Demo & Tutorial: Images and Audio Transmission on Wireless Sensor Networks

C. Pham
Winter School, RIIR, U. Oran
December, 10th, 2013
Oran, Algeria

Prof. Congduc Pham
http://www.univ-pau.fr/~cpham
Université de Pau, France
Search & Rescue, Situation awareness
Dynamic Quality Factor
200x200

Original BMP 40000b  Q=50 S=11045b 142pkts  Q=40 S=9701b 123pkts  Q=30 S=8100b 101pkts

Q=20 S=6236b 76pkts  Q=15 S=5188b 63pkts  Q=10 S=3868b 47pkts  Q=5 S=2053b 24pkts

PSNR=25.1661  PSNR=24.2231  PSNR=23.2264
PSNR=22.1293  PSNR=21.4475  PSNR=20.5255  PSNR=18.937
EAR-IT: audio surveillance in SmartCities and SmartBuildings

See http://www.ear-it.eu
**SmartSantander IoT Node**

- ATmega1281 microcontroller
- 8MHz, 4K RAM & 2G SD card
- 2.4GHz IEEE 802.15.4 Xbee
- Arduino-based IDE

**Images are from Libelium company**

**Gases**
- Carbon Monoxide – CO
- Carbon Dioxide – CO2
- Oxygen – O2
- Methane – CH4
- Hydrogen – H2
- Ammonia – NH3
- Isobutane – C4H10
- Ethanol – CH3CH2OH
- Toluene – C6H5CH3
- Hydrogen Sulfide – H2S
- Nitrogen Dioxide – NO2
- Temperature
- Humidity

**Other Sensors**
- Pressure/Weight
- Bend
- Vibration
- Impact
- Hall Effect
- Tilt
- Temperature (+/-)
- Liquid Presence
- Liquid Level
- Luminosity
- Presence (PIR)
- Stretch
HobNet test-bed at UNIGE

- **MSP430F1611 microcontroller**
- **8Mhz, 48K flash, 10K RAM**
- **2.4GHz IEEE 802.15.4 CC2420**
- **Programmed under TinyOS**
Wireless technologies

- 802.15.4
- 802.20
- 802.16
- 802.16a FWA
- 802.11 WLAN
- 802.15.3a (UWB)
IEEE 802.15.4
Caractéristiques Radio dans les réseaux de capteurs

- Norme ZigBee (IEEE 802.15.4 PHY)

La norme IEEE802.15.4a, adaptées aux réseaux de capteurs, au contrôle industriel et aux dispositifs médicaux (CMI)

IEEE802.15.4 (couches 1 et 2):
- Three bands, 27 channels specified
  - 2.4 GHz: 16 channels, 250 kbps
  - 868.3 MHz : 1 channel, 20 kbps
  - 902-928 MHz: 10 channels, 40 kbps

<table>
<thead>
<tr>
<th>Protocole</th>
<th>Zigbee</th>
<th>Bluetooth</th>
<th>Wi-Fi</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE</td>
<td>802.15.4</td>
<td>802.15.1</td>
<td>802.11a/b/g</td>
</tr>
<tr>
<td>Besoins mémoire</td>
<td>4-32 Kb</td>
<td>250 Kb +</td>
<td>1 Mb +</td>
</tr>
<tr>
<td>Autonomie avec pile</td>
<td>Années</td>
<td>Jours</td>
<td>Heures</td>
</tr>
<tr>
<td>Nombre de nœuds</td>
<td>65 000+</td>
<td>7</td>
<td>32</td>
</tr>
<tr>
<td>Vitesse de transfert</td>
<td>250 Kb/s</td>
<td>1 Mb/s</td>
<td>11-54 et + Mb/s</td>
</tr>
<tr>
<td>Portée</td>
<td>100 m</td>
<td>10-100 m</td>
<td>300 m</td>
</tr>
</tbody>
</table>

- Comparaison entre les normes ZigBee, Bluetooth et Wifi
Spectrum band
MAC FRAME FORMAT

Max size = 127 bytes

<table>
<thead>
<tr>
<th>Octets: 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>Sequence Number</td>
<td>Destination PAN</td>
<td>Destination Address</td>
<td>Source PAN</td>
<td>Source Address</td>
<td>Auxiliary Security Header</td>
<td>Frame Payload</td>
<td>FCS</td>
</tr>
<tr>
<td>CC61</td>
<td>58</td>
<td>3332</td>
<td>0x0013A20040922078</td>
<td>3332</td>
<td>0x0013A2004086D834</td>
<td>0x3332</td>
<td>HELLO</td>
<td>2B32</td>
</tr>
</tbody>
</table>

Max = 102 bytes

MHR

MAC Payload

MFR

---

64-bit 0x0013A2004086D834
16-bit 0x0010
CHANNEL 0x0C
PANID 0x3332

64-bit 0x0013A20040922078
16-bit 0x0020
CHANNEL 0x0C
PANID 0x3332
802.15.4 Gateways

<table>
<thead>
<tr>
<th>Octets: 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>Sequence Number</td>
<td>Destination PAN</td>
<td>Destination Address</td>
<td>Source PAN</td>
<td>Source Address</td>
<td>Auxiliary Security Header</td>
<td>Frame Payload</td>
<td>FCS</td>
</tr>
<tr>
<td>CC61</td>
<td>58</td>
<td>3332</td>
<td>0013A200 4092078</td>
<td>3332</td>
<td>0013A200 4086D834</td>
<td>HELLO</td>
<td>2B32</td>
<td></td>
</tr>
</tbody>
</table>

MHR | MAC Payload | MFR |

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

View as a serial port /dev/ttyUSB0

Some hardware give access to Link-layer information

USB-serial converter

Transparent mode
Or Serial line replacement mode
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, Rasperry, ...)
- **When possible, avoid Java development and priviledge C, or C++ and/or scripts (shell, python)**
Standard Software Tools

- **Libelium WaspMote**
  - Libelium IDE (Arduino-based) & API development environment
- **AdvanticSys TelosB**
  - TinyOS 2.1.2 development environment
- **Audio**
  - Codec2 software (www.codec2.org): c2enc, c2dec
  - Speex software (www.speex.org): speexenc, speexdec
  - sox and play package (Linux)
Customized/dedicated tools

• Serial tools to read host computer serial port
  • XBeeReceive (C LANGUAGE)
  • SerialToStdout (PYTHON SCRIPT)

• Communication tool to send command control packet
  • XBeeSendCmd (C LANGUAGE)

• To get a « pure » SPEEX audio encoded file without any header
  • Modified version of speexdec.c (YES speexdec.c AND NOT speexenc.c) COMPATIBLE WITH SPEEX’S sampledec.c

• Simple « pure » SPEEX audio decoder without any header
  • Modified version of SPEEX’S sampledec.c: speex_sampledec
XBeeReceive

- Main target is XBee-based gateway
- Translates XBee API frame
- Read from the serial port
  - /dev/ttyUSB0, /dev/ttyS0, ...
- Display images in image mode
- Reconstructs file in binary mode (handles packet losses)
  - Assumes each packet with 4 bytes header: 2 bytes for file size & 2 bytes for offset
- Can write to Unix stdout & can act as a transparent serial replacement
- Can act in a data stream fashion: no header for packets
XBeeReceive cmd line

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

USAGE: -p /dev/ttyUSB1

USAGE: -onlydisplay img_file.dat, display the .dat file only

USAGE: -pktd, display received XBee frames

USAGE: -pktf, generate pkt list file

USAGE: -B/-I, -B for binary mode, -I for image mode, default is image mode

USAGE: -framing, expects 0xFF0x55 for binary mode, 0xFFx50 for image mode, default is no framing

USAGE: -ap0, indicates an Xbee in AP mode 0 (transparent mode) so do not decode frame structure

USAGE: -v 77, use 0x77 to fill in missing value in binary mode

USAGE: -stdout, write to stdout for pipe mode, don't work with image mode

USAGE: -stream, assumes no header & write to stdout for pipe mode in binary mode

USAGE: -Q 40, use 40 as Quality Factor, default is 50

USAGE: file_name, for images: give the original bmp file. for binary: give any file name
SerialToStdout.py

• Simple python script to read serial port when no translation is needed
• Change baud rate and port 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()
```

• SerialToStdout.py CAN BE USE INSTEAD OF XBeeReceive WITH AN XBEE IN TRANSPARENT MODE
XBeeSendCmd

XBeeSendCmd

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

Usage:

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

**Usage:**

- `./ XBeeSendCmd -p /dev/ttyUSB1`
- `-mac 0013a2004069165d HELLO`
- `-net 5678 HELLO`
- `-addr 64_or_16_bit_addr HELLO`
- `-b HELLO`
- `-at to send remote AT command: -at -mac 0013a2004069165d ATMM`
- `-L insert Libelium API header`
- `-DM to specify DigiMesh firmware`
- `-tinyos to forge a TinyOS ActiveMessage compatible packet (0x3F0x05 are inserted)`
- `-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
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
Relay nodes

Libelium WaspMote, Imote2, Arduino, TelosB, Micaz

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

Linux PC/Laptop with USB/Serial gateway
Motivations

- Need a controlled environment to
  - Test multi-source scenario
  - Quantify impact of radio interference
  - Test multi-path routing and buffer management for congestion control
  - Know typical latencies

- Adopt a « fully controllable » approach
  - Each node can be dynamically configured...
  - ... to « know » what is going on.
Mote Nodes

R0/1 enable/disable relay mode
D0013A2004086D828 set the 64-bit dest. mac addr
D0080 set the 16-bit dest. mac addr

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#
0x0013A20040762191

Q=20 S=6236b 76pkts

0x0030

0x0060

0x0013A2004086D835

0x0070

QBeeSendCmd –addr 0013A20040762191 /@FjapanQ20.dat#I#
QBeeSendCmd –addr 0013A20040762191 /@D0030#
QBeeSendCmd –addr 0030 /@D0060#
QBeeSendCmd –addr 0060 /@D0013A2004086D835#
QBeeSendCmd –addr 0013A20040762191 /@T90#

QBeeSendCmd –addr 0013A20040762191 /@FjapanQ20.dat#I#
QBeeSendCmd –addr 0013A20040762191 /@D0030#
QBeeSendCmd –addr 0030 /@D0060#
QBeeSendCmd –addr 0060 /@D0013A2004086D835#
QBeeSendCmd –addr 0013A20040762191 /@T90#

QBeeReceive Unix tool

QBeeReceive –I –Q 20 japandisaster-200x200.bmp
Q=20 S=6236b 76pkts

0x0013A20040762191

> XBeeSendCmd --addr 0013A20040762191 /@D0030#
> XBeeSendCmd --addr 0060 /@D0070#
> XBeeSendCmd --addr 0013A20040762191 /@T90#
Audio demo
WaspMote+XBee in raw mode

- Electret mic with amplifier
- XBee in AP0 mode (transparent mode)
- 8-bit 4KHz sampling gives 32000bps
- 8KHz sampling gives 64000bps, requires custom API
Details of pin connection

- VCC on D2
- AUDIO on A2
- GND on GND
void loop() {
    val = analogRead(ANALOG2); // read analog value
    val8bit = ((val >> 2) ); // convert into 8 bit
    // write on UART1, need an XBee module
    // with AP mode 0
    serialWrite(val8bit,1);
}

With XBee GW also in AP0 mode

4KHz sampling
> XBeeReceive -baud 38400 -ap0 -stdout dumb.dat | play --buffer 50 -t raw -r 4000 -u -1 -

8KHz sampling
> XBeeReceive -baud 125000 -ap0 -stdout dumb.dat | play --buffer 50 -t raw -r 8000 -u -1 -

Save raw data for off-line playing
> XBeeReceive -baud 38400 -ap0 -stdout dumb.dat > test.raw
> play -t raw -r 4000 -u -1 test.raw

Alternatively using SerialToStdout python script, at 38400 baud only

> python SerialToStdout | play --buffer 50 -t raw -r 4000 -u -1 -
The receiving XBee module may need to be in packet mode (AP2) due to deployment constraints.

- Adds overhead of XBee API frame decoding: 8KHz sampling may be not supported.

```
4KHz sampling
> XBeeReceive -baud 38400 -stream dumb.dat | play --buffer 50 -t raw -r 4000 -u -1
```

Save raw data for off-line playing
- `XBeeReceive -baud 38400 -stream dumb.dat > test.raw`
- `play -t raw -r 4000 -u -1 test.raw`
Multi-hop audio solution

- Use dedicated audio board for sampling/storing/encoding at 8kbps
- Allows for multi-hop, encoded audio streaming scenarios

Specially designed audio board by INRIA CAIRNS & Feichter Electronics

dsPIC33 with 8kbps speex real-time encoder
speex at 8kbps

160 bytes (20ms)

20 bytes of encoded audio data

2 bytes framing
0xFF0x55

1 byte Seq. No.

24 or 21 bytes frame

1 byte frame size
speex_sampledec
1-hop test-bed w/audio board

0x0090

Speex audio encoding
8kbps

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

0x0100

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

the sounds of smart environments
Relay nodes

Libelium WaspMote

AdvanticSys CM5000, CM3000

Fully configurable:
- Destination node
- Additional relay delay
- Clock synchronization

R0/1 enable/disable relay mode
D0013A2004086D828 set the 64-bit dest. mac addr
D0080 set the 16-bit dest. mac addr

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

Speex audio encoding
8kbps

0x0090
R0/1 enable/disable relay mode
D0100 set the 16-bit dest. mac addr

A1/2/3/4/6 aggregate audio frames
D0200 set the 16-bit dest. mac addr
C0/1 power off/on the audio board

0x0200

0xC823

0x0100

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

Decode & Play
Received audio

the sounds of smart environments
speex at 8kbps requirements

160 bytes (20ms)

20 bytes of encoded audio data

2 bytes framing
0xFF0x55

1 byte Seq. No.

24 or 21 bytes frame

1 byte frame size

speex_sampledec

Need to be able to relay a 24-byte pkt every 20ms
Relay node performances

Packet read time & packet relay time

the sounds of smart environments
Relay node performances

The sounds of smart environments
the sounds of smart environments

Capture 6 audio frames (120ms) but only send 4
Need to be able to relay 96-byte pkt every 120ms

Add framing bytes

A6 aggregate audio frames