Demo: Decoding IEEE 802.11a/g/p OFDM in Software using GNU Radio

Bastian Bloessl*, Michele Segata*†, Christoph Sommer* and Falko Dressler*

*Institute of Computer Science, University of Innsbruck, Austria
†Dept. of Information Engineering and Computer Science, University of Trento, Italy

{bloessl,segata,sommer,dressler}@ccs-labs.org

ABSTRACT

We just released an Open Source receiver that is able to decode IEEE 802.11a/g/p Orthogonal Frequency Division Multiplexing (OFDM) frames in software. This is the first Software Defined Radio (SDR) based OFDM receiver supporting channel bandwidths up to 20 MHz that is not relying on additional FPGA code. Our receiver comprises all layers from the physical up to decoding the MAC packet and extracting the payload of IEEE 802.11a/g/p frames. In our demonstration, visitors can interact live with the receiver while it is decoding frames that are sent over the air. The impact of moving the antennas and changing the settings are displayed live in time and frequency domain. Furthermore, the decoded frames are fed to Wireshark where the WiFi traffic can be further investigated. It is possible to access and visualize the data in every decoding step from the raw samples, the autocorrelation used for frame detection, the subcarriers before and after equalization, up to the decoded MAC packets. The receiver is completely Open Source and represents one step towards experimental research with SDRs.

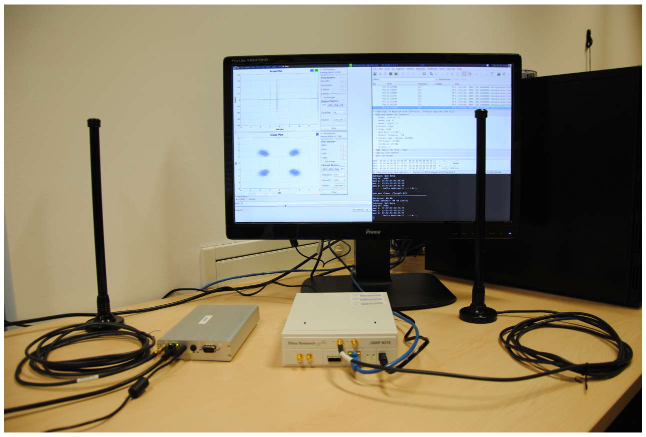


Figure 1: Overview of the demonstration setup. While the GNU Radio receiver is decoding WiFi frames, visitors can interact with the setup by grabbing one of the dipole antennas, move them around, and watch the impact live on the graphical outputs.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; C.4.3 [**Performance of Systems**]: Measurement Techniques

Keywords

IEEE 802.11a/g/p, OFDM, Receiver

1. INTRODUCTION

Given the wide range of applications from IEEE 802.11 a/g/p to WiMAX and to LTE Advanced, Orthogonal Frequency Division Multiplexing (OFDM) gained a lot of attention in the wireless networking community both in industry and in academia. There are many new algorithms that have been proposed for frame detection, frequency offset correction, and channel estimation [2, 4, 7]. Furthermore, OFDM and its applicability in different scenarios and channels has been studied analytically and by means of simulation [4, 5].

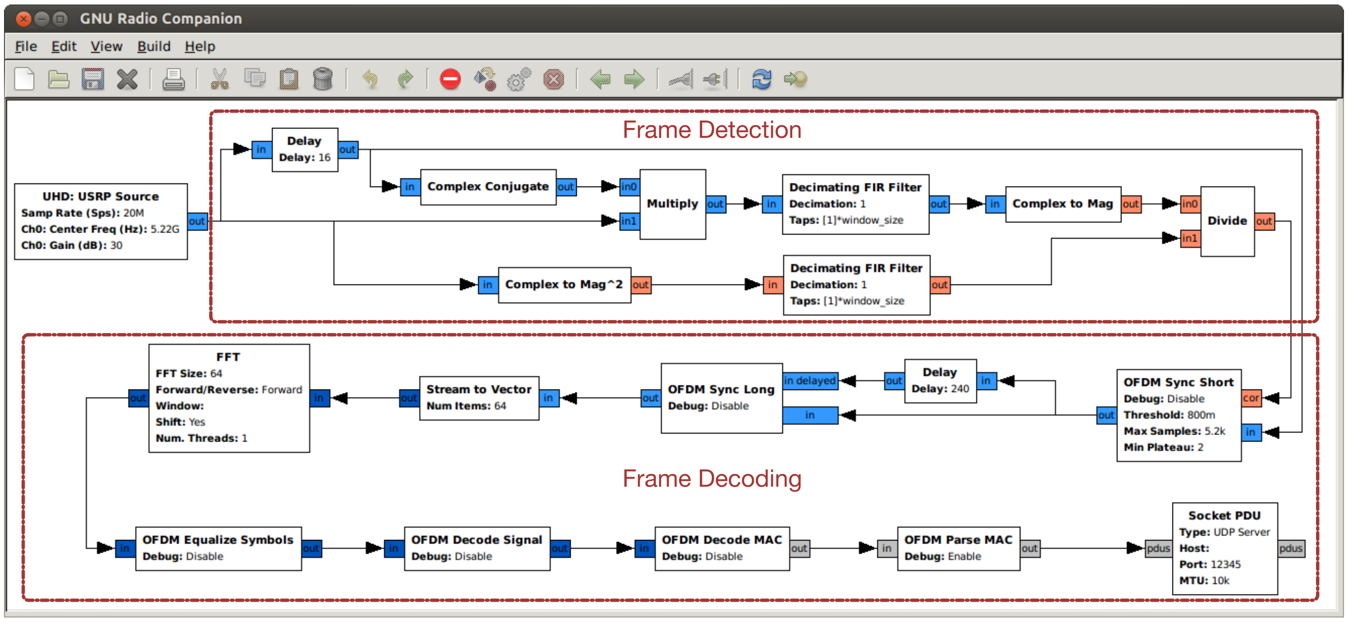
Yet, the possibility to conduct experimental research in this field is extremely limited. In particular, we'd like to mention the custom radio prototypes [9], which are usually based on Field-Programmable Gate Arrays (FPGAs), i.e., rather complex and inflexible.

Even though this approach offers high performance, investigations typically have to focus on small parts of the receive chain, as an implementation of the complete transceiver, together with the design of the hardware platform, incurs substantial effort. Furthermore, the code for custom devices can neither be reused nor verified, nor can results be reproduced by other researchers.

Matt Ettus, the developer of the *USRP* series of devices, supplies an initial GNU Radio OFDM receiver using maximum likelihood estimation [8] and pseudonoise (PN) sequence correlation [6] that did, however, not provide the required bandwidth for WiFi.

To overcome current limitations, we developed a complete OFDM receiver based on GNU Radio and fitted for operation on an *Ettus USRP N210*. This is, to the best of our knowledge, the first prototype of such a Software Defined Ra-

MobiCom'13, September 30–October 4, Miami, FL, USA.
ACM 978-1-4503-1999-7/13/09.
http://dx.doi.org/10.1145/2500423.2505300.



UHD: USRP Source
Samp Rate (Sps): 20M
Ch0: Center Freq (Hz): 5.22G
Ch0: Gain (dB): 30

Delay
Delay: 16

Decimating FIR Filter
Decimation: 1
Taps: [1]*window_size

Decimating FIR Filter
Decimation: 1
Taps: [1]*window_size

FFT
FFT Size: 64
Forward/Reverse: Forward
Window:
Shift: Yes
Num. Threads: 1

Stream to Vector
Num Items: 64

OFDM Sync Long
Debug: Disable

Delay
Delay: 240

OFDM Sync Short
Debug: Disable
Threshold: 800m
Max Samples: 5.2k
Min Plateau: 2

OFDM Equalize Symbols
Debug: Disable

OFDM Decode Signal
Debug: Disable

OFDM Decode MAC
Debug: Disable

OFDM Parse MAC
Debug: Enable

Socket PDU
Type: UDP Server
Host:
Port: 12345
MTU: 10k

Figure 2: Overview of the blocks comprising the OFDM receiver in GNU Radio Companion.

dio (SDR) based OFDM receiver supporting all typical WiFi variants and channel bandwidths up to 20 MHz. Its counter part, the corresponding GNU Radio-based WiFi transmitter, is also already published by Fuxjäger et al. [3].

2. OUR GNU Radio OFDM RECEIVER

As illustrated in Figure 2 the structure of the OFDM receiver is completely exposed to GNU Radio Companion, a graphical tool to setup and configure signal processing flow graphs. The receiver is divided into two functional parts: The first part, depicted in the top half, is responsible for frame detection. The second part, shown in the bottom half, is responsible for decoding the frame. More technical details on the implementation can be found in [1].

We already performed several interoperability tests and verified that the receiver is compatible with off-the-shelf hardware. Table 1 summarizes the results from interoperability tests using different IEEE 802.11a/g/p transmitters conducted in our testbed.

The objective of this demo is to highlight the very straightforward use of the OFDM receiver. In our opinion, the usage is eased by the modularity of the receiver where each of the functional blocks depicted in Figure 2 is responsible for

NIC	Standard	Bandwidth	
MacBook Pro	802.11a/g	20 MHz	✓
Intel Ultimate-N 6300	802.11a/g	20 MHz	✓
Air Live X.USB	802.11a/g	20 MHz	✓
Cohda MK2	802.11p	10 MHz	✓
Unex DCMA-86P2	802.11a/p	10/20 MHz	✓

Table 1: Selection of WiFi and IEEE 802.11p devices we verified to be interoperable with the receiver.

a very specific task. Furthermore, this modularity also offers the possibility to compare different receive algorithm by reimplementing one block with an alternate algorithm and compare the alternative implementations directly by means of simulations with captured input traces or with over the air measurements. Finally, given the possibility to reconfigure the receiver live while it is running, one can also get a feeling for how different parameters impact the performance of the receiver.

3. DEMONSTRATION SETUP

The setup of the demo is depicted in Figure 1. We can see an *Ettus USRP N210* connected to a desktop PC. The second device, which acts as sender, uses an off-the-shelf IEEE 802.11a/g/p Unex DCMA-86P2 WiFi card.

In order to demonstrate the capabilities of the OFDM receiver, we extended it with various graphical outputs that visualize every step of the decoding process. More precisely, we plot the raw complex base band signal as acquired from the SDR in time domain on the top right position in Figure 3. This plot shows the raw input data the receiver is on which the receiver is operating on.

Furthermore, we can visualize the autocorrelation of the incoming sample stream. The autocorrelation of the input samples is relevant for the receiver since it is used to recognize the repeating pattern of the short preamble, which is used by the frame detection algorithm to trigger subsequent decoding operations. This visual output is particularly helpful to find a robust configuration that offers a good trade-off between missing frames and triggering the decoding process just by chance. Even though papers propose certain numeric thresholds for these algorithms we found this indeed helpful, since according to our experience with real hardware and its imperfections these thresholds do not always work too well.

Moving into the frequency domain, we show a constellation plot of the subcarrier symbols after compensation of channel induced phase rotations. A typical screenshot of such a constellation plot is depicted in Figure 3. The QPSK-modulated symbols and the deviation from the ideal positions can clearly be seen. This plot supports an intuitive estimation of receiver performance, channel quality, and bit error rates.

While the constellation plot shows that we manage to equalize the subcarriers reasonably well, we still did not show that we manage to decode the actual payload. For that reason, we output all frames in the PCAP format, which is the de-facto standard for packet capturing and is understood by all network monitoring software. Following Linux, we prepend each frame with a Radiotap header[1], which allows us to annotate frames with further metadata like encoding, bandwidth, channel, and so on. With the help of a Linux pipe, we connect Wireshark directly to the receiver and are able to monitor the traffic live.

In addition to that, we implemented the required blocks to connect the receiver with the Linux network stack. This is done by removing the WiFi MAC header and replacing it with an Ethernet header and feeding the decoded packet into a TUN/TAP interface.[2] Note that one could also capture packets from that virtual interface, but without a Radiotap header we would lose the possibility to annotate frames with metadata.

On the sending side we use a MiniPC built using standard PC components, which is additionally equipped with a Unex DCMA-86P2 WiFi card. This card is based on an Atheros chipset and with minor modifications of the Linux kernel (i.e., removing regulatory domain restrictions), we can use this card to send IEEE 802.11a/g/p frames. More precisely, we can set the channel bandwidth to 10 MHz and 20 MHz and also utilize frequencies at around 5.9 GHz that are dedicated to Intelligent Transportation System (ITS). By setting up static routing and ARP entries on the MiniPC, we can establish a unidirectional communication between the WiFi card and the SDR.

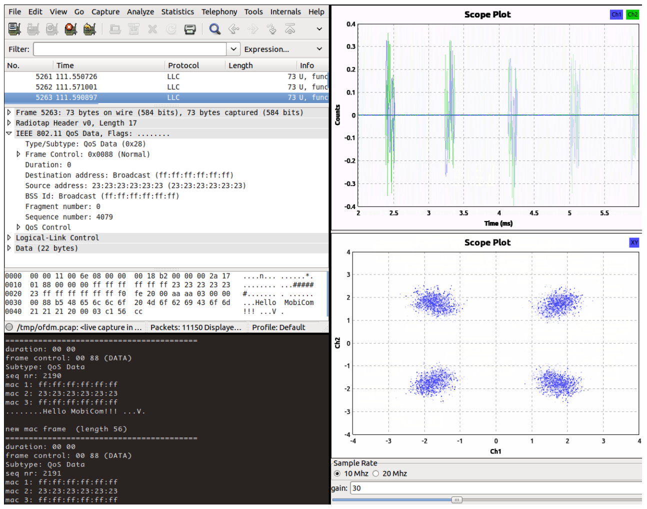
In order to make the demo more interactive for visitors, we connect two large dipole antennas with a pretty long cable to both devices. This way, visitors can grab the antennas and move them around, while watching the impact on the receiver. The constellation plot of the symbols on the different carriers shows very clearly the impact of different antenna placements.

4. REQUIREMENTS FOR THE DEMO

Our equipment consists of a MiniPC from which we send the WiFi frames and a desktop PC where the OFDM receiver is running on. In order to provide the visitors a clear presentation of the live graphs, we connect a screen or a small projector to the desktop PC. In total, we need 4 power plugs (2 PCs, SDR and screen). We do not require much space, a normal table is large enough for the setup. It would be nice if we could pin a poster beside the booth, as this aids to give more detailed explanations of the receiver structure.

Accompanying this demo paper, we made a video that shows the demonstration setup and in particular how visitors can interact. The video, a list of related publications, and all software can be found on our website.[3]

[1]`http://www.radiotap.org`
[2]`https://www.kernel.org/doc/Documentation/networking/tuntap.txt`
[3]`http://www.ccs-labs.org/software/gr-ieee802-11/`



```
No.     Time          Protocol   Length   Info
   5261 111.550726    LLC                   73 U, func
   5262 111.571001    LLC                   73 U, func
   5263 111.590897    LLC                   73 U, func
▷ Frame 5263: 73 bytes on wire (584 bits), 73 bytes captured (584 bits)
▷ Radiotap Header v0, Length 17
▽ IEEE 802.11 QoS Data, Flags: ........
    Type/Subtype: QoS Data (0x28)
  ▷ Frame Control: 0x0088 (Normal)
    Duration: 0
    Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
    Source address: 23:23:23:23:23:23 (23:23:23:23:23:23)
    BSS Id: Broadcast (ff:ff:ff:ff:ff:ff)
    Fragment number: 0
    Sequence number: 4079
  ▷ QoS Control
▷ Logical-Link Control
▷ Data (22 bytes)

0000  00 00 11 00 6e 08 00 00  00 18 b2 00 00 00 2a 17   ....n... ......*.
0010  01 88 00 00 00 ff ff ff  ff ff ff 23 23 23 23 23   ........ ...#####
0020  23 ff ff ff ff ff ff f0  fe 20 00 aa aa 03 00 00   #....... . ......
0030  00 88 b5 48 65 6c 6c 6f  20 4d 6f 62 69 43 6f 6d   ...Hello  MobiCom
0040  21 21 21 20 00 03 c1 56  cc                        !!! ...V .
/tmp/ofdm.pcap: <live capture in ...   Packets: 11150 Displaye...   Profile: Default
```

```
==========================================
duration: 00 00
frame control: 00 88 (DATA)
Subtype: QoS Data
seq nr: 2190
mac 1: ff:ff:ff:ff:ff:ff
mac 2: 23:23:23:23:23:23
mac 3: ff:ff:ff:ff:ff:ff
........Hello MobiCom!!! ...V.

new mac frame  (length 56)
==========================================
duration: 00 00
frame control: 00 88 (DATA)
Subtype: QoS Data
seq nr: 2191
mac 1: ff:ff:ff:ff:ff:ff
mac 2: 23:23:23:23:23:23
mac 3: ff:ff:ff:ff:ff:ff
```

Figure 3: Screenshot of the live visualizations while the receiver is running: Packets in Wireshark (left), time domain signal (right), and constellation plot of (here) QPSK-modulated symbols (bottom right).

5. REFERENCES

[1] B. Bloessl, M. Segata, C. Sommer, and F. Dressler. An IEEE 802.11a/g/p OFDM Receiver for GNU Radio. In *ACM SIGCOMM 2013, 2nd ACM SIGCOMM Workshop of Software Radio Implementation Forum (SRIF 2013)*, Hong Kong, China, August 2013. ACM.

[2] L. Chia-Horng. On the design of OFDM signal detection algorithms for hardware implementation. In *IEEE GLOBECOM 2003*, pages 596–599, San Francisco, CA, December 2003. IEEE.

[3] P. Fuxjäger, A. Costantini, D. Valerio, P. Castiglione, G. Zacheo, T. Zemen, and F. Ricciato. IEEE 802.11p Transmission Using GNURadio. In *6th Karlsruhe Workshop on Software Radios (WSR)*, pages 1–4, Karlsruhe, Germany, March 2010.

[4] T. Hrycak, S. Das, G. Matz, and H. G. Feichtinger. Practical Estimation of Rapidly Varying Channels for OFDM Systems. *IEEE Transactions on Communications*, 59(11):3040–3048, November 2011.

[5] M. Morelli and U. Mengali. A Comparison of Pilot-Aided Channel Estimation Methods for OFDM Systems. *IEEE Transactions on Signal Processing*, 49(12):3065–3073, December 2001.

[6] T. Schmidl and D. Cox. Robust frequency and timing synchronization for OFDM. *IEEE Transactions on Communications*, 45(12):1613–1621, 1997.

[7] E. Sourour, H. El-Ghoroury, and D. McNeill. Frequency Offset Estimation and Correction in the IEEE 802.11a WLAN. In *IEEE VTC2004-Fall*, pages 4923–4927, Los Angeles, CA, September 2004. IEEE.

[8] J.-J. van de Beek, M. Sandell, and P. O. Borjesson. ML estimation of time and frequency offset in OFDM systems. *IEEE Transactions on Signal Processing*, 45(7):1800–1805, 1997.

[9] A. van Zelst and T. C. W. Schenk. Implementation of a MIMO OFDM-based wireless LAN system. *IEEE Transactions on Signal Processing*, 52(2):483–494, Feburary 2004.