Electrical & Computer ENGINEERING

18-441/741: Computer Networks

Project 1: Exploring the Wi-Fi PHY

Questions? Email TA

1. The Wi-Fi Physical Layer

Have you ever wondered what a Wi-Fi signal from your computer or phone actually looks like? The objective of this project is to make you learn in action, how a popular physical layer protocol (Wi-FI) works. In this project, you will build a very simple Wi-Fi decoder in software that mimics the very same processes that your phone/laptop's Wi-Fi chip does. You will be given a reference MATLAB design that simulates Wi-Fi packets: taking the contents of a packet and creating a Wi-Fi signal. Your role is to reverse this process: take as input a Wi-Fi signal and return the corresponding Wi-Fi message.

2. You are given a Wi-Fi transmitter code

You are given a MATLAB .m file which generates a Wi-Fi packet, a function defined as follows:

```
function txsignal = wifitransmitter(message, level, snr)
```

The `wifitransmitter` function takes as input:

(1) `message` : A text 'message' that contains the contents of a Wi-Fi packet. It should be able to handle arbitrary textual input (i.e. any sequence of ASCII characters). The sequence will be no longer than `10000` bytes.

(2) `level` : Level indicates various stages of encoding. Each message undergoes five levels of encoding before becoming a wireless signal: (1) Level-1/Interleaving: The bits are permuted by a well-known permutation; (2) Level-2/Coding: A turbo code is applied to the bits; (3) Level-3/Modulation: The bits are mapped to complex numbers by a simple mapping (4-QAM); (4) Level-4/OFDM: The bits are converted to an OFDM sequence; (5) Level-5/Noise: Some noise is added to the packet and we apply some zero-padding to the beginning and end of the packet. Default level is 5.

(3) `snr` : Specifies the signal to noise ration (in dB) applied to the signal. Value could be any positive or negative real number. Default snr is infinity.

Once the function is called with appropriate inputs, the value `txsignal` equals the sequence of complex numbers containing the encoded packet.

You are encouraged to run this program in MATLAB for different inputs and observe the output values. You are also encouraged to thoroughly read the .m file to better understand the different levels.

Note: The implementation you are given is a very simplistic version of the Wi-Fi physical layer that excludes many bells and whistles for simplicity (e.g. cyclic prefix, pilots, guard bands, etc.). The goal is to give you a sense of how both a Wi-FI transmitter and receiver work at the PHY layer. However, since we are not following the Wi-Fi protocol to the tee, you will be unable to solve this project with standard Wi-Fi decoding libraries or tools. You are therefore strongly encouraged to write your own receiver code from scratch. You are welcome to use libraries for basic signal processing mathematical functions (Fourier transform, Turbo code libraries, etc.).

3. Your Task: Write the Wi-Fi receiver code

Your task is very simple: create a Wi-Fi receiver program called that takes as input the transmitted signal `txsignal` generated by `wifitransmitter` and outputs `message`. We will run your program by running the following function in MATLAB that you need to write:

```
[message, length, start] = wifireceiver(txsignal, level)
```

Your main deliverable is the function `wifireceiver` .m in MATLAB. While you are encouraged to use MATLAB to write this, you are permitted to invoke a system call to C/C++/Java/Python within the MATLAB script, should you choose to do so. However please note that `wifireceiver` .m must handle all required compilation. You should submit your original source code and **not** an executable.

We will run the following tests as explained below. For illustrative purposes, the following examples use '`hello world`' as the transmitted message. Your program should work correctly for any valid input message to `wifitransmitter`.

(1) Level 1 Test (10 pts)

In the level-1 test, only interleaving is enabled at the transmitter.

On MATLAB:

```
>> txsignal = wifitransmitter('hello world', 1)

>> wifireceiver(txsignal, 1)
```

Should output the following:

```
hello world
```

Your program at this level should undo the interleaving correctly.

(2) Level 2 Test (20 pts)

In the level-2 test, the turbo encoding and interleaving steps are enabled at the transmitter.

On MATLAB:

```
>> txsignal = wifitransmitter('hello world', 2)
```

```
>> wifireceiver(txsignal, 2)
```

Should output the following:

```
hello world
```

Your program at this level should undo the turbo decoding and interleaving correctly.

(3) Level 3 Test (10 pts)

In the level-3 test, the turbo encoding, interleaving and modulation steps are enabled at the transmitter.

On MATLAB:

```
>> txsignal = wifitransmitter('hello world', 3)

>> wifireceiver(txsignal, 3)
```

Should output the following:

```
hello world
```

Your program at this level should undo the turbo decoding, interleaving and modulation correctly.

(4) Level 4 Test (10 pts)

In the level-4 test, the turbo encoding, interleaving, modulation and OFDM signal generation steps are enabled at the transmitter.

On MATLAB:

```
>> txsignal = wifitransmitter('hello world', 4)

>> wifireceiver(txsignal, 4)
```

Should output the following:

```
hello world
```

Your program at this level should undo the turbo decoding, interleaving, modulation and OFDM steps correctly.

(5) Level 5 Test (30 pts)

In the level-5 test, all steps are enabled at the transmitter.

On MATLAB:

```
>> txsignal = wifitransmitter('hello world', 5, 30)

>> [message, length, start] = wifireceiver(txsignal, 5);
```

Should output the following:

```
message = 'hello world'
length = 11
start = <number of padded zeros before the packet>
      (= length(noise_pad_begin) in wifitransmitter.m)
```

Your program at this level should undo the turbo decoding, interleaving, modulation and OFDM steps correctly. You will be awarded 10 points each for each of the three outputs. To get the full 30 points, you are guaranteed that the snr input will be 30 dB or higher. Supporting snr values below 30 dB will be an extra credit problem for 18-441 and a required task for 18-741, worth ten points. **Remember that snr values can be negative.**

5. Discussions

I'm confused, where do I start?

Relax; the project may not be as hard (or as easy) as you think initially. Make sure you start early, there are plenty of good online resources available to get started. CMU has a software license for ECE students to install MATLAB: https://www.cmu.edu/computing/software/all/matlab/. Note that you are not required to use MATLAB to program the bulk of your code, but MATLAB's built-in libraries make it relatively easy to implement the basic signal processing tools you will need. A good starting point is to start with level-1 and then progressively include levels 2, 3, 4 and 5 and build your script. Reach out to the TAs or use piazza should you face any questions.

Help! I accidentally delete my working code.

Make a lot of backups, if you do not have the source when the deadline comes, you do not have the project. You may also want to use version control tools to manage your source. The ECE undergraduate cluster machines are equipped with both `git` and `subversion` tools (although learning to use them is not within the scope of this course). Make sure you have a working branch/revision available for final submission before start tuning for performance or adding features. You could always submit an early working version to the hand-in directory. So make sure you contact the TA and give us information about your submission early.

7. Hand-in Procedure

You will submit your project to canvas and we will test your codes using our own server. Create a file named `README` in the primary submission directory so we know that it is the primary directory that we need to grade (we will ignore the other).

5. Deliverable Items

The deliverables are enumerated as follows,

1. **The source code** for the entire project (if you use external libraries, provide the binaries needed for compilation of your project)

2. **Do not submit the executable files** (we will DELETE them and deduct your logistic points).

3. **A brief design document** regarding your server design in `design.txt / design.pdf` in the submission directory (1-2 pages). Tell us about the following,
    a. Your decoder design
    b. How did you infer the length of the packet?
    c. Libraries used (optional; name, version, homepage-URL; if available)
    d. Extra capabilities you implemented (optional)
    e. Extra instructions on how to execute the code, if needed

6. Grading

Partial credit will be available according to the following grading scheme,

Items	Points (100 – **441** + 110 – **741**)
Logistics - Successful submission & compilation - Design documents	 10 10
Levels - Level-1 test - Level-2 test - Level-3 test - Level-4 test - Level-5 test (10 points per output)	 10 20 10 10 30
Required for 18-741 only (bonus for 18-441) Handle snr < 30 dB (student(s) who achieve(s) lowest snr will get maximum points, others will be graded on a curve) Remember that **SNR values can be negative**.	 10