Networking Level Laboratory

Mote-Computer Serial Communication

Developed by UNIVERSITY of HOUSTON ♦ UNT UNIVERSITY OF NORTH TEXAS Discover the power of ideas ♦ SAN JACINTO COLLEGE 2009-2012

Purpose/Objective:

The goal of this experiment is to learn how to communicate with a mote node from a PC. This will allow you to collect data from a sensor network, send commands to mote nodes, and monitor the network traffic. Students will also learn the Java-based infrastructure for communicating with motes, and display the collected data as waveform on a graphical user interface (GUI).

Introduction

Being able to talk to a wireless node directly from a computer can greatly increase the application of the wireless sensor network. It not only will allow the users to collect data from a sensor network, but also allows them to control the WSN by sending commands to a particular node or a group of mote nodes. In addition, various programs already exist on computer that monitor the performance of a network can be adapted and used to monitor the wireless communication traffic and the performance of a WSN.

Most wireless sensor nodes provide serial port or similar interface so that they can talk to the serial port in a computer directly. For example, the mica family can directly control a serial port: programming boards basically connect the mote's serial port pins to the actual serial port on the board.

The basic abstraction for mote-PC communication is a **packet source**: a communication medium over which an application can receive packets from and send packets to a mote node. Some examples of packet sources are **serial ports**, **TCP sockets**, and the **SerialForwarder** tool. You can specify which packet source to use by using an optional `–comm` parameter (e.g., `$java net.tinyos.tools.Listen –comm serial@/dev/ttyUSB0:micaz` tells a Listen tool to use the serial port `@/dev/ttyUSB0` (on a Unix machine) at correct speed for a micaz mote.

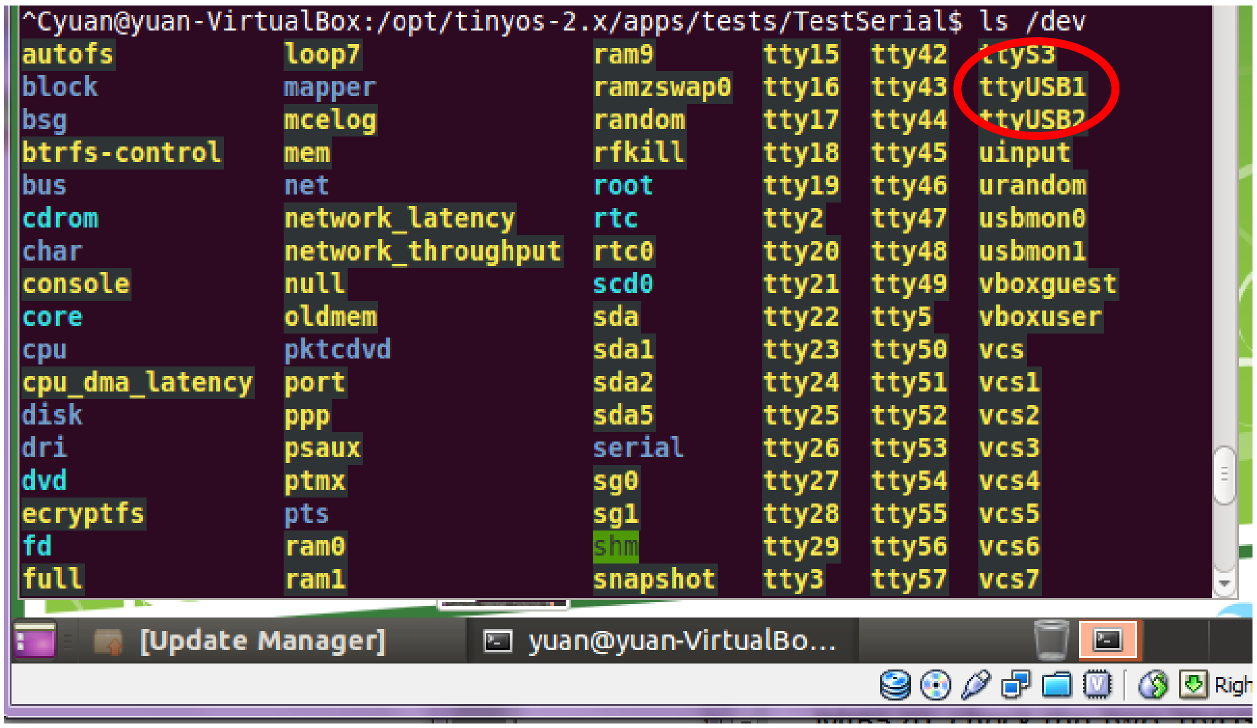
Experiment Procedure:

First we need to test whether the communication between the serial ports of the computer and Mica node functions correctly by using the `apps/tests/TestSerial` application. This application sends a packet to the serial port every second, and when it receives a packet over the serial port it displays the packet's sequence number on the LEDs.

Step 1: Program Mica node with `TestSerial` application:

1. Attach a Mica node securely onto the programming board (MIB520).
2. Change the terminal's active directory to '**/opt/tinyos-2.x/apps/tests/TestSerial**'.
3. Use command '**ls /dev**' to see the list of devices available to your system. For our serial connection from MIB520, check the two 'ttyUSBx' ports. Use the port with lower number (e.g., ttyUSB1 as shown in the figure below) for programming, and the port with higher number (e.g., ttyUSB2) for data reading. Make sure that you gave the control of the MIB520 to the Virtual Box by clicking on 'Devices -> USB Devices -> MEMSIC MIB520CA [0500]'.
4. Use command '**make micaz install.1 mib510,/dev/ttyUSB1**' to download the **`TestSerial`**.
5. Use command '**java TestSerial –comm serial@/dev/ttyUSB2:micaz**' to run the corresponding Java application that communicate with the node over the serial port.

Note: remember the 'serial@/dev/ttyUSB2:micaz' is identical with 'serial@/dev/ttyUSB2:57600', so you can use command '**java TestSerial –comm serial@/dev/ttyUSB2:micaz**' or '**java TestSerial –comm serial@/dev/ttyUSB2:57600** ' to read from the serial ports.



```
^Cyuan@yuan-VirtualBox:/opt/tinyos-2.x/apps/tests/TestSerial$ ls /dev
autofs           loop7               ram9       tty15  tty42  ttyS3
block            mapper              ramzswap0  tty16  tty43  ttyUSB1
bsg              mcelog              random     tty17  tty44  ttyUSB2
btrfs-control    mem                 rfkill     tty18  tty45  uinput
bus              net                 root       tty19  tty46  urandom
cdrom            network_latency     rtc        tty2   tty47  usbmon0
char             network_throughput  rtc0       tty20  tty48  usbmon1
console          null                scd0       tty21  tty49  vboxguest
core             oldmem              sda        tty22  tty5   vboxuser
cpu              pktcdvd             sda1       tty23  tty50  vcs
cpu_dma_latency  port                sda2       tty24  tty51  vcs1
disk             ppp                 sda5       tty25  tty52  vcs2
dri              psaux               serial     tty26  tty53  vcs3
dvd              ptmx                sg0        tty27  tty54  vcs4
ecryptfs         pts                 sg1        tty28  tty55  vcs5
fd               ram0                shm        tty29  tty56  vcs6
full             ram1                snapshot   tty3   tty57  vcs7
```

If you see output like the following and the mote LEDs blink, your communication between mote and computer through serial port is successful!

```
Sending packet 1
Received packet sequence number 4
Sending packet 2
Received packet sequence number 5
Sending packet 3
Received packet sequence number 6
Sending packet 4
Received packet sequence number 7
Received packet sequence number 8
Sending packet 5
Received packet sequence number 9
Sending packet 6
```

Step 2: Use BaseStation application and net.tinyos.tools.Listen tool

BaseStation (in /opt/tinyos-2.x/apps) is an application that acts as a simple Active Message bridge between the serial port and radio links. When it receives a packet from the serial port, it transmits the packet on the radio; when it receives a packet over the radio, it transmits the packet to the serial port. **BaseStation** toggles LED0 (GREEN) whenever it sends a packet to the radio; LED1(YELLOW) whenever it sends a packet to the serial port; and LED2 (RED) whenever it drops a packet. **BaseStation** drops a packet when one of the two receives packets faster than the other can send them (e.g., receiving micaZ radio packets at 256kbps but sending serial packets at 57.6kbps). It can be installed on a Mica node that is attached to the computer via USB and will relay the data broadcasted from other nodes in the sensor network to the computer for further processing.

The Java tool `Listen` is a basic packet sniffer. It prints out the binary contents of any packet it hears. It creates a packet source and prints out every packet it hears.

1. Attach a Mica node securely onto the programming board (MIB520).
2. Change the terminal's active directory to '**/opt/tinyos-2.x/apps/BaseStation**'.
3. Use command '**make micaz install.1 mib510,/dev/ttyUSB1**' to download the **`BaseStation`**. (Note: we are using ttyUSB1 because as shown in the figure before, that's what MIB520 shows in our system.)
4. Turn on the node with `BlinkToRadio` installed (from lab 3). You should see LEDs on the BaseStation node blinking.
5. Use command '**java net.tinyos.tools.Listen –comm serial@/dev/ttyUSB2:micaz**' to run the corresponding Java Listen tool that print out the packets coming from mote nodes over the serial port.

If you see printout similar to the following, then both your BaseStation and computer serial port function successfully!

```
00 FF FF 00 00 04 22 06 00 02 00 01
00 FF FF 00 00 04 22 06 00 02 00 02
00 FF FF 00 00 04 22 06 00 02 00 03
00 FF FF 00 00 04 22 06 00 02 00 04
00 FF FF 00 00 04 22 06 00 02 00 05
00 FF FF 00 00 04 22 06 00 02 00 06
00 FF FF 00 00 04 22 06 00 02 00 07
00 FF FF 00 00 04 22 06 00 02 00 08
00 FF FF 00 00 04 22 06 00 02 00 09
00 FF FF 00 00 04 22 06 00 02 00 0A
00 FF FF 00 00 04 22 06 00 02 00 0B
```

The overall message format for the BlinkToRadioC application is (ignoring the first 00 byte):

- **Destination address** (2 bytes)
- **Link source address** (2 bytes)
- **Message length** (1 byte)
- **Group ID** (1 byte)
- **Active Message handler type** (1 byte)
- **Payload** (up to 28 bytes):
    - **source mote ID** (2 bytes)
    - **sample counter** (2 bytes)

So the data packet can be interpreted as follows:

dest addr	link source addr	msg len	groupID	handlerID	source addr	counter
ff ff	00 00	04	22	06	00 02	00 0B

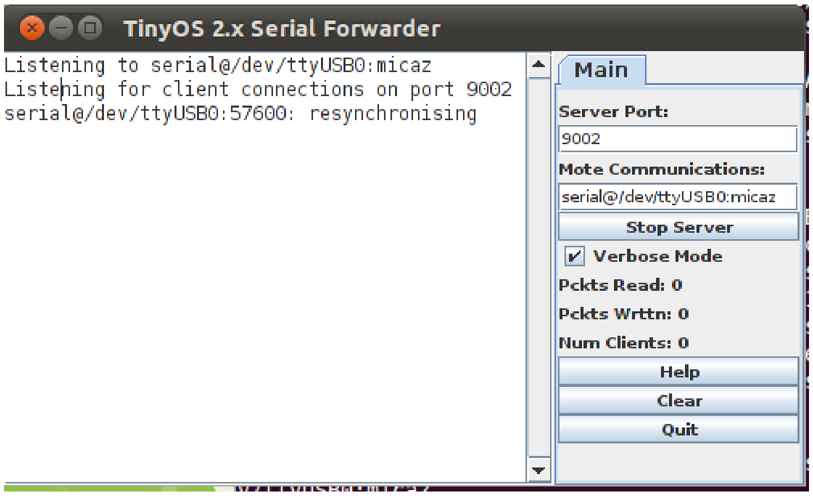
As you watch the packets scroll by, you should see the counter field increases as the BlinkToRadio app increments its counter.

Step 3: Use SerialForwarder tool

Using the serial port directly allows only one PC program to interact with the mote and requires running the application on a PC that is physically connected to the wireless node. The **SerialForwarder** tool is a simple way to address both limitations.

The **SerialForwarder** (**sf**) program opens a packet source and lets many applications connect to it over a TCP/IP stream in order to use that source. For example, you can run a SerialForwarder whose packet source is the serial port. Then many applications can connect to the SerialForwarder, which acts as a proxy, to read and write packets. Since applications connect to SerialForwarder over TCP/IP, they can physically connect over the Internet.

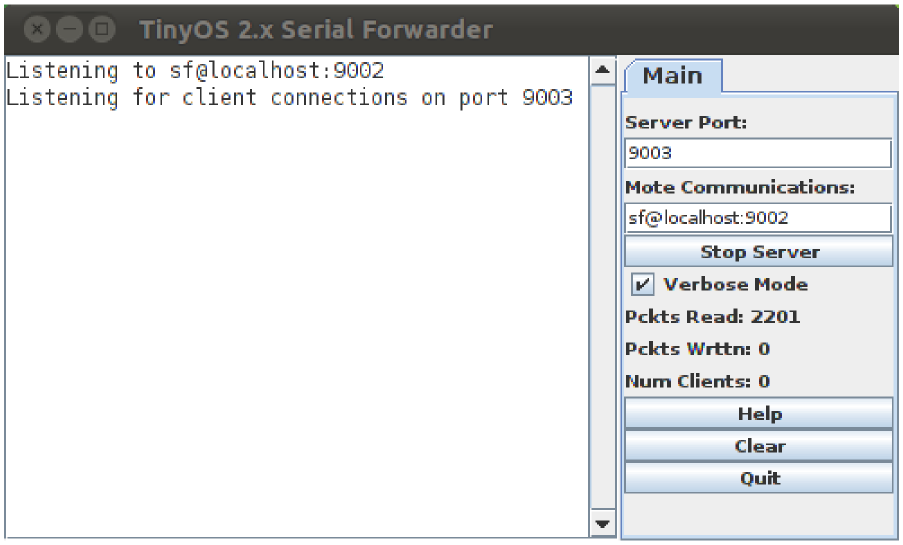
Use command '**java net.tinyos.sf.SerialForwarder -comm serial@/dev/ttyUSB2:micaz**' to connect to a micaz platform using SerialForwarder. You should see a window like the following pops up:



```
Listening to serial@/dev/ttyUSB0:micaz
Listening for client connections on port 9002
serial@/dev/ttyUSB0:57600: resynchronising
```

Server Port: 9002
Mote Communications: serial@/dev/ttyUSB0:micaz
☑ Verbose Mode
Pckts Read: 0
Pckts Wrttn: 0
Num Clients: 0

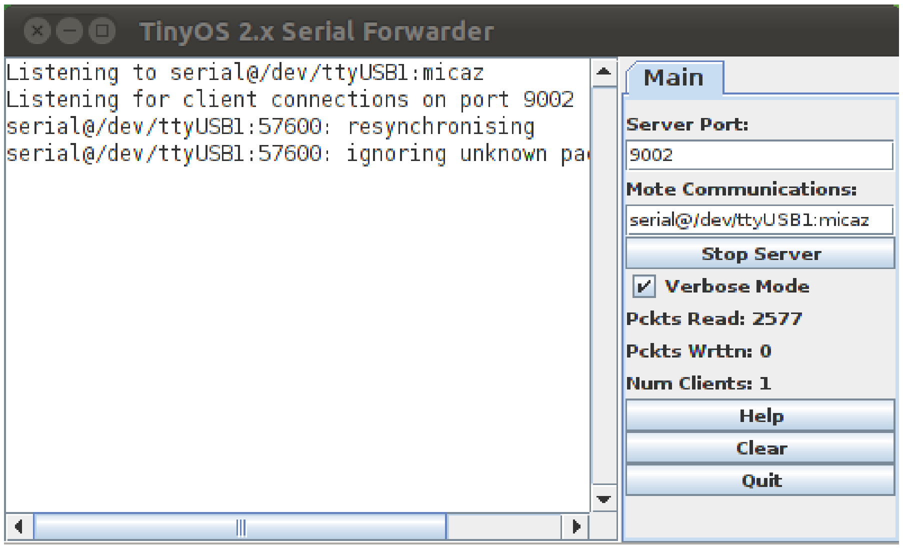
You can choose any port to listen to the data coming in from the serial port. For example, use command '**java net.tinyos.sf.SerialForwarder –port 9003 -comm sf@localhost:9002**' will open a second SerialForwarder as shown in the figure below, whose source is the first SerialForwarder.



```
Listening to sf@localhost:9002
Listening for client connections on port 9003
```

Server Port: 9003
Mote Communications: sf@localhost:9002
[✓] Verbose Mode
Pckts Read: 2201
Pckts Wrttn: 0
Num Clients: 0

You can also see the client count of the first one has increased to one. It demonstrates that in the message support libraries you can use a variety of packet sources.



```
Listening to serial@/dev/ttyUSB1:micaz
Listening for client connections on port 9002
serial@/dev/ttyUSB1:57600: resynchronising
serial@/dev/ttyUSB1:57600: ignoring unknown pa
```

Server Port: 9002
Mote Communications: serial@/dev/ttyUSB1:micaz
[✓] Verbose Mode
Pckts Read: 2577
Pckts Wrttn: 0
Num Clients: 1

Close the second SerialForwarder (the one listening on port 9003).

Step 4: Your turn!

Now it's your turn to test your mastery of the WSN hardware platform and TinyOS software platform.

Modify the BlinkToRadio application so that instead of sending the three least significant bits of the counter to radio, send the packet directly to the serial port.

Exercise Questions:

1. What is the packet source used for our MicaZ platform? What parameter you use to specify such a packet source?

2. What different interfaces to use in order to send message to Radio versus to Serial Port directly?

3. Why SerialForwarder (sf.SerialForwarder) is preferred than directly using the serial port?

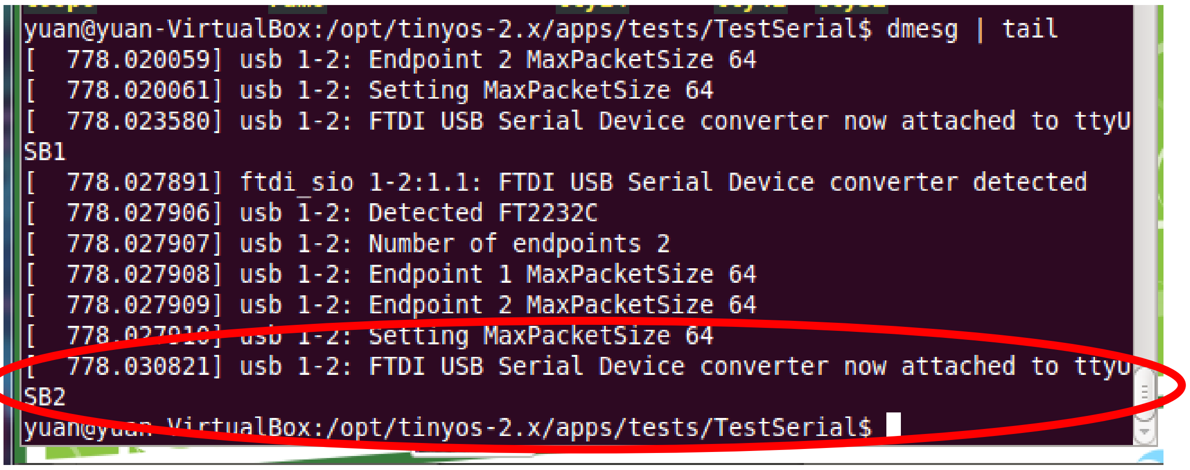
4. What Java tool you use to display received packets on PC?

References:

[1] TinyOS Tutorial Lesson 4: http://docs.tinyos.net/tinywiki/index.php/Mote-PC_serial_communication_and_SerialForwarder

Tips:

1) Use 'Snipping Tool' in the windows system to create snap shots of the lab results in the virtual box environment.
2) Debugging commands that are useful:
   Use **'Dmesg | tail'** to give you a list of device related actions the OS did. Use this to check the USB device you just connected to the computer.



```
yuan@yuan-VirtualBox:/opt/tinyos-2.x/apps/tests/TestSerial$ dmesg | tail
[  778.020059] usb 1-2: Endpoint 2 MaxPacketSize 64
[  778.020061] usb 1-2: Setting MaxPacketSize 64
[  778.023580] usb 1-2: FTDI USB Serial Device converter now attached to ttyU
SB1
[  778.027891] ftdi_sio 1-2:1.1: FTDI USB Serial Device converter detected
[  778.027906] usb 1-2: Detected FT2232C
[  778.027907] usb 1-2: Number of endpoints 2
[  778.027908] usb 1-2: Endpoint 1 MaxPacketSize 64
[  778.027909] usb 1-2: Endpoint 2 MaxPacketSize 64
[  778.027910] usb 1-2: Setting MaxPacketSize 64
[  778.030821] usb 1-2: FTDI USB Serial Device converter now attached to ttyU
SB2
yuan@yuan-VirtualBox:/opt/tinyos-2.x/apps/tests/TestSerial$ 
```