Software Radio Implementation of Shortrange Wireless Standards for Sensor Networking

Thomas Schmid, Mani B. Srivastava
{thomas.schmid, mbs}@ucla.edu
Outline

• Introduction to Software Defined Radio
• What is GNU Radio
• The Universal Software Radio Peripheral (USRP)
• Latency measurements of the USRP
• IEEE 802.15.4 and its implementation
• Mica2 FSK and its implementation
• Round Trip Time measurements and its implications on the MAC Layer
What is Software Defined Radio?

Conventional System

Ideal SDR System

What one usually has
Why SDR?

- Dynamic Flexibility (dynamic modulation, MA methods)
- Portability (can run on different hardware)
- Compatibility (interoperation with different hardware)
- Software reuse
- Easy access to development environments will lead to new and better approaches to signal processing
- Advantages by coupling the “radio” with applications
- New spectrum access methods
GNU Radio

- Started in 1998
- Signal processing package distributed under the terms of the GPL
- Began as a fork of Pspectra that was developed at MIT within the SpectrumWare project
- Hybrid Python/C++
- Write simple Python code to wire blocks together into flowgraphs
- Blocks are implemented in C++
- SWIG is used to generate interfaces that connect the Python and C++ code
- FFTW libraries used for FFT
- wxPython for GUIs
Example Code

```python
#!/usr/bin/env python

from gnuradio import gr
from gnuradio import audio

def build_graph():
    sampling_freq = 48000
    ampl = 0.1

    fg = gr.flow_graph()
    src0 = gr.sig_source_f(sampling_freq, gr.GR_SIN_WAVE, 350, ampl)
    src1 = gr.sig_source_f(sampling_freq, gr.GR_SIN_WAVE, 440, ampl)
    dst = audio.sink(sampling_freq)
    fg.connect((src0, 0), (dst, 0))
    fg.connect((src1, 0), (dst, 1))

    return fg

if __name__ == '__main__':
    fg = build_graph()
    fg.start()
    raw_input('Press Enter to quit: ')
    fg.stop()
```
Universal Software Radio Peripheral

- low-cost high speed SDR hardware
- Developed by Matt Ettus
- USB 2 interface to PC
- 4 High-Speed ADCs (64 MS/s)
- 4 High-Speed DACs (128 MS/s)
- Altera Cyclone FPGA
- Different RF frontends (LFRX, BasicRX, RFX400, RFX900, RFX1200, RFX1900, RFX2400)
IEEE 802.15.4

- Low power wireless standard
- Widely used in Sensor Networks
- TMote Sky and MicaZ use the Chipcon CC2420 chip which uses IEEE 802.15.4 Phy (rest of the standard can be added through software libraries)
- Has different bands available, but 2.4GHz is most widely used. We focus on that one.
- Uses Spread Spectrum techniques
- O-QPSK modulation with half-sine pulse shapes ≡ MSK (though different encoding)
IEEE 802.15.4 SDR Block Diagram

**RX**

1. **USRP**
2. **Squelch Filter**
3. **FM Demod**
4. **Clock Recovery**
5. **Packet Sink:**
   - Find start of frame
   - Decode chips to symbols
6. **Python Callback**

**TX**

1. **USRP**
2. **Multiply**
3. **Q-Phase Delay**
4. **QPSK Modulator**
5. **Symbols to Constellation**
6. **Chips to Symbols**
7. **Bytes to Chips**
8. **Packet Source**
9. **Python send_pkt()**
Mica2 FSK SDR Implementation

cc1k_sos_pkt.cc1k_demod_pkts
cc1k.cc1k_demod
fmdemod correlator
ucla.sos_packet_sink
msg_queue
queue_watcher thread
USRP
float soft
symb vector char
python msg object
callback
msg_queue
vector char

Python Code
C Code
RTT in IEEE 802.15.4 SDR implementation

RTT in Conventional IEEE 802.15.4

RTT in FSK SDR implementation

RTT in Conventional FSK
Latency Implications on MAC Layer

- Statistical real-time performance
- We can treat packets which arrive too late as lost. It becomes a new source of packet loss and shouldn’t do much harm as long as it is not big.
- SDR Hardware latencies are too big for conventional wireless protocols
  - IEEE 802.11: SIFS: 10μs, DIFS: 50 μs
  - IEEE 802.15.4: RX-TX turnaround time: 12 symbol periods = 192 μs
  - RX-TX turnaround time for IEEE 802.15.4 SDR implementation: 26 ms on average ~ 70 symbol periods
Theoretic Impact on IEEE 802.11

802.11 Throughput Impact for Longer Interframe Spacings

Throughput [Mbps]

SIFS time [us]

DIFS time [us]
Increasing Interframe Spacing

802.15.4 Throughput Impact for Longer Interframe Spacings

Throughput [kbps]

Packet Rate [Hz]

RX-TX Turnaround Time [symbol periods]
And How to Fix Them?

• Decrease hardware latency by using different bus system (in planing)
• Decrease software latency through parallelization (in work, new scheduler for GNU Radio)
• RSSI gating: instruct the hardware that it should send out the samples if the RSSI measurements show a certain criteria
• Investigate in protocols which do not need hard deadlines
  • TDMA, has hard deadlines but they are longer periods
  • Delayed ACK, i.e., if the channel is free when you are ready to send ACK, send it. If it is busy (other node acquired it), wait until transmission is done and send it then
GNU Radio & Sensor Data?

• What else could GNU Radio do?

• SDR and cognitive radios are part of a bigger group, the cognitive dynamic systems

• Stream processing made easy?
Hypothetical Example

Diagram showing Microcontroller, GNU Radio, Camera, Microphone, Ozone, and SensorBase connected through GNU Radio nodes.
How Could GNU Radio Look?

Node

Microserver / Sensor Base

GNU Radio

Audio Source → Sound Pressure
Image Source → MPEG Encode
Ozone Source → LP Filter

Packet Sink

Packet Source

Picture Correlator
Encryption
Compression
Packet / Sensor Base Sink
Conclusion

• SDR system get a lot of academic and industrial attention lately (and I hope you understand now why)
• We started off with two low power radio implementations
• The implementations allow us to do further research on different aspects of SDR
• We did a performance study of our hardware, though we need to extend it to get more details on where the bottle necks are and where CPU cycles are burnt
• Investigate in other uses of GNU Radio
Questions?

Discussion...