Software Defined Radio for Wi-Fi Jamming

Jorge Alberto Duarte García
Computer Engineering – International Exchange Student
Villanova University – Universidad El Bosque
Villanova, PA, USA – Bogotá, Colombia
jduarte@villanova.edu - jduarteg@unbosque.edu.co

***Abstract*— The present article revolves around using commercially available Software Defined Radio (SDR) equipment to interfere, or jam, WiFi networks. The USRP B210 SDR was used for this purpose. The main jamming methodology explored by this research was the broadcast of unmodulated Gaussian noise at various output power levels and at different sample rates, in order to characterize the adequate parameters of an effective jamming signal within controlled test environments. The article starts with a brief introduction, defines SDR and then presents the hardware and software tools that were used. A condensed run-down of the installation procedure to set up the software environment is also presented. Subsequently, an overview of WiFi and WiFi Jamming is shown and then, the main objectives and performance measurements are presented. Afterward, the methodology used is explained, after it the results are shown, and finally conclusions are drawn based on the observed data.**

***Index Terms*— Software defined radio, SDR, Wi-Fi, jamming, interference, gnuradio, gnuradio companion, wireless, network security, USRP, USRP B210.**

I. Introduction

In recent years, the costs of Software Defined Radio (SDR) equipment has decreased drastically allowing a handful of companies to manufacture and sell this kind of equipment to anyone for an affordable price. SDRs must be considered as useful tools in the wireless network security and communications field as they open a whole new world of implementation possibilities (*i.e.* Software Defined Networking) and provide a flexible platform for researchers. Nonetheless, due to the recent facilitated access to this kind of equipment, SDRs could be also considered a possible threat as they can be used for various kinds of attacks. This article will deal with Jamming attacks in particular, targeted towards disrupting the physical layer in a wireless communication scenario over a controlled wireless local area network.

II. Software Defined Radio & GNU Radio

A SDR is a communications equipment which is capable of receiving and/or transmitting radio signals, in which some or all of the physical layer functions are s*oftware defined* [1]. Specifically, previously predominant analog signal processing components (hardwired circuits) are replaced by reprogrammable (algorithmic or logic) digital signal processing components. This way, SDRs can change their performing capabilities (modulation, operating frequency, sample rate, *etc.*) on-the-fly [2], making them very useful and powerful tools for individuals or researchers interested in gaining access to the RF spectrum. Also, due to their digital operational nature, they can also be used to perform advanced digital signal processing and signal analysis.

A. Ettus Research USRP B210 & UHD

Ettus Research, a *National Instruments* (NI) company, has been selling the Universal Serial Radio Peripheral (USRP) line of products (along with NI) for some years. Their products aim to provide an affordable, flexible and reliable hardware platform for Software Defined Radio prototyping and research. The USRP B210 is a bus-powered (over USB 3.0 and 2.0) SDR solution with full-duplex capabilities, MIMO (2 Tx and 2 Rx channels) operation with up to 56 MHz of real-time bandwidth and continuous RF coverage from 70 MHz to 6 GHz [3]. For its versatility, capabilities and for its portability, the B210 was chosen for this research.

Picture 1. USRP B210 front and back connection ports

Ettus Research also provides the USRP Hardware Driver (UHD) which allows the Computer OS (Windows, Linux and Mac OS X) to communicate with the B210, make use of its functionalities as well as send and receive signals from the SDR. UHD also provides an Application Programming Interface (API) which allows developers and applications (such as LabVIEW, Simulink, OpenBTS and GNU Radio) to interface with the USRP hardware [4].

B. GNU Radio & GNU Radio Companion

GNU Radio is an Open-Source framework for designing, simulating and deploying real-world radio systems [5]. Along with the GNU Radio Companion (GRC) application, GNU Radio provides a modular *flowgraph*-oriented approach for modeling and developing radio communication protocols, digital signal processing and interfacing with radio equipment.

GRC's main Graphical User Interface (GUI) allows the user to add and chain *blocks* together to create *flowgraphs* that are translated into python code (by GRC). This code makes use of several python libraries, including *gnuradio* and *uhd*, and then is executed by the host computer. The following block diagram illustrates how UHD, Python, GNU Radio and GRC interact with each other, the host OS and the USRP hardware:

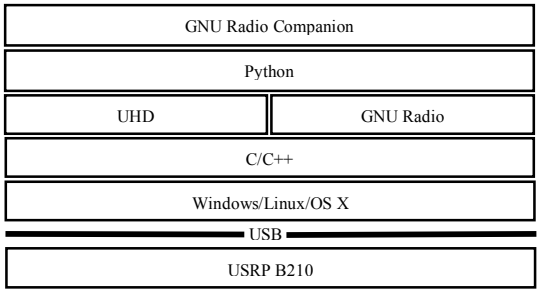


Figure 1. GRC software architecture

Even though GRC is very extensible, one of the advantages of having flowgraphs directly translated into python is that after generating the code, it can still be edited to add additional calculations or functionalities. For this reason, GRC was chosen to be the main interface for interacting with the USPR B210. This allowed to quickly prototype and execute experiments, and when required, additional routines could be programmed and executed for running simulations.

III. WiFi & WiFi Jamming

WiFi networks implementing the IEEE 802.11b/g/n standard have 14 channels in the 2.4 GHz band, a general channel spacing of 5 MHz (12 MHz between ch13 and ch14), a 22 MHz bandwidth per channel. This means non-overlapping channels are channels 1, 6, and 11, which tend to be the most used channels. Also, WiFi communications implement Orthogonal Frequency Division Multiplexing (OFDM) for its physical layer (PHY), and signals are modulated using Quadrature Phase Shift Keying (QPSK) for its Wireless LAN (WLAN) Medium Access Control (MAC) [6].

Jamming can be defined as the act of making a broadcast, or any other electronic signal, unintelligible by causing interference [7]. In principle, it can be considered as a physical attack on communications, because it targets the medium through which the communication occurs. Particularly for this research, Jamming will be explored as an attack on the wireless transmission of radiofrequency over the air (medium), between two parties.

Having highlighted the technicalities of the 802.11b/g/n standard, the following list summarizes the capabilities that are needed from the SDR in order to jam WiFi communications:

a) transmit a signal in the 2.4 GHz band
b) transmit a signal with a bandwidth similar to the one being used to communicate by the two parties
c) modulate the transmitted jamming signal using QPSK
d) transmit a signal at least as strong as the two parties communicating
e) transmit a signal at a sample rate similar to the one at which the two parties are communicating (20 MHZ [6]).

By contrasting them with the specifications of the USRP B210, it becomes evident that the SDR meets with all the requirements mentioned above. Therefore, by emitting signals that comply with the standard's requirements the B210 should in theory be able to interfere with WiFi communications.

IV. Objectives & Performance Measurement

Having defined WiFi Jamming in the previous section, and having also discussed the viability of the experiment, the main objective for this research is to effectively interrupt communications between a Wi-Fi client and a wireless Access Point (AP) by introducing a Jamming signal in their operating environment.

As an effective performance measure for the experiments executed during this project, the time difference between a PING from the client and a response from the AP (delay) will be used. By doing so, the relationship between the parameters for each experiment (transmission sample rate and signal output power) and the effectiveness of the Jamming signal can be measured in a consistent and repeatable way.

V. Methodology

In this section, the install instructions for the necessary software tools and drivers will be outlined, including the gr-foo and gr-ieee802_11 libraries. Even though these last two where not used in the execution of the experiments presented in this research, they are relevant for the study and implementation of the IEEE 802.11 protocol using GNU Radio. Please note that Mac OS X 10.11.4 (El Capitan) was used for this research. Additionally, the two experimental scenarios used to perform this study will be presented and explained.

A. Software and Environment Setup

The installation process of the required drivers, software and libraries for this research can be summarized in the following steps:

1. Install MacPorts, a third-party package manager for Mac OS [8].
2. Install UHD and its dependencies via MacPorts [9].
3. Install GNU Radio via MacPorts [10].
4. Download gr-foo [11] and gr-ieee802-11 [12].
5. Install both libraries from source following the provided instructions [11, 12], but using MacPorts' python environment variables as arguments for running CMAKE as suggested in the *GNU Radio Mac installation guide* [10] (under the section *Compiling GNU Radio from Source)*:

```
"CC=/usr/bin/llvm-gcc CXX=/usr/bin/llvm-g++ cmake
-DCMAKE_INSTALL_PREFIX = /opt/local/ -DPYTHON_EXECUTABLE =
/opt/local/bin/python2.7
-DPYTHON_INCLUDE_DIR =
/opt/local/Library/Frameworks/Python.framework/Versions/2.7/Headers
-DPYTHON_LIBRARY =
/opt/local/Library/Frameworks/Python.framework/Versions/2.7/Python
.."
```

Code 1. Example CMAKE code

6. Edit bash profile to set MacPorts' Python library as the PYTHONPATH environment variable on every login (`export PYTHONPATH = /opt/local/lib/python2.7/site-packages`).

By following these steps correctly an adequate environment is set up, in which GRC, gr-foo and gr-ieee802-11 make use of MacPorts' default python 2.7 installation. Also GNU Radio and UHD are able to work together to communicate with the USRP Hardware via USB. This can be tested by running the commands `uhd_find_devices` (test if the SDR is connected and recognized) and `uhd_usrp_probe` (test if the SDR is accessible and operational) provided by the installed USRP Hardware Driver.

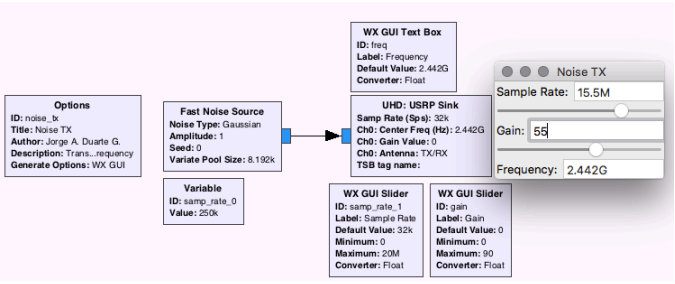


Figure 2. GRC Blocks and flowgraph execution

B. Experimental Setup

The IEEE 802.11 standard defines WiFi implementations that are able to perform *Channel Hopping* to avoid interference (*i.e.*802.11g). However, in order to keep this experiment simple, communications between the client and the access point were held using a single WiFi channel. Channel 7 was chosen for this experiment, which means that the exact frequency that was transmitted by the B210 was 2,442 MHz (2.442 GHz).

GRC's block that models the B210's transmission (*UHD: USRP Sink*) allows the user to define the sample rate at which a signal is transmitted (in Samples/Second, implemented as Hz). It also allows to modify the gain applied to the transmitting channel (amplification). This gain factor can be a value from 0 to 90. However, when at its maximum, the B210's power output is 10dBm (roughly 10mW). *Figure 2* shows the GRC flowgraph that was implemented and executed for the experimental scenarios presented in the next section.

As stated before, the procedure used to test the effectiveness of the Jamming signal was the delay time in Pings from the client machine to the WiFi AP. To make the testing process more automatized, after compiling the GRC flowgraph, the generated python script was modified to perform the ping requests to the AP in an automated fashion, as well as changing the SDR's sample rate and gain parameters sequentially (decreasingly, every 5 pings). *Figure 3* illustrates the layout and main idea behind the experimental setup for this research.

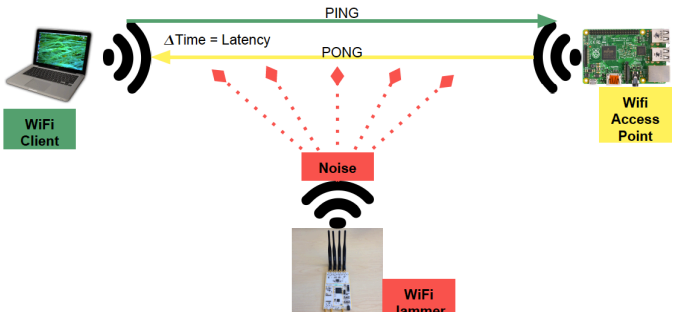


Figure 3. Experimental Setup Diagram

C. Experimental Scenario 1

For this first experimental scenario, all the components (Client, AP and Jammer) where placed close together over the same table, being separated only by a few of inches. This was done to determine the minimum amount of gain (power amplification) needed to interfere with the communication between the two parties (AP and client). The SDR's gain was tested from 90 dBm to 0 dBm of amplification, and sample rates ranging from 25 MS/S to 16 MS/S where tested for each gain level as well. The gain factor was diminished by 1 dBm every time all sample rates (in 1 MS/S steps) were tested for that specific gain factor by attempting 5 pings to the AP, whose delay times were logged into a text file.

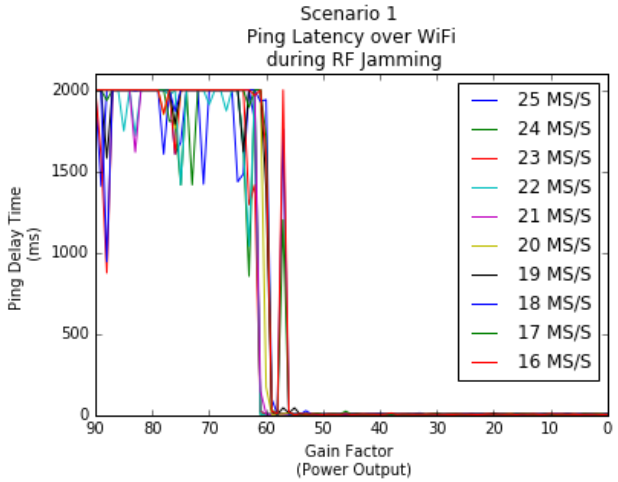
D. Experimental Scenario 2

This second experimental scenario was very similar to the first one, except for two details. First of all, the client was separated from the AP by 15 feet, the SDR was separated from the client by 4 feet and from the AP by 7 feet. Secondly, measurements were taken in the same way as in the previous scenario, however the gain was only tested from 90 dBm to 51 dBm.

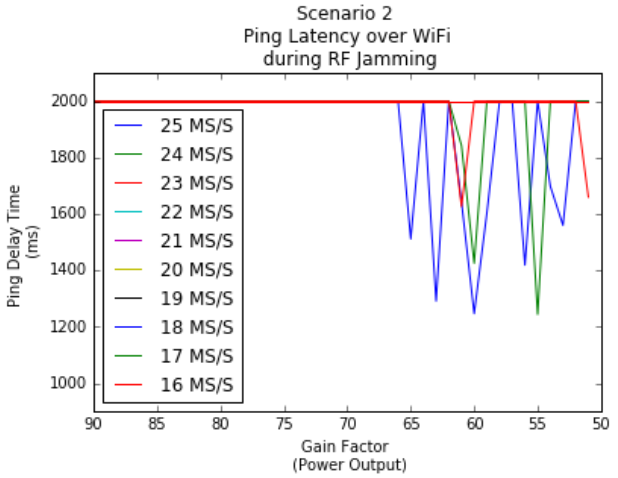
The results for both scenarios are presented and discussed in the following section.

VI. Results

Having executed both experimental scenarios and recorded their corresponding data, graphs where plotted in order to visualize and compare the effects of the jamming signal on the communication between the client and the AP. These are presented in *Graph 1* and *Graph 2*.



Graph 1. Scenario 1 data



Graph 2. Scenario 2 data

From the first scenario, the first thing to note is how the jamming signal has almost no effect on the packet latency when the gain factor is below 50 dBm, regardless of the sample rate at which the signal was transmitted and taking into account how close the components where to each other. This is why on the second scenario the gain factor was tested only from 90 dBm to 51 dBm.

When contrasting the graphs generated by the data, it is evident that on the first scenario there is a lot more variation in the impact sample rates seemed to have on the communications between the two parties. However, when the elements were more distanced (on the second scenario), only a handful of sample rates seemed to exhibit that kind of behavior, and only before a definitive threshold where the gain of the jamming signal seems to prevent all further communications between the client and the AP.

Ultimately for both scenarios, it can be observed in both graphs that at some point the latency of the ping packets is so high, the communication is totally and effectively interrupted between the two parties. In other words, their communication was *jammed*.

VII. Conclusions

From the data collected in the experimental scenarios and presented in the previous section, it can be concluded that SDR is a viable platform to perform WiFi jamming. It was also possible to observe that the amplification factor for the USRP B210's jamming signal had to be above 51 dBm in order to start interfering with the communications. Nonetheless, effective communications jamming only occurred above 75 dBm for most of the sample rates implemented in both experimental scenarios.

Regarding sample rates, it was observed that those over the 22 MS/S rate seem to perform consistently in both scenarios, and this is most likely because the specified sample rate for WiFi communications is 20MHz. By "firing" noise at a faster rate than that at which communication is occurring, there is a higher chance of interfering the receiver's (client and AP) ability to capture valid packets.

VIII. Future Research

Even though the main objective of this research was met, further testing by implementing more realistic scenarios will be required to really explore and take advantage of all the capabilities offered by SDR equipment such as the B210.

Some of the future suggested work topics include practical research focused on:

- Effectively interrupting communications between a Wi-Fi client and a wireless router by introducing a Jamming signal in their currently active Wi-Fi channel, and being able to reactively change channels if the communications is reestablished on another one.
- Effectively interrupting communications between a Wi-Fi client and a wireless router by introducing Jamming signals in different channels simultaneously and randomly.
- Effectively interrupting communications between a Wi-Fi client and a wireless router by systematically capturing the client's packets (Man in the middle attack) and forwarding them in an invalid format to the router (network poisoning) and *vice versa*.

Acknowledgment

The author would like to thank D. Chasaki, PhD. for providing the necessary equipment that made this research possible. Also, many thanks to C. Mansour, Ms.CE. for his help while troubleshooting the SDR environment installation.

REFERENCES

[1] SDR Forum, SDRF Cognitive Radio Definitions. 2007, Available: http://www.sdrforum.org/pages/documentLibrary/documents/SDRF-06-R-0011-V1_0_0.pdf.

[2] F. Van Hooft. A heterogeneous software defined radio architecture for electronic signal interception, identification and jamming. 2003, Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1290364. DOI: 10.1109/MILCOM.2003.1290364.

[3] Ettus Research, USRP B210 product page. 2016, Available: https://www.ettus.com/product/details/UB210-KIT

[4] Ettus Research, UHD Github Repository. 2016, Available: https://github.com/EttusResearch/uhd

[5] GNU Radio Website, Introduction to GNU Radio and Software Radio. 2013, Available: http://gnuradio.org/redmine/projects/gnuradio/wiki/Guided_Tutorial_Introduction

[6] IEEE Standards Association, IEEE 802.11 Standard. 2012, Available: http://standards.ieee.org/getieee802/download/802.11-2012.pdf

[7] Merriam-Webster Dictionary, Definition of Jamming. 2013, Available: http://www.merriam-webster.com/dictionary/jam

[8] MacPorts.org, Download & Installation. 2016, Available: https://www.macports.org/install.php

[9] Ettus Research, USRP Hardware Driver and USRP Manual. 2016, Available:http://files.ettus.com/manual/page_build_guide.html#build_instructions_osx

[10] GNU Radio, GNU Radio Mac Installation Guide. 2016, Available:http://gnuradio.org/redmine/projects/gnuradio/wiki/MacInstall

[11] Bastian Bloessl's GitHub, gr-foo repository. 2016, Available: https://github.com/bastibl/gr-foo

[12] Bastian Bloessl's GitHub, gr-ieee802-11 repository. 2016, Available: https://github.com/bastibl/gr-ieee802-11