Serial Port, Radio Communication and Serial Forwarder on TinyOS2.x

Qingquan Sun
ECE Department
University of Alabama
Outline

- Mote to Mote Radio Communication
- Mote to PC Serial Port Communication
- Serial Forwarding
Mote-mote Radio Communication

- Basic communication interfaces
  - Packet: accessors for `message_t` abstraction, clearing message’s content, getting payload length
  - Send: sending message function, an event function, getting a pointer to payload area
  - Receive: receiving message function, an event function
Mote-mote Radio Communication

- Active message interfaces
  - AMPacket: AM accessors for message_t abstraction, AM address, AM packet’s destination, AM type
  - AMSend: Active message sending interface
Mote-mote Radio Communication

- Communication components
  - AMReceiverC
  - AMSenderC
  - ActiveMessageC
  - SerialActiveMessageC
  - ActiveMessageAddressC
Active Message

- An asynchronous communication mechanism (user-level handler)
- Integration with communication and computation
- Multiple access to radio
- Effective use of hardware, offer tremendous flexibility
Active Message

App

[AM_id = 1]

GenericComm

AMStandard

Transmitter

AM Handler 1

AM Handler 2

AM Handler 3

Signal [1]

GenericComm

AMStandard

Receiver

Signal sendDone

Signal receive

Radio

Call send

Return buffer
IDs and Addresses

- **Group ID**
  - an 1 byte value specified a virtual network
- **Mote ID**
  - specified using make command
- **Active Message ID**
  - used for distinction among AMs
- **Destination Address**
  - define destination
typedef nx_struct message_t {
    nx_uint8_t header[sizeof(message_header_t)];
    nx_uint8_t data[TOSH_DATA_LENGTH];
    nx_uint8_t footer[sizeof(message_footer_t)];
    nx_uint8_t metadata[sizeof(message_metadata_t)];
} message_t;

- message_t can only be accessed through Packet, AMPacket interfaces
Sending

- Create a header file with a structure to define the message data format

```c
typedef nx_struct BlinkToRadioMsg {
    nx_uint16_t node_id;
    nx_uint16_t counter;
} BlinkToRadioMsg;
```

- Define an unique active message ID

```c
emun {
    AM_BLINKTORADIO = 6
};
```
Sending

1. Define a message structure
2. Assign data to message element
3. Pass message structure to message buffer
4. Specify recipient, message buffer, size
5. Send message buffer to operating system
6. Release message buffer
Receiving

Buffer received message → Verify message structure → Extract data from message payload → Release message buffer → Signal application
<table>
<thead>
<tr>
<th>dest addr</th>
<th>link addr</th>
<th>msg len</th>
<th>group ID</th>
<th>hanwer ID</th>
<th>source addr</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>ff ff</td>
<td>00 00</td>
<td>04</td>
<td>00</td>
<td>06</td>
<td>00 01</td>
<td>00 00</td>
</tr>
</tbody>
</table>
Signal Visualization

PC

Base station

App
Mote-PC Serial Port Communication

- **Substitute** `ActiveMessageC()` **by** `SerialActiveMessageC()`

- Implemented through `support/sdk/java`
Mote-PC Serial Port Communication

```
Terminal - xubuntu@xubuntu-tinyos:/opt/tinyos-2.1.0/apps/tests/TestSerial

Received packet sequence number 191
Received packet sequence number 192
Sending packet 23
Received packet sequence number 193
Sending packet 24
Received packet sequence number 194
Sending packet 25
Received packet sequence number 195
Sending packet 26
Received packet sequence number 196
Sending packet 27
Received packet sequence number 197
Sending packet 28
Received packet sequence number 198
Sending packet 29
Received packet sequence number 199
Sending packet 30
Received packet sequence number 200
Sending packet 31
Received packet sequence number 201
Sending packet 32
Received packet sequence number 202
Sending packet 33
```
Message Interface Generator (MIG)

- The Listen program can only print packet information in hex format
- MIG tool can automatically parse each fields of packets and directly print sensor data
- Built in Makefile
Message Interface Generator (MIG)

COMPONENT=BlinkToRadioMsgC
BUILD_EXTRA_DEPS += BlinkToRadioMsg.class
CLEAN_EXTRA = *.class BlinkToRadioMsg.java

CFLAGS += -I$(TOSDIR)/lib/T2Hack

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

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

include $(MAKERULES)

Tells that BlinkToRadioMsg.class depends on all of the .java file

Tells that how to create BlinkToRadioMsg.java
Message Interface Generator (MIG)
Serial Forwarder

- Packet source: serial port, TCP socket, serial forwarder
- Only one PC program can interact with mote through serial port directly
- Provide multi-access to serial port for multiple applications
- Acts as a proxy
Serial Forwarder

- Get packets from serial port
  - start serial forwarder
    ```java
    java net.tinyos.sf.SerialForwarder
    ```
  - specify the port number
    ```bash
    -comm serial@/dev/ttyUSB1
    ```
Serial Forwarder
Serial Forwarder

- Create a serial forwarder
  - start serial forwarder
    ```
    java net.tinyos.sf.SerialForwarder
    ```
  - specify the port number
    ```
    -port 9003  -comm sf@localhost:9002
    ```
Serial Forwarder

- Read and display data from second serial forwarder source through message reader
Thanks and Questions?