MAC-PHY Layer Design for Cognitive Radio

Utpal Solanki, utpal@drexel.edu

Abstract:

The final design project includes implementation of MAC, PHY and DSA layer in MATLAB that ultimately follow a file transfer protocol with following constraints. First, transmitter has to perform a wideband spectrum sensing to learn about present Primary Users (PU). Transmitter then classify PUs and follow DSA policy to find spectrum holes. After that, transmitter and receiver go to a predefined frequency band where they perform rendezvous handshake. During this handshake, transmitter tell receiver about which frequencies are available within that wideband sensing. After successful handshake, transmitter and receiver comes to that spectrum and start test file transfer. File transfer happens with simple ARQ protocol. With rest of the report, we discuss unique PHY, MAC and DSA design used to make robust functioning application.

This design is targeted to use MATLAB scripts as software and USRP hardware as SDR radios.

PHY Design:

A very unique part of our design is different PHY layer which is simple, steady and more robust in comparison to other PHY layers designed. We have used FSK modulation technique where symbol 1 is represented as 57 KHz sine wave and symbol 0 is represented as no wave (zeros). At sampling rate of 200K samples per seconds, a symbol length is 21 samples that correspond to 105 us. Theoretically the data rate is 9.52 Kbps, however maximum achieved data rate was 1.17 Kbps. Limiting factor was heavy number of FFT iterations that took significant time under MATLAB computing.

Modulation is straightforward, MAC layer sends packet bits to PHY layer. PHY layer modulate this bit patterns as FSK modulated signals with described configuration. Before sending this packet to further down, PHY layer append a unique preamble to this modulated packet. In the end, PHY layer perform standard CSMA and on success, put this packet in USRP's outgoing buffer. Unique preamble is designed to avoid "Packet in Packet" attack that can easily exploit PHY layer security. Consider if preamble is placed at bit layer, there are also chances that user can intentionally send same bit patterns in data payload. A targeted receiver may sync to this preamble bits in data payload and following bits will be processed as if it is a packet. This way, user can send a full packet as data payload within a packet and can exploit PHY layers. The best and easy solution to avoid such attack is to design preamble after modulation and not in bit layer. Signal waves generated by PHY must not be repeatable by any data bit patterns in payload. We simply used different length of symbols as preamble that can never occur with any data bit patterns in payload.

Our design implements custom demodulator blocks and less depends on communication toolbox of MATLAB. As a result, we have great flexibility in tuning robust PHY layer. During demodulation, when MAC layer request to receive a frame, PHY layer look through abrupt change in energy level within received frame from USRP. Looking at energy change in time domain sample is fast enough to process all frames generated by USRP in real time. We only go for further order of processing if there is any detection in energy change. Also, the abrupt energy change detection algorithm is robust enough such that PHY layer has very less false detection with very low SNR condition.

Once PHY detect possible signal present in frame, it performs frame synchronization and trimmed out samples that correspond to potential packet. Actual demodulation happens now. There is a moving window with same size as symbol length that moves from start to end of frame buffer. This window compute FFT and detect dB level of center frequency (57 KHz). This window move from left to right by 1 sample every window iteration. Figure 1 below shows how symbols look after applying above algorithm. It is clearly visible that even though symbols in time domain are imposed with undesired waves, the demodulation is still robust and there is little reflection about change in magnitude of symbols since demodulation is interpreted at logarithmic scale. This unique demodulation technique makes whole project so stable and robust that it allows some overhead to be skipped at MAC layer.

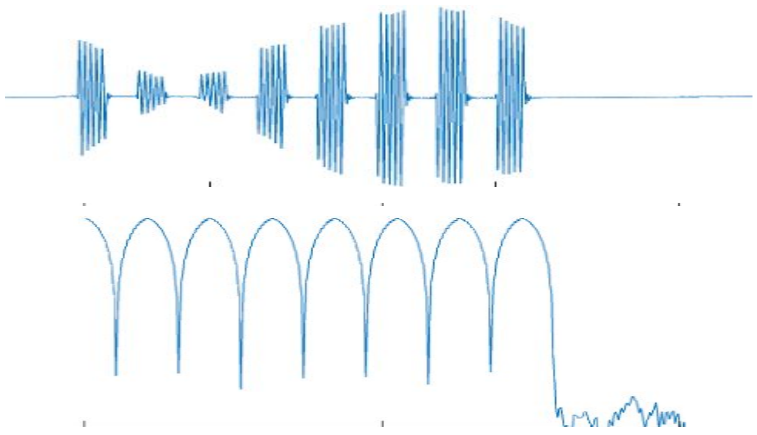

Figure 1. Symbols in time domain and respective fft windowing plot.

Provided that we were receiving USRP frame with 100K samples and we have to perform nearly around 100K FFT iteration, MATLAB's FFT implementation was able to give us such heavy computing within 1 to 2 seconds. This performance penalty can be greatly improved with more optimum demodulation algorithm which should be as much robust as fft windowing is. For example, Matched filter or DFT calculation of center frequency can help demodulate symbols. If given more time, we could have optimize this penalty that completes within few milliseconds.

Once the fft windowing plot is ready, the plot is sampled at fixed symbol size interval and decide bit 0 if dB value is 5dB below maximum value or else bit 1. This demodulation is so robust that during entire testing period, we hardly found any corrupted packet. If packet is correctly detected and synced in frame buffer, it gave always 0% Bit Error Rate.

While we transmit data packet with this modulation technique, it should have either 57 KHz frequency or just zeros. Since we were transmitting real as well as imaginary part, packet took almost 57 - (-57) = 114 KHz of bandwidth in spectrum. We tried removing imaginary part so that we can have a narrow band taken in spectrum, however we could not resolved issue in time where we were seeing abnormal spectrum spread of signal. Therefor, we sticked to sending 117 KHz bandwidth signal. Figure 2 below shows two type of signal in spectrum.

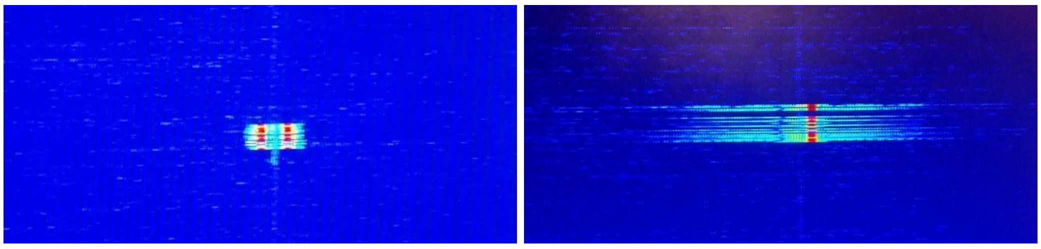

Figure 2. Left, signal with Real and Imaginary frequency. Right, signal with only real frequency.

MAC Layer Design:

We have made very simple MAC layer since PHY layer was working robust enough that allow us to skip overhead at MAC layer. Starting with packet header format, we have 12 bits to indicate data payload length, thus we can send maximum 4K Bytes of payload in one shot. We have 4 bits to indicate packet type, Ex. Data packet, Ack packet, Control packet. Following, 8 bit sequence number, 8 bits flag of which one is End of File flag and other reserved bits and 32 bit CRC32 of data payload after header. In the end of header there is 8 bit CRC8 which is computed on all the above information in header. If we see any corruption in header CRC, we discard packet rightway instead of processing other data bits. If CRC8 is okey, we can trust information within header.

Packet data payload is pure payload in bit stream, there is no data interleaving and data encoding applied on this field. We can easily apply one, but as mentioned previously, our PHY layer is robust enough that take care of everything. We have sufficient error detection using CRC32. This MAC layer packet and PHY layer packet frame is true variable data, variable frame implementation that takes only required samples in time domain and no extra zero paddings.

MAC layer code has simple “send packet” and “receive packet” API set. Average round trip time for DATA send and ACK is 3 seconds of which 2.5 seconds is spent in fft window computing. If given an optimized method, this round trip time can be significantly improved. We used parallel for loop in MATLAB that divide computation load of for loop iteration on available multi core in processor, however we didn’t see significant improvement with this.

DSA Design:

One of the primary design challenge of this project was to implement a software radio with more cognition part in it. Radio needs to be smart enough to gently avoid primary user and silently utilize appropriate spectrum whole as secondary user. With our designed, transmitter first sense 2.5 MHz wide spectrum at center frequency of 2.520 GHz. The sampled data is then processed through See-Far algorithm. We see a banding of spectrum band as shown in figure 3, is corrected using trained data collected when there is not PU present. Spectrum is normalized on horizontal line so that it can be used accurately for appropriate thresholding. We then smooth the fft plot with moving average filter and then do downsampling. After that, we use “envelope” feature with MATLAB to detect accurate PU with minimum width and prominence threshold. Figure 4 shows detected primary users.

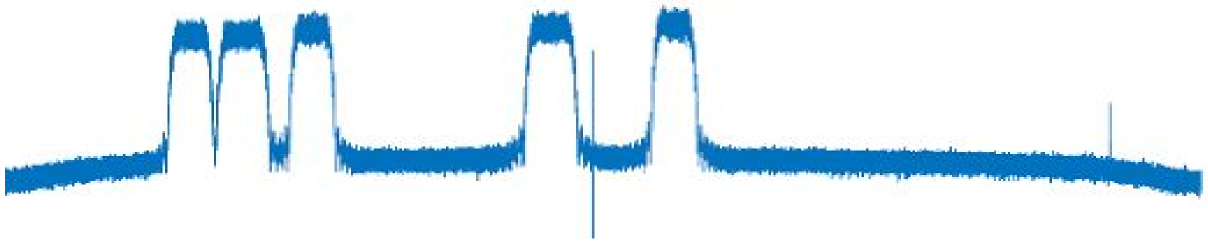

Figure 3. Spectrum banding effect due to hardware limitation

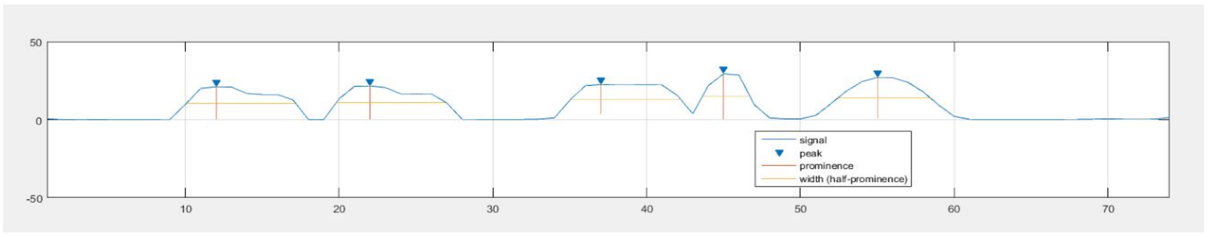

Figure 4. Detection of primary users

To detect best spectrum hole, we look through the curve in figure 3 and find where that curve stay at 0 level continuously. Best algorithm to compute this is, have a window that calculate sum of element in that window, slide that window on above curve from left to right that results in a curve of figure 5. Peaks on this curve represent best spectrum whole. We remember all free spectrum slots and make use of it if file transfer is not working smoothly in a particular slot.

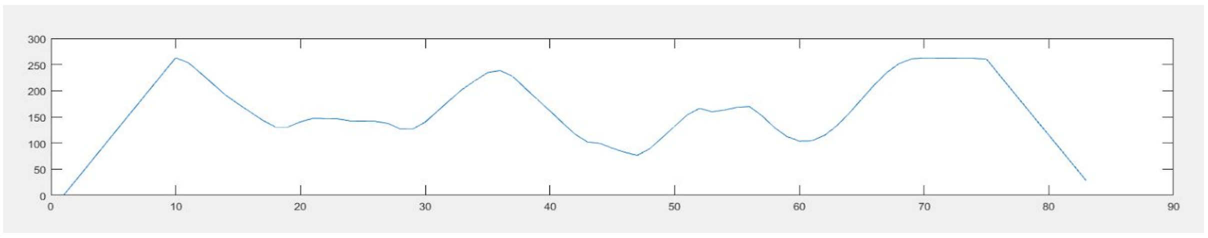

Figure 5. Peaks on curve represent best spectrum holes.

Application Layer Design:

Application layer first invoke DSA sensing routine that list all PUs and possible classification. Transmitter initiate rendezvous handshake with receiver. This handshake is close loop. Even though transmitter and receiver ends up at different frequency, they will still timeout and come back to handshake band. It takes a while to learn timeout but it just works and transmitter-receiver never ends up in deadlock.

Out of many spectrum hole, transmitter start sending file through best spectrum hole. If transmitter learn that file sending is not working out smoothly, both transmitter and receiver will go back to handshake band and start all over again on next best available spectrum hole. Once file transfer works for the first time, transmitter will send file two more time through same frequency.

Modular Code:

Our software design is modular from top to down. Application, DSA, MAC and PHY layer talk with each other through generic API. We were easily able to plug in-out new layers as needed. Infact, we dramatically changed PHY layer in significantly less time.

Whole project is delivered in single source directory. Source can work as transmitter or receiver by just modifying state flag in configuration script. Transmitter and receiver make reuse of code in MAC and PHY API. For example, transmitter use "MAC_sendPacket" function call to send data and control packet, where receiver use same function to send acknowledgement of data packets.

Conclusion and Results:

During final project run, we were consistently able to transfer file within 30 to 40 seconds. Since project policy was to send packet as line by line, we could have send whole file in just one packet. This one shoot transfer time was 8 seconds. Our motive was to implement a robust design that works well in almost all condition. During testing, it is found that we were able to transmit file at more than 25 feets distance with just 0 dB transmission gain. That proves how robust our design is even though there is very low SNR.

References:

1. "Packets in Packets: Orson Welles' In-Band Signaling Attacks for Modern Radios" Travis Goodspeed, University of Pennsylvania.
2. USRP with MATLAB, http://www.mathworks.com/hardware-support/usrp.html
3. Source code repository for this projet, https://github.com/utpalsolanki/CognitiveRadio