reproduce the above copyright
  *      notice, this list of conditions and the following disclaimer in the
  *      documentation and/or other materials provided with the
  *      distribution.
  * - Neither the name of the University of California nor the names of
  *      its contributors may be used to endorse or promote products derived
  *      from this software without specific prior written permission.
  *
  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS
  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,
BUT NOT
  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS
  * FOR A PARTICULAR PURPOSE ARE DISCLAIMED.    IN NO EVENT
SHALL
  * THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
DIRECT,
```

```
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 */

/**
 * Implementation of the BlinkToRadio application.  A counter is
 * incremented and a radio message is sent whenever a timer fires.
 * Whenever a radio message is received, the three least significant
 * bits of the counter in the message payload are displayed on the
 * LEDs.  Program two motes with this application.  As long as they
 * are both within range of each other, the LEDs on both will keep
 * changing.  If the LEDs on one (or both) of the nodes stops changing
 * and hold steady, then that node is no longer receiving any messages
 * from the other node.
 *
 * @author Prabal Dutta
 * @date   Feb 1, 2006
 */
#include <Timer.h>
#include "BlinkToRadio.h"

module BlinkToRadioC
{
  uses interface Boot;
  uses interface Leds;
  uses interface Timer<TMilli> as Timer0;
  uses interface Packet;
  uses interface AMPacket;
```

```
    uses interface AMSend;
    uses interface Receive;
    uses interface SplitControl as AMControl;
}

implementation
{

    uint16_t counter=1;
    message_t pkt;
    bool busy = FALSE;

    void setLeds(uint16_t val)
    {
        if (val & 0x01)
            call Leds.led0On();
        else
            call Leds.led0Off();
        if (val & 0x02)
            call Leds.led1On();
        else
            call Leds.led1Off();
        if (val & 0x04)
            call Leds.led2On();
        else
            call Leds.led2Off();
    }

    event void Boot.booted()
    {
        call AMControl.start();
    }

    event void AMControl.startDone(error_t err)
    {
        if (err == SUCCESS)
        {
            call Timer0.startPeriodic(TIMER_PERIOD_MILLI);
        }
        else
```

```
        {
            call AMControl.start();
        }
    }

    event void AMControl.stopDone(error_t err)
    {
    }

    event void Timer0.fired()
    {
        counter++;
        if (!busy)
        {
            BlinkToRadioMsg* btrpkt = (BlinkToRadioMsg*)(call
Packet.getPayload(&pkt, sizeof(BlinkToRadioMsg)));
            if (btrpkt == NULL)
            {
                return;
            }
            btrpkt->nodeid = TOS_NODE_ID;
            //btrpkt->counter = counter;
            btrpkt->counter = counter+1;
            if (call AMSend.send(AM_BROADCAST_ADDR, &pkt,
sizeof(BlinkToRadioMsg)) == SUCCESS)
            {
                busy = TRUE;
            }
        }
    }

    event void AMSend.sendDone(message_t* msg, error_t err)
    {
        if (&pkt == msg)
        {
            busy = FALSE;
        }
    }

    event message_t* Receive.receive(message_t* msg, void* payload, uint8_t len)
```

```
    {
        if (len == sizeof(BlinkToRadioMsg))
        {
            BlinkToRadioMsg* btrpkt = (BlinkToRadioMsg*)payload;
            if(btrpkt->nodeid==NODE_ID_2)
            {
                setLeds(btrpkt->counter);
                counter = btrpkt->counter;
            }
        }
        return msg;
    }
}
```

Makefile

```
COMPONENT=BlinkToRadioAppC
TINYOS_ROOT_DIR?=/home/hover/tinyos-main

PELAGS += -DCC2420_DEF_CHANNEL = 7

include $(TINYOS_ROOT_DIR)/Makefile.include
```

Lab2

BlinkToRadio.h

```
/* FileName:        BlinkToRadio.h
  * Author:          Hover
  * E-Mail:          hover@hust.edu.cn
  * GitHub:          HoverWings
  * Description: BlinkToRadio Program,the message head
  */
// $Id: BlinkToRadio.h,v 1.4 2006-12-12 18:22:52 vlahan Exp $

#ifndef BLINKTORADIO_H
#define BLINKTORADIO_H

enum
{
   AM_BLINKTORADIO = 6,
   TIMER_PERIOD_MILLI = 1000,
   NODE_ID_1 = 1,
   NODE_ID_2 = 2
};

typedef nx_struct BlinkToRadioMsg
{
   nx_uint16_t nodeid;
   nx_uint16_t counter;
}BlinkToRadioMsg;

#endif
```

BlinkToRadioAppC.nc

```
/* FileName:        BlinkToRadioAppC.nc
  * Author:          Hover
  * E-Mail:          hover@hust.edu.cn
  * GitHub:          HoverWings
  * Description: BlinkToRadio Program, the app configuration and wiring
  */

// $Id: BlinkToRadioAppC.nc,v 1.5 2010-06-29 22:07:40 scipio Exp $

/*
```

```
 *
 */

/**
 * Application file for the BlinkToRadio application.  A counter is
 * incremented and a radio message is sent whenever a timer fires.
 * Whenever a radio message is received, the three least significant
 * bits of the counter in the message payload are displayed on the
 * LEDs.  Program two motes with this application.  As long as they
 * are both within range of each other, the LEDs on both will keep
 * changing.  If the LEDs on one (or both) of the nodes stops changing
 * and hold steady, then that node is no longer receiving any messages
 * from the other node.
 *
 * @author Prabal Dutta
 * @date   Feb 1, 2006
 */
#include <Timer.h>
#include "BlinkToRadio.h"

configuration BlinkToRadioAppC
{

}
implementation
{
    components MainC;
    components LedsC;
    components BlinkToRadioC as App;
    components new TimerMilliC() as Timer0;
    components ActiveMessageC;
    components new AMSenderC(AM_BLINKTORADIO);
    components new AMReceiverC(AM_BLINKTORADIO);

    App.Boot -> MainC;
    App.Leds -> LedsC;
    App.Timer0 -> Timer0;
    App.Packet -> AMSenderC;
    App.AMPacket -> AMSenderC;
    App.AMControl -> ActiveMessageC;
```

```
    App.AMSend -> AMSenderC;
    App.Receive -> AMReceiverC;
}
```

BlinkToRadioC.nc

```
/* FileName:       BlinkToRadioC.nc
  * Author:         Hover
  * E-Mail:         hover@hust.edu.cn
  * GitHub:         HoverWings
  * Description: BlinkToRadio Program, the module configuration and
implementation
  */

// $Id: BlinkToRadioC.nc,v 1.6 2010-06-29 22:07:40 scipio Exp $

/*
  * Copyright (c) 2000-2006 The Regents of the University    of California.
  * All rights reserved.
  *
  * Redistribution and use in source and binary forms, with or without
  * modification, are permitted provided that the following conditions
  * are met:
  *
  * - Redistributions of source code must retain the above copyright
  *     notice, this list of conditions and the following disclaimer.
  * - Redistributions in binary form must reproduce the above copyright
  *     notice, this list of conditions and the following disclaimer in the
  *     documentation and/or other materials provided with the
  *     distribution.
  * - Neither the name of the University of California nor the names of
  *     its contributors may be used to endorse or promote prolsducts derived
  *     from this software without specific prior written permission.
  *
  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS
  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,
BUT NOT
  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS
  * FOR A PARTICULAR PURPOSE ARE DISCLAIMED.    IN NO EVENT
```

```
SHALL
 * THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 */

/**
 * Implementation of the BlinkToRadio application.  A counter is
 * incremented and a radio message is sent whenever a timer fires.
 * Whenever a radio message is received, the three least significant
 * bits of the counter in the message payload are displayed on the
 * LEDs.  Program two motes with this application.  As long as they
 * are both within range of each other, the LEDs on both will keep
 * changing.  If the LEDs on one (or both) of the nodes stops changing
 * and hold steady, then that node is no longer receiving any messages
 * from the other node.
 *
 * @author Prabal Dutta
 * @date   Feb 1, 2006
 */
#include <Timer.h>
#include "BlinkToRadio.h"

module BlinkToRadioC
{
  uses interface Boot;
  uses interface Leds;
```

```
    uses interface Timer<TMilli> as Timer0;
    uses interface Packet;
    uses interface AMPacket;
    uses interface AMSend;
    uses interface Receive;
    uses interface SplitControl as AMControl;
}

implementation
{

        uint16_t counter=1;
        message_t pkt;
        bool busy = FALSE;

        void setLeds(uint16_t val)
        {
                if (val & 0x01)
                        call Leds.led0On();
                else
                        call Leds.led0Off();
                if (val & 0x02)
                        call Leds.led1On();
                else
                        call Leds.led1Off();
                if (val & 0x04)
                        call Leds.led2On();
                else
                        call Leds.led2Off();
        }

        event void Boot.booted()
        {
                call AMControl.start();
        }

        event void AMControl.startDone(error_t err)
        {
                if (err == SUCCESS)
                {
```

```
            call Timer0.startPeriodic(TIMER_PERIOD_MILLI);
        }
        else
        {
            call AMControl.start();
        }
    }

    event void AMControl.stopDone(error_t err)
    {
    }

    event void Timer0.fired()
    {
        counter++;
        if (!busy)
        {
            BlinkToRadioMsg* btrpkt = (BlinkToRadioMsg*)(call
Packet.getPayload(&pkt, sizeof(BlinkToRadioMsg)));
            if (btrpkt == NULL)
            {
                return;
            }
            btrpkt->nodeid = TOS_NODE_ID;
            //btrpkt->counter = counter;
            btrpkt->counter = counter+1;
            if (call AMSend.send(AM_BROADCAST_ADDR, &pkt,
sizeof(BlinkToRadioMsg)) == SUCCESS)
            {
                busy = TRUE;
            }
        }
    }

    event void AMSend.sendDone(message_t* msg, error_t err)
    {
        if (&pkt == msg)
        {
            busy = FALSE;
        }
```

```
    }

    event message_t* Receive.receive(message_t* msg, void* payload, uint8_t len)
    {
        if (len == sizeof(BlinkToRadioMsg))
        {
            BlinkToRadioMsg* btrpkt = (BlinkToRadioMsg*)payload;
            setLeds(btrpkt->counter);
            counter = btrpkt->counter;
        }
        return msg;
    }
}
```

RadioAndSerial.h

```
/* FileName:       RadioAndSerial.h
 * Author:         Hover
 * E-Mail:         hover@hust.edu.cn
 * GitHub:         HoverWings
 * Description: RadioAndSerial Program, the message head
 */

#ifndef RadioAndSerial_H
#define RadioAndSerial_H

enum
{
  AM_RadioAndSerialMSG = 6,
  TIMER_PERIOD_MILLI = 1000
};

typedef nx_struct RadioAndSerialMsg
{
  nx_uint16_t nodeid;
  nx_uint16_t counter;
} RadioAndSerialMsg;

#endif
```

RadioAndSerial.java

```
/* FileName:       RadioAndSerial.java
 * Author:         Hover
 * E-Mail:         hover@hust.edu.cn
 * GitHub:         HoverWings
 * Description: decode the message packet send by TinyOS node throgh serial
 */
import java.io.IOException;
//import java.io.*;
import java.util.Scanner;
import net.tinyos.message.*;
import net.tinyos.packet.*;
import net.tinyos.util.*;

public class RadioAndSerial implements MessageListener
{
    private MoteIF moteIF;
    int counter = 0;
    int nodeid = 0;

    public RadioAndSerial(MoteIF moteIF) {
        this.moteIF = moteIF;
        this.moteIF.registerListener(new RadioAndSerialMsg(), this);
    /*}*/

    /*public void sendPackets() {*/
        RadioAndSerialMsg payload = new RadioAndSerialMsg();
        Scanner in = new Scanner(System.in);

        try {
            while(true) {
                System.out.println("Input the nodeid and counter(like 1 2): ");
                if(in.hasNextInt()){
                    nodeid = in.nextInt();
                    if(nodeid == 0) break;
                    counter = in.nextInt();
                }
                System.out.println("Sending to " + nodeid + " number is : " +
counter);
                payload.set_counter(counter);
```

```
                payload.set_nodeid(nodeid);
                moteIF.send(4, payload);
                try {
                    Thread.sleep(1000);
                }catch(InterruptedException exception){}
            }
        }catch(IOException exception){
            System.err.println("Exception thrown when sending packets. Exiting.");
            System.err.println(exception);
        }
    }

    public void messageReceived(int to, Message message) {
        RadioAndSerialMsg msg = (RadioAndSerialMsg)message;
        System.out.println("Received packet to: " + to + "    nodeid: " +
            msg.get_nodeid() + "    counter: " + msg.get_counter());

    }

    private static void usage() {
        System.err.println("usage: RadioAndSerial [-comm <source>]");
    }

    public static void main(String[] args) throws Exception {
        String source = null;
        if(args.length == 2) {
            if(!args[0].equals("-comm")) {
                usage();
                System.exit(1);
            }
            source = args[1];
        }
        else if(args.length != 0) {
            usage();
            System.exit(1);
        }

        PhoenixSource phoenix;

        if(source == null) {
```

```
            phoenix = BuildSource.makePhoenix(PrintStreamMessenger.err);
        }
        else {
            phoenix = BuildSource.makePhoenix(source,
PrintStreamMessenger.err);
        }

        MoteIF mif = new MoteIF(phoenix);
        RadioAndSerial radio = new RadioAndSerial(mif);
        //radio.sendPackets();
    }
}
```

RadioAndSerialAppC.nc

```
/* FileName:       RadioAndSerialAppC.nc
 * Author:         Hover
 * E-Mail:         hover@hust.edu.cn
 * GitHub:         HoverWings
 * Description: RadioAndSerial Program, the app configuration and wiring
 */

/*
 * Copyright (c) 2000-2006 The Regents of the University  of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * - Redistributions of source code must retain the above copyright
 *     notice, this list of conditions and the following disclaimer.
 * - Redistributions in binary form must reproduce the above copyright
 *     notice, this list of conditions and the following disclaimer in the
 *     documentation and/or other materials provided with the
 *     distribution.
 * - Neither the name of the University of California nor the names of
 *     its contributors may be used to endorse or promote products derived
 *     from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,
BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS
 * FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT
SHALL
 * THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR
```

```
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 */

/**
 * Application file for the RadioAndSerial application.    A counter is
 * incremented and a radio message is sent whenever a timer fires.
 * Whenever a radio message is received, the three least significant
 * bits of the counter in the message payload are displayed on the
 * LEDs.    Program two motes with this application.    As long as they
 * are both within range of each other, the LEDs on both will keep
 * changing.    If the LEDs on one (or both) of the nodes stops changing
 * and hold steady, then that node is no longer receiving any messages
 * from the other node.
 *
 * @author Prabal Dutta
 * @date      Feb 1, 2006
 */
#include <Timer.h>
#include "RadioAndSerial.h"

configuration RadioAndSerialAppC
{

}
implementation
{
      components MainC;
      components LedsC;
      components RadioAndSerialC as App;
      components new TimerMilliC() as Timer0;
      components new TimerMilliC() as Timer1;
```

```
    components ActiveMessageC;
    components SerialActiveMessageC as SM;

    components new AMSenderC(AM_RadioAndSerialMSG) as AC;
    components new AMReceiverC(AM_RadioAndSerialMSG);

    App.Boot -> MainC;
    App.Leds -> LedsC;
    App.Timer0 -> Timer0;
    App.Timer1 -> Timer1;

    App.Packet -> AC;
    App.AMControl -> ActiveMessageC;
    App.AMSend -> AC;
    App.Receive -> AMReceiverC;

    App.SMPacket -> SM;
    App.SMPacket1 -> SM;
    App.SMSend -> SM.AMSend[AM_RadioAndSerialMSG];
    App.SMReceive -> SM.Receive[AM_RadioAndSerialMSG];
    App.SMControl -> SM;

}
/* FileName:       RadioAndSerialC.nc
  * Author:         Hover
  * E-Mail:         hover@hust.edu.cn
  * GitHub:         HoverWings
  * Description: RadioAndSerial Program, the module configuration and
implementation
  */

/*
  * Copyright (c) 2000-2006 The Regents of the University    of California.
  * All rights reserved.
  *
  * Redistribution and use in source and binary forms, with or without
  * modification, are permitted provided that the following conditions
  * are met:
  *
```

```
 * - Redistributions of source code must retain the above copyright
 *     notice, this list of conditions and the following disclaimer.
 * - Redistributions in binary form must reproduce the above copyright
 *     notice, this list of conditions and the following disclaimer in the
 *     documentation and/or other materials provided with the
 *     distribution.
 * - Neither the name of the University of California nor the names of
 *     its contributors may be used to endorse or promote products derived
 *     from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 * FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL
 * THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 */

/**
 * Implementation of the RadioAndSerial application.  A counter is
 * incremented and a radio message is sent whenever a timer fires.
 * Whenever a radio message is received, the three least significant
```

```
 * bits of the counter in the message payload are displayed on the
 * LEDs.  Program two motes with this application.  As long as they
 * are both within range of each other, the LEDs on both will keep
 * changing.  If the LEDs on one (or both) of the nodes stops changing
 * and hold steady, then that node is no longer receiving any messages
 * from the other node.
 *
 * @author Prabal Dutta
 * @date    Feb 1, 2006
 */
#include <Timer.h>
#include "RadioAndSerial.h"

module RadioAndSerialC
{
    uses interface Boot;
    uses interface Leds;
    uses interface Timer<TMilli> as Timer0;
    uses interface Timer<TMilli> as Timer1;
    uses interface Packet;
    uses interface AMSend;
    uses interface Receive;

    uses interface Packet as SMPacket;
    uses interface Packet as SMPacket1;
    uses interface AMSend as SMSend;
    //uses interface AMSend as SMSend1;
    uses interface Receive as SMReceive;

    uses interface SplitControl as AMControl; //AMControl
    uses interface SplitControl as SMControl; //SerialControl
}

implementation
{
    uint16_t counter;
    uint16_t nodeid;
    message_t pkt;              //packet
    bool busy_a = FALSE;    //active message busy tag
```

```
    //uint16_t counter_s;       //send counter
    message_t ppkt;             //packet
    bool busy_s = FALSE;        //serial message busy tag

    message_t spkt;


    event void Boot.booted() {
        call AMControl.start();
        call SMControl.start();
    }

    event void AMControl.startDone(error_t err) {
        while(err != SUCCESS) {
            call AMControl.start();
        }
    }
    event void SMControl.startDone(error_t err) {
        while(err != SUCCESS) {
            call SMControl.start();
        }
    }

    event void AMControl.stopDone(error_t err) {}
    event void SMControl.stopDone(error_t err) {}


    event void Timer0.fired()
    {
        //call Leds.led2Off();
        if(busy_s)
        {
        RadioAndSerialMsg* btrpkt = (RadioAndSerialMsg*)(call
SMPacket.getPayload(&ppkt, sizeof(RadioAndSerialMsg)));
        if (btrpkt == NULL)
            return;
        btrpkt->nodeid = nodeid;
        btrpkt->counter = counter;
        call SMSend.send(AM_BROADCAST_ADDR, &ppkt,
sizeof(RadioAndSerialMsg));
```

```
        }
    }
    event void Timer1.fired()
    {
        if(busy_a)
        {
        RadioAndSerialMsg* btrpkt =
        (RadioAndSerialMsg*)(call SMPacket1.getPayload(&spkt,
sizeof(RadioAndSerialMsg)));
            if (btrpkt == NULL)
                return;
        btrpkt->nodeid = nodeid;
        btrpkt->counter = counter;
        busy_a = FALSE;

        //call AMSend.send(AM_BROADCAST_ADDR, &pkt,
sizeof(RadioAndSerialMsg));
        }
    }

    event void AMSend.sendDone(message_t* msg, error_t err)
    {
        if (&pkt == msg) {
            busy_a = FALSE;
        }
    }
    event void SMSend.sendDone(message_t* msg, error_t err)
    {
        if(&ppkt == msg)
        {
            busy_s = FALSE;
        }
        call Leds.led0Toggle();
    }


    event message_t* Receive.receive(message_t* msg, void* payload, uint8_t len)
    {
        if (busy_s == TRUE) return msg;
        if (len == sizeof(RadioAndSerialMsg))
```

```
        {
            RadioAndSerialMsg* btrpkt = (RadioAndSerialMsg*)payload;
            nodeid = btrpkt->nodeid;
            counter = btrpkt->counter;
            //call Leds.led2On();
            call Leds.led2Toggle();
            busy_s = TRUE;
            call Timer0.startOneShot(0);
        }
        return msg;
    }

    event message_t* SMReceive.receive(message_t* smsg, void* payload, uint8_t
len)
    {
        if (busy_a == TRUE) return smsg;

        if (len == sizeof(RadioAndSerialMsg))
        {
            RadioAndSerialMsg* btrpkt = (RadioAndSerialMsg*)payload;
            nodeid = btrpkt->nodeid;
            counter = btrpkt->counter;
            call Leds.led0Toggle();
            busy_a = TRUE;
            call AMSend.send(btrpkt->nodeid, &spkt, sizeof(RadioAndSerialMsg));
            call Timer1.startOneShot(0);
        }
        return smsg;
    }

}
```

Makefile

```
COMPONENT=BlinkToRadioAppC
TOSMAKE_PRE_EXE_DEPS += BlinkToRadio.class
TOSMAKE_CLEAN_EXTRA = *.class BlinkToRadioMsg.java

BlinkToRadio.class: $(wildcard *.java) BlinkToRadioMsg.java
    javac -target 1.6 -source 1.6 *.java
```

```
BlinkToRadioMsg.java:
	nescc-mig java $(CFLAGS) -java-classname=BlinkToRadioMsg BlinkToRadio.h
BlinkToRadioMsg -o $@


TINYOS_ROOT_DIR?=/home/hover/tinyos-main
include $(TINYOS_ROOT_DIR)/Makefile.include
```

Lab3

BlinkToRadio.h

```
/* FileName:       BlinkToRadio.h
 * Author:         Hover
 * E-Mail:         hover@hust.edu.cn
 * GitHub:         HoverWings
 * Description: BlinkToRadio Program,the message head
 */

// $Id: BlinkToRadio.h,v 1.4 2006-12-12 18:22:52 vlahan Exp $

#ifndef BLINKTORADIO_H
#define BLINKTORADIO_H

enum {
  AM_BLINKTORADIO = 6,
  TIMER_PERIOD_MILLI = 1000
};

typedef nx_struct BlinkToRadioMsg {
  nx_uint16_t nodeid;
  nx_uint16_t counter;
} BlinkToRadioMsg;

#endif
```

BlinkToRadioAppC.nc

```
/* FileName:       BlinkToRadioAppC.nc
 * Author:         Hover
 * E-Mail:         hover@hust.edu.cn
 * GitHub:         HoverWings
 * Description: BlinkToRadio Program, the app configuration and wiring
 */

// $Id: BlinkToRadioAppC.nc,v 1.5 2010-06-29 22:07:40 scipio Exp $

/*
 * Copyright (c) 2000-2006 The Regents of the University  of California.
 * All rights reserved.
 *
```

```
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * - Redistributions of source code must retain the above copyright
 *     notice, this list of conditions and the following disclaimer.
 * - Redistributions in binary form must reproduce the above copyright
 *     notice, this list of conditions and the following disclaimer in the
 *     documentation and/or other materials provided with the
 *     distribution.
 * - Neither the name of the University of California nor the names of
 *     its contributors may be used to endorse or promote products derived
 *     from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 * FOR A PARTICULAR PURPOSE ARE DISCLAIMED.    IN NO EVENT SHALL
 * THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 */
```

```
/**
 * Application file for the BlinkToRadio application.  A counter is
 * incremented and a radio message is sent whenever a timer fires.
 * Whenever a radio message is received, the three least significant
 * bits of the counter in the message payload are displayed on the
 * LEDs.  Program two motes with this application.  As long as they
 * are both within range of each other, the LEDs on both will keep
 * changing.  If the LEDs on one (or both) of the nodes stops changing
 * and hold steady, then that node is no longer receiving any messages
 * from the other node.
 *
 * @author Prabal Dutta
 * @date   Feb 1, 2006
 */
#include <Timer.h>
#include "BlinkToRadio.h"

configuration BlinkToRadioAppC {
}
implementation {
  components MainC;
  components LedsC;
  components BlinkToRadioC as App;
  components new TimerMilliC() as Timer0;
  components ActiveMessageC;
  components new AMSenderC(AM_BLINKTORADIO);
  components new AMReceiverC(AM_BLINKTORADIO);

  App.Boot -> MainC;
  App.Leds -> LedsC;
  App.Timer0 -> Timer0;
  App.Packet -> AMSenderC;
  App.AMPacket -> AMSenderC;
  App.AMControl -> ActiveMessageC;
  App.AMSend -> AMSenderC;
  App.Receive -> AMReceiverC;
}
```

BlinkToRadioC.nc

```
/* FileName:       BlinkToRadioC.nc
  * Author:          Hover
  * E-Mail:          hover@hust.edu.cn
  * GitHub:          HoverWings
  * Description: BlinkToRadio Program, the module configuration and
implementation
  */

// $Id: BlinkToRadioC.nc,v 1.6 2010-06-29 22:07:40 scipio Exp $

/*
  * Copyright (c) 2000-2006 The Regents of the University    of California.
  * All rights reserved.
  *
  * Redistribution and use in source and binary forms, with or without
  * modification, are permitted provided that the following conditions
  * are met:
  *
  * - Redistributions of source code must retain the above copyright
  *      notice, this list of conditions and the following disclaimer.
  * - Redistributions in binary form must reproduce the above copyright
  *      notice, this list of conditions and the following disclaimer in the
  *      documentation and/or other materials provided with the
  *      distribution.
  * - Neither the name of the University of California nor the names of
  *      its contributors may be used to endorse or promote products derived
  *      from this software without specific prior written permission.
  *
  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS
  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,
BUT NOT
  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS
  * FOR A PARTICULAR PURPOSE ARE DISCLAIMED.    IN NO EVENT
SHALL
  * THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
DIRECT,
  * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
```

```
DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 */

/**
 * Implementation of the BlinkToRadio application.  A counter is
 * incremented and a radio message is sent whenever a timer fires.
 * Whenever a radio message is received, the three least significant
 * bits of the counter in the message payload are displayed on the
 * LEDs.  Program two motes with this application.  As long as they
 * are both within range of each other, the LEDs on both will keep
 * changing.  If the LEDs on one (or both) of the nodes stops changing
 * and hold steady, then that node is no longer receiving any messages
 * from the other node.
 *
 * @author Prabal Dutta
 * @date   Feb 1, 2006
 */
#include <Timer.h>
#include "BlinkToRadio.h"

module BlinkToRadioC {
    uses interface Boot;
    uses interface Leds;
    uses interface Timer<TMilli> as Timer0;
    uses interface Packet;
    uses interface AMPacket;
    uses interface AMSend;
    uses interface Receive;
```

```
    uses interface SplitControl as AMControl;
}
implementation {
    //uint16_t counter = 0;
    uint16_t aim_node = 1;
    message_t pkt;
    bool busy = FALSE;

    void setLeds(uint16_t val) {
        if (val & 0x01)
            call Leds.led0On();
        else
            call Leds.led0Off();
        if (val & 0x02)
            call Leds.led1On();
        else
            call Leds.led1Off();
        if (val & 0x04)
            call Leds.led2On();
        else
            call Leds.led2Off();
    }
    void LedsOff() {
        call Leds.led0Off();
        call Leds.led1Off();
        call Leds.led2Off();
    }

    event void Boot.booted() {
        call AMControl.start();
    }

    event void AMControl.startDone(error_t err) {
        while (err != SUCCESS) {
            call AMControl.start();
        }
    }

    event void AMControl.stopDone(error_t err) {
    }
```

```
    event void Timer0.fired() {
        if (!busy) {
            LedsOff();
            if (call AMSend.send(aim_node, &pkt, sizeof(BlinkToRadioMsg)) ==
SUCCESS) {
                busy = TRUE;
            }
        }
    }

    event void AMSend.sendDone(message_t* msg, error_t err) {
        if (&pkt == msg) {
            busy = FALSE;
        }
    }

    event message_t* Receive.receive(message_t* msg, void* payload, uint8_t len)
     {
        if (len == sizeof(BlinkToRadioMsg))
          {
            BlinkToRadioMsg* btrpkt = (BlinkToRadioMsg*)payload;
            BlinkToRadioMsg* trpkt = (BlinkToRadioMsg*)(call
Packet.getPayload(&pkt, sizeof(BlinkToRadioMsg)));

            if(TOS_NODE_ID == btrpkt->nodeid)
              {
                trpkt->counter = btrpkt->counter + TOS_NODE_ID;
                trpkt->nodeid = 1;
                setLeds(btrpkt->counter);
                aim_node = TOS_NODE_ID - 1;
                call Timer0.startOneShot(3000);
            }
            else if(btrpkt->nodeid > TOS_NODE_ID)
              {
                trpkt->nodeid = btrpkt->nodeid;
                trpkt->counter = btrpkt->counter;
                setLeds(btrpkt->nodeid);
                aim_node = TOS_NODE_ID + 1;
                call Timer0.startOneShot(1000);
```

```
            }
            else if(btrpkt->nodeid < TOS_NODE_ID)
              {
                trpkt->nodeid = btrpkt->nodeid;
                trpkt->counter = btrpkt->counter;
                setLeds(btrpkt->nodeid);
                aim_node = TOS_NODE_ID - 1;
                call Timer0.startOneShot(1000);
            }
        }
        return msg;
    }
}
```