GNU Radio Meets Scapy

Bastian Bloessl

mail@bastibl.net

www.bastibl.net

February 2019  ▪  FOSDEM SDR Dev Room  ▪  Brussels, Belgium
WLAN and ZigBee Transceiver

IEEE 802.11a/g/p – WiFi

WIFI networks, which are already ubiquitous today, will soon spread even further by providing the base for inter-vehicular communication systems. To study these networks, we provide a complete physical layer implementation, including a tool chain for simulation and experimentation.

IEEE 802.15.4 – ZigBee

Energy-efficient wireless networks have many important applications in health care, industrial automation, and form the base for an Internet of Things. We provide a complete IEEE 802.15.4 stack that is interoperable with the ContikiOS up to LoWPAN.
Instant GNU Radio

https://github.com/bastibl/instant-gnuradio
Instant GNU Radio

https://github.com/bastibl/instant-gnuradio
Instant GNU Radio

- Built with Packer
- VM Image
- Many applications pre-installed
- Easy to extend and customize

Ubuntu  →  SDR Apps  →  Your stuff?
Import the VM

The VM comes as an OVA file, i.e., in Open Virtualization Format. That means it is a whole appliance, comprising VM setup and the disk image.
GNU Radio WLAN/ZigBee

- PHY only (MAC adds only static wrapper)
- No CSMA, no ACKs, no network stack
- How can I send data?
- How can I interact with devices?
A GNU Radio Transceiver

“hello world!”

MAC

PHY

Message Strobe
Message PMT: hello world!
Period (ms): 300

WiFi MAC
SRC MAC: 35, 35,...35, 35, 35
DST MAC: 66, 66,...66, 66, 66
BSS MAC: 255, 25,..., 255, 255

WiFi PHY Hier
bandwidth: 10M
chan_est: 0
encoding: 2
frequency: 5.89G
sensitivity: 560m

Packet Pad2
Debug: Disable
Delay: Disable
Delay Sec: 1m
Pad Front: 500
Pad Tail: 500

Channel Model
Noise Voltage: 79.4328m
Frequency Offset: 0
Epsilon: 1
Taps: 1
Seed: 0
A Simple WLAN Frame

"hello world!"

MAC

PHY
A Simple WLAN Frame

Socket PDU
Type: UDP Server
Host: 127.0.0.1
Port: 52001
MTU: 10k

WiFi MAC
SRC MAC: 35, 35,...,35, 35, 35
DST MAC: 66, 66,...,66, 66, 66
BSS MAC: 255, 25,...,255, 255

File Sink
File: /tmp/wifi.pcap
Unbuffered: On
Append file: Append

Wireshark Connector
Technology: WiFi
Debug: Disable

WiFi PHY Hier
bandwidth: 10M
chan_est: 0
encoding: 2
frequency: 5.89G
sensitivity: 560m

Packet Pad2
Debug: Disable
Delay: Disable
Delay Sec: 1m
Pad Front: 500
Pad Tail: 500

Channel Model
Noise Voltage: 79.4328m
Frequency Offset: 0
Epsilon: 1
Taps: 1
Seed: 0
Connecting to UDP Socket

- Netcat

```
31 nc -u 127.0.0.1 52001
```

- Python

```
1 #!/usr/bin/env python
2
3 import socket
4
5 MESSAGE = "Hello, from python!\n"
6
7 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
8 sock.sendto(MESSAGE, ("127.0.0.1", 52001))
```
A Simple WLAN Frame

<data from UDP>  MAC  PHY
Connecting to the Network Stack

- **TUNTAP PDU**
  - Interface Name: tap0
  - MTU: 440
  - Flag: TAP(Ethernet Frame)

- **Ethernet Encapsulation**
  - Debug: Disable

- **WiFi MAC**
  - SRC MAC: 35, 35,..., 35, 35
  - DST MAC: 66, 66,..., 66, 66
  - BSS MAC: 255, 25..., 255, 255

- **File Sink**
  - File: /tmp/wifi.pcap
  - Unbuffered: On
  - Append file: Append

- **Wireshark Connector**
  - Technology: WiFi
  - Debug: Disable

- **WiFi PHY Hier**
  - bandwidth: 10M
  - chan_est: 0
  - encoding: 2
  - frequency: 5.89G
  - sensitivity: 560m

- **Packet Pad2**
  - Debug: Disable
  - Delay: Disable
  - Delay Sec: 1m
  - Pad Front: 500
  - Pad Tail: 500

- **Channel Model**
  - Noise Voltage: 79.4328m
  - Frequency Offset: 0
  - Epsilon: 1
  - Taps: 1
  - Seed: 0

---

GNU Radio meets Scapy // Bastian Bloessl
Connecting to the Network Stack

... IP  MAC  PHY
Virtual WLAN Device

User

Network stack

Driver

WLAN Card

Kernel

cfg80211

mac80211

Virtual WLAN

GNU Radio meets Scapy // Bastian Bloessl
Packet Crafting

- Ethernet frame (the hard way)

```python
frame = struct.pack("!6s6sH",
                 binascii.unhexlify(dst.replace("::","")),
                 binascii.unhexlify(src.replace("::","")),
                 protocol)
```
Scapy

- Python turned into a domain-specific language
- Open Source
- [https://scapy.net/](https://scapy.net/)

```python
frames = Ether(dst="ff:ff:ff:ff:ff:ff")
/IP(dst=["gnuradio.org","libvolk.org"],ttl=(1,9))
/UDP(sport=12345, dport=52001)/"Hello World!"

list(frames)[0].show()
```
Packet Crafting

```python
frames = Ether(dst="ff:ff:ff:ff:ff:ff") / IP(dst=["gnuradio.org","libvolk.org"],ttl=(1,9)) / UDP(sport=12345, dport=52001)="/Hello World!"

list(frames)[0].show()
```

### Ethernet
- **dst** = ff:ff:ff:ff:ff:ff
- **src** = 08:00:27:d0:52:2e
- **type** = 0x800

### IP
- **version** = 4
- **ihl** = None

### UDP
- **sport** = 12345
- **dport** = 52001
- **len** = None
- **chksum** = None

### Raw
- **load** = 'Hello World!'
Advantages

- More flexibility
  - Drivers
  - No device configuration

- No Prototypes (802.11p)

- More accessible (ZigBee)
Flow Graph with Scapy

Socket PDU
Type: UDP Server
Host:
Port: 52001
MTU: 10k

WiFi PHY Hier
bandwidth: 10M
chan_est: 0
encoding: 2
frequency: 5.89G
sensitivity: 560m

Packet Pad2
Debug: Disable
Delay: Disable
Delay Sec: 1m
Pad Front: 500
Pad Tail: 500

Channel Model
Noise Voltage: 79.4328m
Frequency Offset: 0
Epsilon: 1
Taps: 1
Seed: 0
WLAN Frames

- **Beacon frame**

```python
frame = Dot11FCS(addr1='ff:..', addr2='23:..', addr3='23:..')
/Dot11Beacon()/Dot11Elt(ID='SSID', info='GR WLAN')
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.sendto(bytes(frame), ("127.0.0.1", 52001))
```

- **Fuzzing**

```python
frame = Dot11FCS(addr1='ff:..', addr2='23:..', addr3='23:..')
/fuzz(Dot11Beacon()/Dot11Elt(ID='SSID', info='GR WLAN'))
```

- **Deauth**

```python
frame = Dot11FCS(addr1=client, addr2=bssid, addr3=bssid)
/ Dot11Deauth()
```
## Smart Meter

### GNU Radio meets Scapy // Bastian Bloessl

![Smart Meter Image](image)

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>2239</td>
<td>526.623187</td>
<td>0xee64</td>
<td>0x0000</td>
<td>ZigBee</td>
</tr>
<tr>
<td>2240</td>
<td>526.626813</td>
<td></td>
<td></td>
<td>IEEE 802.15.4</td>
</tr>
<tr>
<td>2241</td>
<td>527.249026</td>
<td></td>
<td></td>
<td>IEEE 802.15.4</td>
</tr>
<tr>
<td>2242</td>
<td>527.270500</td>
<td>0xee64</td>
<td>0x0000</td>
<td>ZigBee</td>
</tr>
<tr>
<td>2243</td>
<td>527.337580</td>
<td>0xee64</td>
<td>0x0000</td>
<td>ZigBee</td>
</tr>
<tr>
<td>2244</td>
<td>527.371972</td>
<td></td>
<td></td>
<td>IEEE 802.15.4</td>
</tr>
<tr>
<td>2245</td>
<td>527.375773</td>
<td>0xee64</td>
<td>0x0000</td>
<td>ZigBee</td>
</tr>
<tr>
<td>2246</td>
<td>527.412023</td>
<td>0xee64</td>
<td>0x0000</td>
<td>ZigBee</td>
</tr>
<tr>
<td>2247</td>
<td>527.436228</td>
<td></td>
<td></td>
<td>IEEE 802.15.4</td>
</tr>
<tr>
<td>2248</td>
<td>527.437268</td>
<td>0xee64</td>
<td>0x0000</td>
<td>ZigBee</td>
</tr>
<tr>
<td>2249</td>
<td>527.539784</td>
<td>0xee64</td>
<td>0x0000</td>
<td>ZigBee</td>
</tr>
<tr>
<td>2250</td>
<td>527.543641</td>
<td>0xee64</td>
<td>0x0000</td>
<td>ZigBee</td>
</tr>
<tr>
<td>2251</td>
<td>527.620555</td>
<td>0xee64</td>
<td>0x0000</td>
<td>ZigBee</td>
</tr>
</tbody>
</table>

- Frame 11: 55 bytes on wire (440 bits), 55 bytes captured (440 bits)
- IEEE 802.15.4 Data, Dst: 0x0000, Src: 0xee64

- Frame Control Field: 0x8861, Frame Type: Data, Acknowledge Request, PAN ID Compression, Destination Addressing Mode:
  - Sequence Number: 187
  - Destination PAN: 0x47d0
  - Destination: 0x0000
  - Source: 0xee64

  [Extended Source: SecureMe_00:00:28:45:44 (b8:79:7e:00:00:28:45:44)]

[Origin: 1]

FCS: 0x4e60 (Correct)
ZigBee Frame Injection

- **Data**

```python
1 frame = Dot15d4FCS()
2   /Dot15d4Data(dest_panid=0x4242, dest_addr=0x2323)
3   /"Hello World!"
4
5 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
6 sock.sendto(bytes(frame), ("127.0.0.1", 52001))
```

- **Fuzzing**

```python
1 frame = fuzz(Dot15d4FCS())
2   /Dot15d4Data(dest_panid=0x4242, dest_addr=0x2323)
3
4 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
5 sock.sendto(bytes(frame), ("127.0.0.1", 52001))
```
GNU Radio Meets Scapy

Bastian Bloessl

mail@bastibl.net
www.bastibl.net

February 2019 ▪ FOSDEM SDR Dev Room ▪ Brussels, Belgium