Tutorial & Demo
Image and audio transmission
on wireless sensor networks

CARI 2014 tutorial - PART III
Gaston Berger University
October, 17th, 2014
Saint-Louis, Senegal

Prof. Congduc Pham
http://www.univ-pau.fr/~cpham
Université de Pau, France
Development environments

- Linux-based systems for higher flexibility and better interoperability
  - Most of software tools are targeted for Unix
  - Most of gateways devices are Linux-based (Meshlium, Beagle, Raspberry, ...)
- When possible, avoid Java development and privileged C, or C++ and/or scripts (Shell, Python)
XBEE RADIO MODULE FROM Digi

Implements IEEE 802.15.4 standard

64-bit hardware MAC address

0x0013A200409C0343
Radio (802.15.4) gateways to get data in computer

<table>
<thead>
<tr>
<th>Octets:</th>
<th>2</th>
<th>1</th>
<th>0/2</th>
<th>0/2/8</th>
<th>0/2</th>
<th>0/2/8</th>
<th>0/5/6/10/14</th>
<th>variable</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame Control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC61</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination PAN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3332</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0013A200 40922078</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source PAN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3332</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0013A200 4086D834</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auxiliary Security Header</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame Payload</td>
<td>Leo</td>
<td>FCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC Payload</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MFR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

64-bit 0x0013A2004086D834
16-bit 0x0010
PANID 0x3332

USB-serial converter

Seen as a serial port
/dev/ttyUSB0

Some hardware give access to Link-layer information

Read from /dev/ttyUSB0
Sensor Mote as Gateway

- Program a sensor mote to write to serial port all received data from radio
- Ex: AdvanticSys TelosB with BaseStation-like TinyOs program

Read from /dev/ttyUSB0
FROM serial TO
APPLICATIONS/USERS

- Write custom applications that read the serial port: /dev/ttyUSBx, /dev/ttySx, ...

- Use tools to read serial port and print on standard output, e.g. UNIX stdout
  - Much more flexible, and reliable
  - Can use Unix redirection to feed data into other programs
    - > unix_tool_1 | myapp | unix_tool_2 | unix_tool_3 ...

// get data from radio
data=xbee.getData();

// data = 'hello'
// print data to serial port
Serial.print(data);
SerialToStdout.py

- **Simple python script to read serial port when no translation is needed**
- **Change port and baud rate as needed**

```python
import serial
import sys

ser = serial.Serial('/dev/ttyUSB0', 38400, timeout=0)

# flush everything that may have been received on the port to make sure
# that we start with a clean serial input
ser.flushInput()

while True:
    out = ''
    sys.stdout.write(ser.read(1024))
    sys.stdout.flush()
```
Display_image

- Read standard input for image packet
- Write packet to file
- Read from file and display image

> python SerialToStdout | display_image
XBeeReceive (obsolete)

- Main target is XBee-based gateway
- Translates XBee API frame
- Read from the serial port
  - /dev/ttyUSB0, /dev/ttyS0, ...
- Can write to Unix stdout & can act as a transparent serial replacement
- Display images in image mode
- Reconstructs file in binary mode
- Can act in a data stream fashion: no header for packets
- SerialToStdout.py can be use instead of XBeeReceive with an XBee in transparent mode
XBeeReceive cmd line
(obsolete)

Usage:

./XBeeReceive -baud b -p dev -onlydisplay img_file.dat -pktd -pktf -B/-I -ap0 -v val
-stdout -stream -Q 40 file_name

Usage:
- baud, set baud rate, default is 38400
- p /dev/ttyUSB1
- onlydisplay img_file.dat, display the .dat file only
- pktd, display received XBee frames
- pktf, generate pkt list file
- B/-I, -B for binary mode, -I for image mode, default is image mode
- framing, expects 0xFF0x55 for binary mode, 0xFFx50 for image mode, default is no framing
- ap0, indicates an Xbee in AP mode 0 (transparent mode) so do not decode frame structure
- v 77, use 0x77 to fill in missing value in binary mode
- stdout, write to stdout for pipe mode, don’t work with image mode
- stream, assumes no header & write to stdout for pipe mode in binary mode
- Q 40, use 40 as Quality Factor, default is 50
- file_name, for images: give the original bmp file. for binary: give any file name
XBeeSendCmd

- Uses an XBee gateway to send ASCII string command, e.g. « /@D0030# »

**Usage:**

```
./XBeeSendCmd -p dev [-L][-DM][-at] -tinyos -tinyos_amid id_hex -mac|-net|-addr|-b message
```

- **Usage:** `-p /dev/ttyUSB1`
- **Usage:** `-mac 0013a2004069165d HELLO`
- **Usage:** `-net 5678 HELLO`
- **Usage:** `-addr 64_or_16_bit_addr HELLO`
- **Usage:** `-b HELLO`
- **Usage:** `-at to send remote AT command: -at -mac 0013a2004069165d ATMM`
- **Usage:** `-L insert Libelium API header`
- **Usage:** `-DM to specify DigiMesh firmware`
- **Usage:** `-tinyos to forge a TinyOS ActiveMessage compatible packet (0x3F0x05 are inserted)`
- **Usage:** `-tinyos_amid 6F, to set the ActiveMessage identifier to 0x6F (0x05 is the default)`

**Example:**

- XBeeSendCmd -addr 0013A2004086D835 hello
- XBeeSendCmd -addr 0013A2004086D835 /@Z50#
Image demo
Previous Image sensor motes

iMote2

iMote2 with IMB400 multimedia board
**More generic solution: file sender node**

**Fully configurable:**
- File to send
- Size of packet chunk
- Inter-packet delay
- Image/Binary mode
- Destination node
- Clock synchronization

**Cost:**
~132€
**IMAGE ENCODING**

- Encode any .bmp image file
  - **JPEG encoding** -MSS 90 -Q 5 original_image_file.bmp
  - Specify a quality factor

**Original BMP 128x128**

Collaboration with CRAN laboratory, Nancy, France.
Very robust image encoding techniques against packet losses.
Adjustable Quality Factor

200x200, suitable for Situation-Awareness

Original BMP 40000b

Q=50 S=11045b 142pkts
PSNR=25.1661

Q=40 S=9701b 123pkts
PSNR=24.2231

Q=30 S=8100b 101pkts
PSNR=23.2264

Q=20 S=6236b 76pkts
PSNR=22.1293

Q=15 S=5188b 63pkts
PSNR=21.4475

Q=10 S=3868b 47pkts
PSNR=20.5255

Q=5 S=2053b 24pkts
PSNR=18.937
TRANSMIT IMAGE PACKET
Relay nodes

AdvanticSys TelosB
MicaZ and iMote2

Libelium WaspMote
Arduino Mega2560

Fully configurable:
Destination node
Additional relay delay
Clock synchronization
**Sink node**

- **Linux PC/Laptop with**
  - (a) USB/Serial gateway Xbee radio
  - (b) TelosB as BaseStation
Transmission & Relay Time

Original BMP 40000b
Q=50 S=11045b 142pkts
250kbps: 1.28s
400pkt of 100bytes: 400*0.0115=4.6s
Relay overhead: 400*0.0157=6.28s
PSNR=25.1661

Q=20 S=6236b 76pkts
Q=15 S=5188b 63pkts
250kbps: 0.199s
76pkt of 95bytes: 76*0.0113=0.858s
Relay overhead: 76*0.0145=1.102s
PSNR=22.1293

PSNR=21.4475
Multi-hop forwarding increases pkt loss rate.
Test Bed

T130 transmit with inter pkt time of 130ms
Z50 set the pkt size for binary mode
FdesQ5.dat set the file name to desQ5.dat
D0013A2004086D828 set the 64-bit dest. mac addr
D0080 set the 16-bit dest. mac addr
I or B set to image mode/set to binary mode

All commands must be prefixed by « /@ » and ended/separated by « # »

Examples:
/@T130#, /@FjapanQ20.dat#I#
IMAGE DEMO

0x0013A200408BC81B

Q=20 S=6236b 76pkts

serial Unix tool

> XBeeSendCmd –addr 0013A200408BC81B /@FjapanQ20.dat#I#
>XBeeSendCmd –addr 0013A200408BC81B /@D0013A20040762191#
>XBeeSendCmd –addr 0013A200408BC81B /@T90#

python SerialToStdout | display_image –framing –Q 20 japandisaster-200x200.bmp

0x0013A20040762191
Q=20 S=6236b 76pkts

.format: python SerialToStdout | display_image -framing -Q 20 japandisaster-200x200.bmp
**Image Demo**

0x0013A200408BC81B

---

**Q=20 S=6236b 76pkts**

- XBeeSendCmd –addr 0013A200408BC81B /@D0020#
- XBeeSendCmd –addr 0010 /@D0080#
- XBeeSendCmd –addr 0013A20040762191 /@T90#

- python SerialToStdout | display_image -framing –Q 20 japandisaster-200x200.bmp

---

**serial Unix tool**
AUTOMATIZED RELAY CONFIGURATION

- Shell script to XbeeSendCmd for configuring route
- `relay_conf.sh`

```
> relay_conf.sh /dev/ttyUSB0 /@FjapanQ20.dat#I# 0013A200408BC81B 0020 0010 0080 0013A20040762191
```
Audio demo
Multi-hop audio solution

- Use dedicated audio board for sampling/storing/encoding at 8kbps

- Allows for multi-hop, encoded audio streaming scenarios
the sounds of smart environments

speex at 8kbps

160 8-bit samples (20ms)

20 bytes of encoded audio data

2 bytes framing 0xFF0x55

1 byte Seq. No.

1 byte pkt size (21)

25 or 21 bytes frame

1 byte # samples (20)

speex_sampledec_wframing
1-hop test-bed w/audio board

0x0090

A1/2/3/4 aggregate audio frames
D0100 set the 16-bit dest. mac addr
C0/1 power off/on the audio board

Speex audio encoding
8kbps

Python script,
standard Unix scripting tool

Speex decoding tool from
open-source speex distrib

Play tool from sox
open-source distrib

python 115200SerialToStdout.py | speex_sampledec_wframing | play --buffer 100 -t raw -r 8000 -s 2 -

The sounds of smart environments
2-hop test-bed w/audio board

0x0090
A1/2/3/4/6 aggregate audio frames
0x0020 set the 16-bit dest. mac addr
C0/1 power off/on the audio board

0x0020
D0100 set the 16-bit dest. mac addr

0x0100

Speex audio encoding
8kbps

python script,
standard Unix scripting tool

Speex decoding tool from
open-source speex distrib

Play tool from sox
open-source distrib

Python 115200SerialToStdout.py | speex_sampledec_wframing | play --buffer 100 -t raw -r 8000 -s -2 -

The sounds of smart environments
the sounds of smart environments

**speex at 8kbps**

160 8-bit samples (20ms)

20 bytes of encoded audio data

2 bytes framing
0xFF0x55

1 byte Seq. No.

25 or 21 bytes frame

1 byte pkt size (21)

1 byte # samples (20)

speex_sampledec_wframing

Need to be able to relay a 25-byte pkt every 20ms
Relay node performances

Packet read time & packet relay time

- Packet relay time (th), TelosB
- Packet relay time (measured), TelosB
- Packet read time, TelosB

The sounds of smart environments
Relay node performances

![Image of relay node]

**Read time & processing w/relay time**

<table>
<thead>
<tr>
<th>XBeep payload in bytes</th>
<th>Read time, WaspMote (ms)</th>
<th>WaspMote processing w/relay time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>30</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>40</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>50</td>
<td>30</td>
<td>34</td>
</tr>
<tr>
<td>60</td>
<td>34</td>
<td>38</td>
</tr>
<tr>
<td>70</td>
<td>38</td>
<td>41</td>
</tr>
<tr>
<td>80</td>
<td>41</td>
<td>45</td>
</tr>
<tr>
<td>90</td>
<td>45</td>
<td>50</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

The sounds of smart environments
The sounds of smart environments

**Speex at 8kbps on slow relay nodes**

- 160 bytes (20ms)
- 20 bytes of encoded audio data

Capture 6 audio frames (120ms) but only send 4

Need to be able to relay 100-byte pkt every 120ms

Add framing bytes

A6 aggregate audio frames
Some links


http://web.univ-pau.fr/~cpham/WSN-MODEL/wvsn-castalia.html