EE312 Midterm Report: Thesis Synthesis

Liu Yuanzhuo,Qiu Kunyuan,Wu Shuhao
Deng Yu, Ji Chenqing, Zhao Qingyu

Department of Electronics and Electric Engineering, SUSTech

2021.4.13

SYNTHESIS

Software-defined radio (SDR) can still be time-consuming to design and implement, as they typically require thorough knowledge of the operating environment and a careful tuning of the program.
Our contribution is the design of a bidirectional transceiver that runs on the commonly used **USRP** platform and implemented in MATLAB using standard tools like **MATLAB Coder** and **MEX** to speed up the processing steps.

System Architecture Overview

The architecture of the whole project can be sectioned as the three stages below:

- 1.**Parameter Initialization**: Recommended parameters are preset in the system.
- 2.**Simulation**: Do simulation to explore the parameter with less than 5% packet loss at receiver.
- 3.**Experiment**: Apply the parameter to USRPs for over-the-air experiments.

And the USRP SDR platform is adopted as the hardware platform of this experiment, as the Figure 1 shows.

Figure 1: End-to-end TX/RX chain using USRP

System Architecture Overview

IEEE **802.11b PHY and MAC** layer packet structure specifications is adopted. USRP frames will compose the 802.11b packet.

DBPSK(differential binary phase shift keying) is adpoted in the research, implying **MCS**=0.

SDR Software Platforms

specialized software needed to work with SDR system for modulation,preamble detection, encoding, and filtering.

Example

- **GNU Radio**:open source, hardware-independent, and modifiable
- **Software Communications Architecture(SCA)**:open-source,HW-independent framework,using data flow diagram
- **OSSIE**:using the SCA framework for interaction with the USRP board

SDR SOFTWARE PLATFORMS

The user interface of a SDR backend software (GNU Radio) is shown as Figure 2.

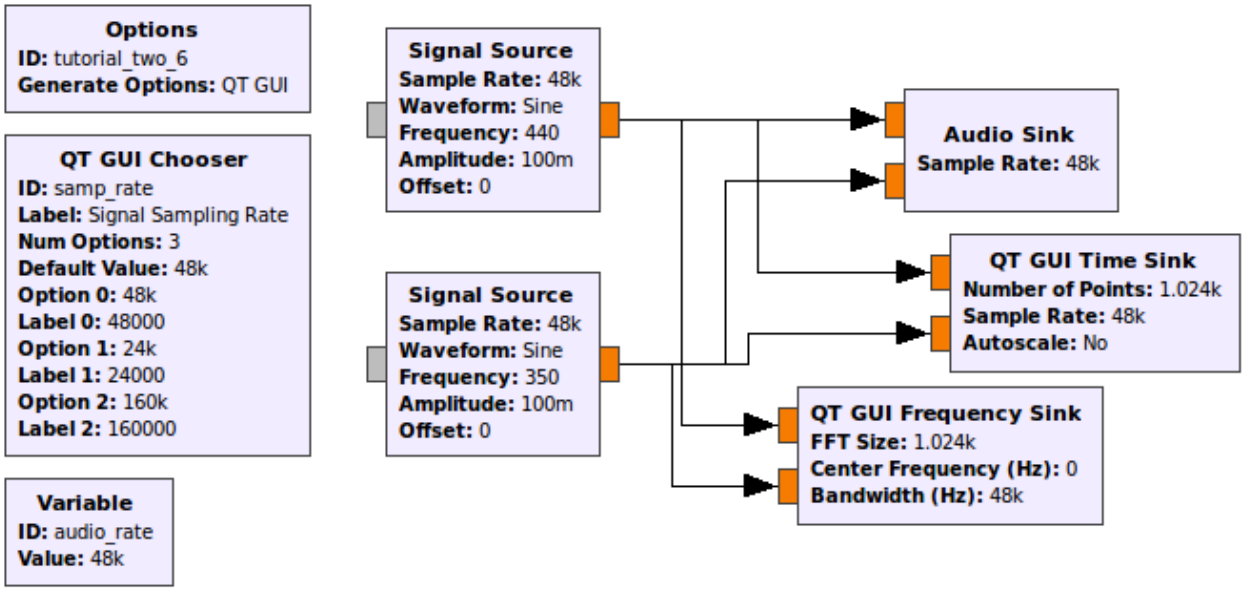

Figure 2: GNU Radio UI

SDR On Heterogeneous Systems

Some SDR projects are needed to combine with specialized hardware components, e.g. digital signal processors (**DSP**), application-specific integrated circuits (**ASIC**), and field-programmable gate arrays (**FPGA**).

Example

- SDR projects that are implemented in both hardware and software on a platform that comprises both processor and FPGA
- There are other SDR projects that are implemented using Xilinx Zynq SoC, utilizing both the PS/ARM processor and PL/FPGA fabric

SDR On Heterogeneous Systems

The USRP is a typical SDR device utilizing heterogeneous systems. The architecture of the USRP is shown as Figure 3.

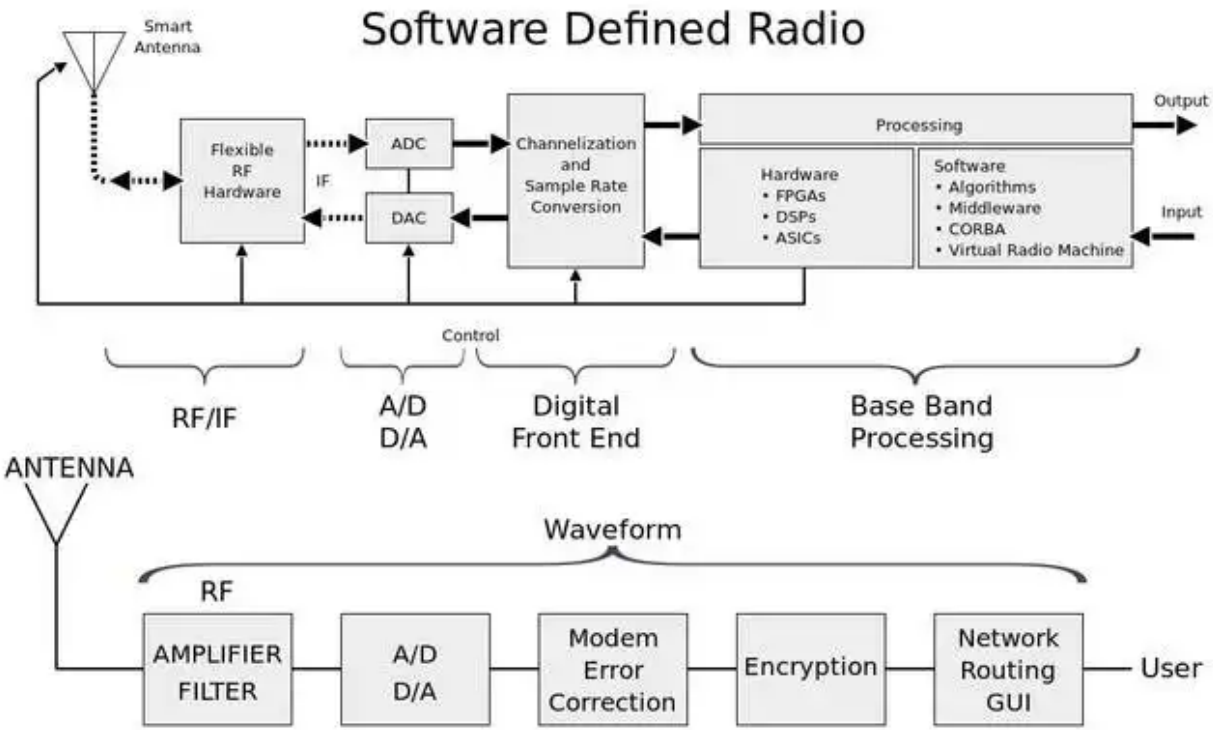

Figure 3: Architecture of USRP

Slot-Time Synchronized Operations

Any IEEE 802.11-based wireless transceiver implementation must operate based on time slot timing, for example, having a node wait to back off before sending a packet.
Timing synchronization of transmitter and receiver - synchronization of the same node at a certain state.

- **TX state**: when a node (DTx or DRx) enters to TX state, it sends samples in the transmit buffer and ignores all samples in the RX buffer.
- **RX state**: it retrieves samples from the RX buffer for processing and puts zeros into the send buffer. In this way, we ensure that the samples in the TX and RX buffers are current and relevant.

Slot-Time Synchronized Operations

An outline of such a procedure in one single time slot is illustrated as Figure 4.

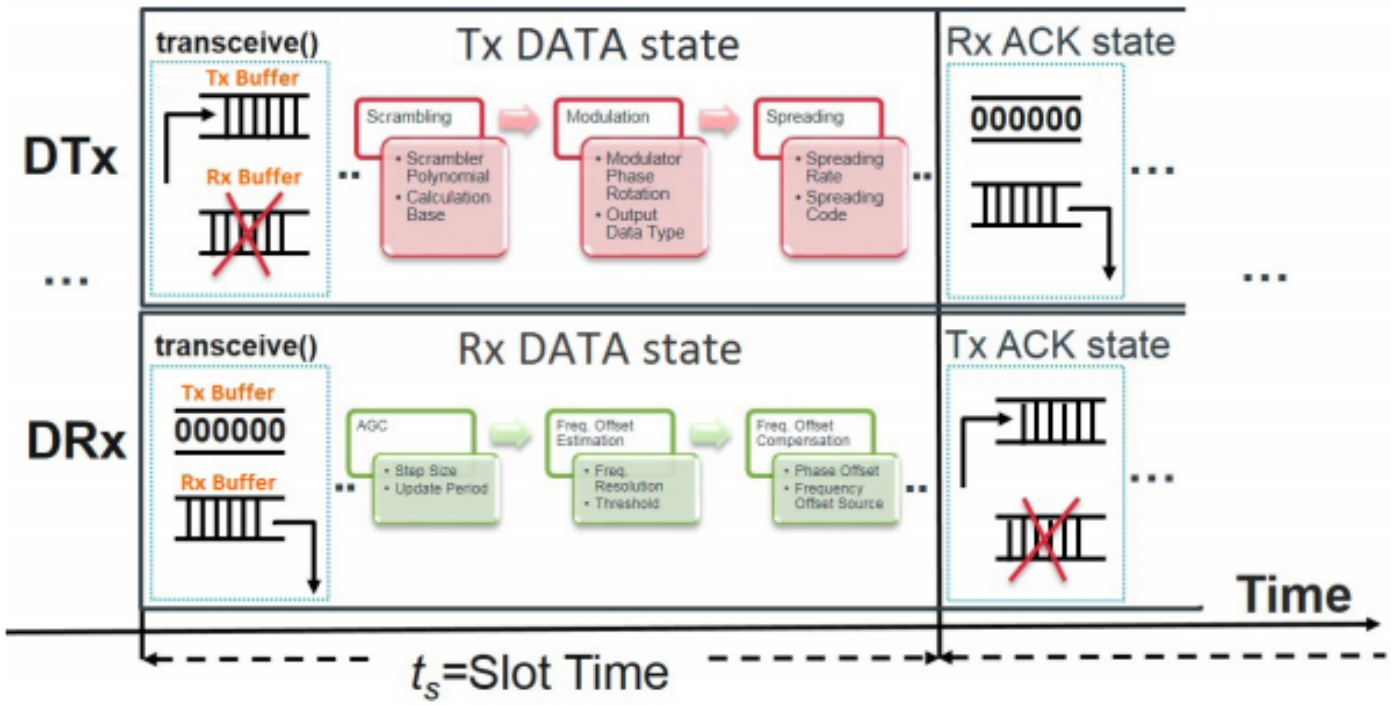

Figure 4: Time-Slot based Procedures

Slot-Time Synchronized Operations

If the receiver does not receive enough data, the `step()` call of the receiver object will return 0 as the length of the received frame. Once sufficient data has been collected by the radio, the next call will return a non-zero length value and valid data. Since we know the sampling rate of the data and the number of samples in a single frame, we can calculate how long it takes to get a frame of data from the radio. The `while` loop blocks the transceiver function until a data frame is received. Thus, we can use the duration of the call to this function as our clock source.

Code Implementation of Time Synchronize

The code structure of the time synchronization is shown as Figure 5.

```
function dr = transceive(ft, d2s)
persistent hrx htx;
% Initialize received data variables
dr = complex(zeros(nspf,1));
ns = 0;
% Initialize system objects once
if isempty(hrx)
  hrx = ...; htx = ...;
end
% Flag to release system objects
if ft
  release(hrx); release(htx);
else
  step(htx,d2s);
  while (ns == 0)
    [dr,ns] = step(hrx);
  end
end
```

Figure 5: Time Synchronization Code

Designed Transmitter State Machine

There are 4 status in the transmitting procedure.

1. **Energy detection** Determine the state of the channel by energy detection: idle or congested.
   - idle: start sending data
   - occupied: start CSMA/CA backoff
2. **Initialize Transmitting** Start the transmit procedure and send the packets.
3. **ACK determination** Determine whether the transmission is success or fail by the ACK packet.
   - Received ACK: Check the correctness of the received MAC packet prefix, and start another transmission. Remove the transmitted packet from the TX buffer.
   - No ACK message received: packet transmission error due to channel congestion, packet conflict with another STA's transmission, retransmittion required.

Designed Transmitter State Machine

4. **Transmission termination** When the TX buffer is emptied, the transmission is terminated.

The whole process of transmission is illustrated as Figure 6.

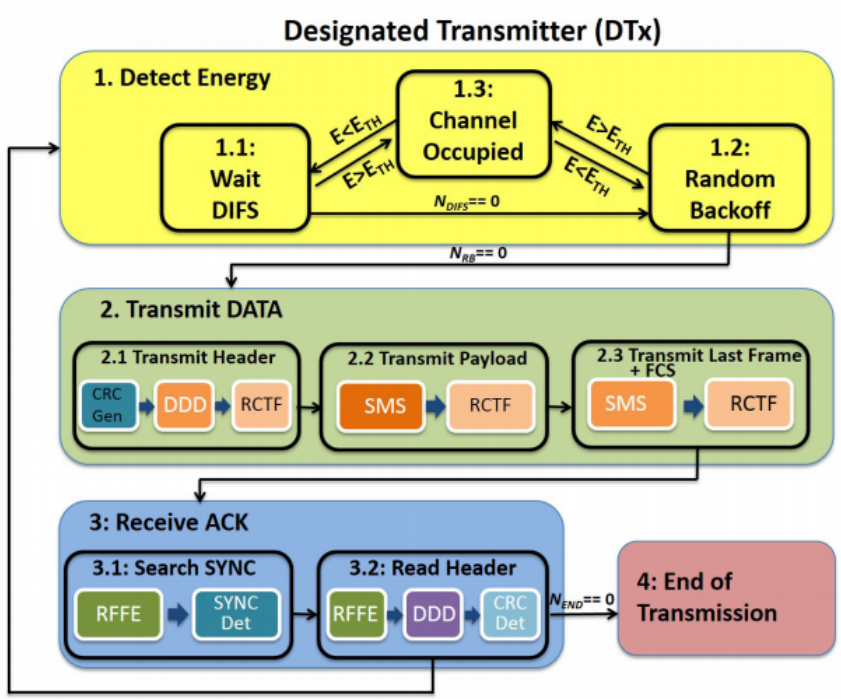

Figure 6: TX State Machine

Designed Receiver State Machine

The energy detection is **NOT** required in the receiver, therefore this state is removed from the receiver state machine. Thus, there are 3 status in the receiver state machine.

1. **Data reception**: Decode the PHY header and MAC header, and then read the valid information.
2. **Wait for SIFS**: wait for a SIFS time interval before sending an ACK reply packet to the transmitter
3. **ACK reply**: When the receiver successfully detects all the payload bits, sends an ACK message to the transmitter

Designed Receiver State Machine

The block diagram of the receiver state machine is shown as Figure 7.

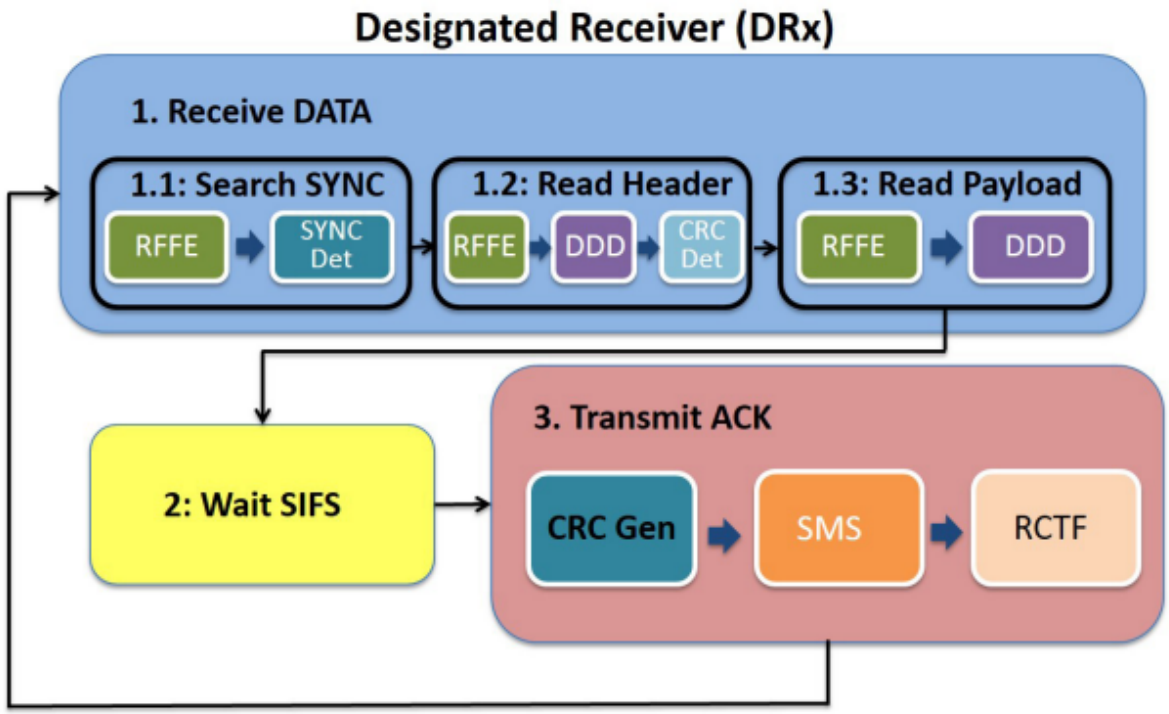

Figure 7: Receiver State Machine

System Blocks

There are sequential operations that need to be performed in each sub-state of the diagram. We segment these operations into multiple reusable blocks, as shown in the Figure 8 below. Identifying the grouping of blocks with related sub-states helps to better organize and reorganize the implemented code.

System Blocks

Block	Block Components
SMSRC	Scrambling, Modulation, Spreading, and Raised Cosine Transmit Filter (RCTF)
RFFE	Radio Frequency Front End: includes Automatic Gain Control (AGC), Coarse Frequency Offset Estimation (CFOE), Frequency Offset Compensation (FOC), and Raised Cosine Receive Filter (RCRF)
PD	Preamble/SYNC Detection: Find SYNC in Rx'd USRP frames
DDD	Despreading, Demodulation, and Descrambling

Figure 8: Sequential System Blocks

RF Front End Algorithms

The RF frontend, or the digital baseband, includes all the system blocks required in a typical quadrature receiver:

- raised cosine filtering(RCF)
- automatic gain control (AGC)
- frequency offset estimation and compensation

MATLAB provides multiple blocks implementing similar functions. The consideration in choosing the implementation is maximizing the speed.

RF Front End Algorithms

The selection of the modules shown in Figure 9 maximizes the computation speed.

```
(1) dsp.FIRDecimator('DecimationFactor',22);
(2) comm.PSKCoarseFrequencyEstimator(
        'Algorithm','FFT-based', ...
        'FrequencyResolution',cef,...
        'ModulationOrder',2,...
        'SampleRate',(2e5/22));
```

Figure 9: MATLAB Block Selection

Preamble Detection Algorithm

Take the cross-correlation of the received complex data after raised cosine fifiltering as the criteria of the decision. Multiple algorithms for the cross correlation is compared in Figure 10.

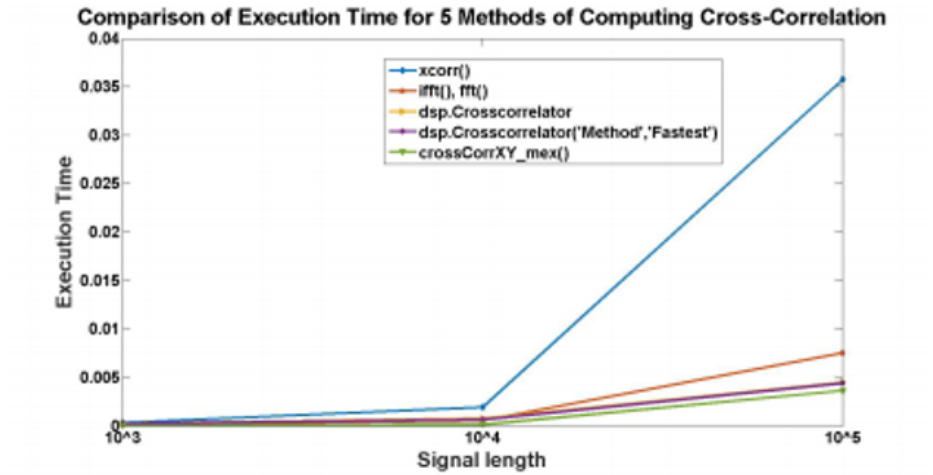

Figure 10: Performance Comparation

The benchmarking result indicates that the cross correlation function provided by the DSP toolbox is the fastest after C++ native compilation.

Parameter Selection

- USRP & 802.11 packets: Large frame size versus small frame size
- Key parameters (seeing below)

$$\begin{aligned} t_{\text{sample}} &= R_i/(100 \times 10^6) \\ &= 5 \times 10^{-6} \\ t_{\text{radio}} &= L_f R_i/(100 \times 10^6) \\ &= 7.04 \times 10^{-3} \end{aligned} \tag{1}$$

The parameters of the RFFE(RF Front End) can be tuned and optimized.

Same-Frequency Channel Operation

The receive-only port, RF2(RX2), of the USRP leaks about 7 dBm into the transmit & receive port, RF1(TX1/RX1). The effect of this leakage causes the DTx to detect the preamble in its own DATA packet while it is waiting for an ACK.

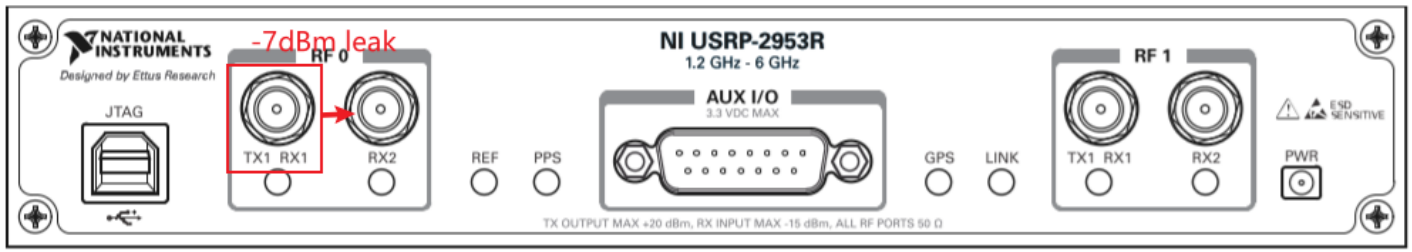

Figure 11: USRP Front Panel and Leak

CSMA/CA Overview

The MAC layer employs the Distributed Coordination Function (DCF) strategy incorporating the CSMA/CA mechanism as it is described in the IEEE 802.11 specification.
The state machine structure of the DCF and CSMA/CA algorithm consists of 3 steps:

- Energy detection
- DIFS period
- Binary exponential backoff

These steps has already been discussed thoroughly in the previous lectures of the course.

CSMA/CA Overview

The time sequential illustration of the DCF and CSMA/CA algorithm is shown in Figure 43.

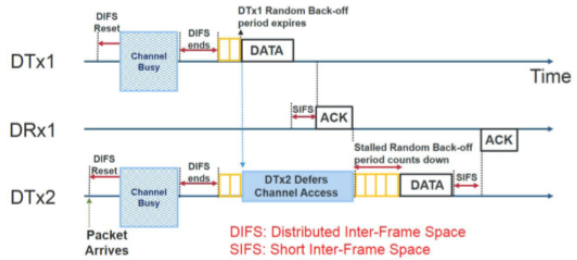

Figure 13: CSMA/CA Time Schedule

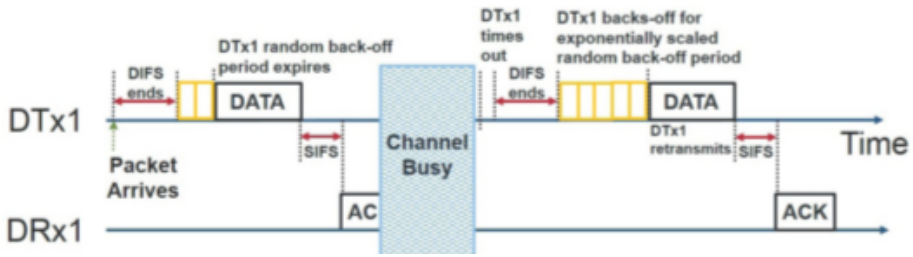

Figure 14: CSMA/CA Energy Detection and Random Backoff

Implementation of CSMA/CA

1. **Energy Detection** The energy of the signal is evaluated using the easiest 2nd norm.

$$E = \|x[n]\|^2 = \sum_{n=1}^{N} |x[n]|^2 \tag{2}$$

2. **DIFS** The DIFS period is defined as the specification of the 802.11 standard.
3. **Random Backoff** Use the binary exponential algorithm to generate a random backoff as the 802.11 standard definition.

```
1 function [backoff] = random_backoff(k,t_radio)
2     backoff = randi([0,2^k])*t_radio;
3 end
```

Experimental Setups

MATLAB operating on the Ubuntu 20.04 is used for the SDR backend software, and the HW support package for the USRP is the SDRu support package to cooperate with USRP N210.

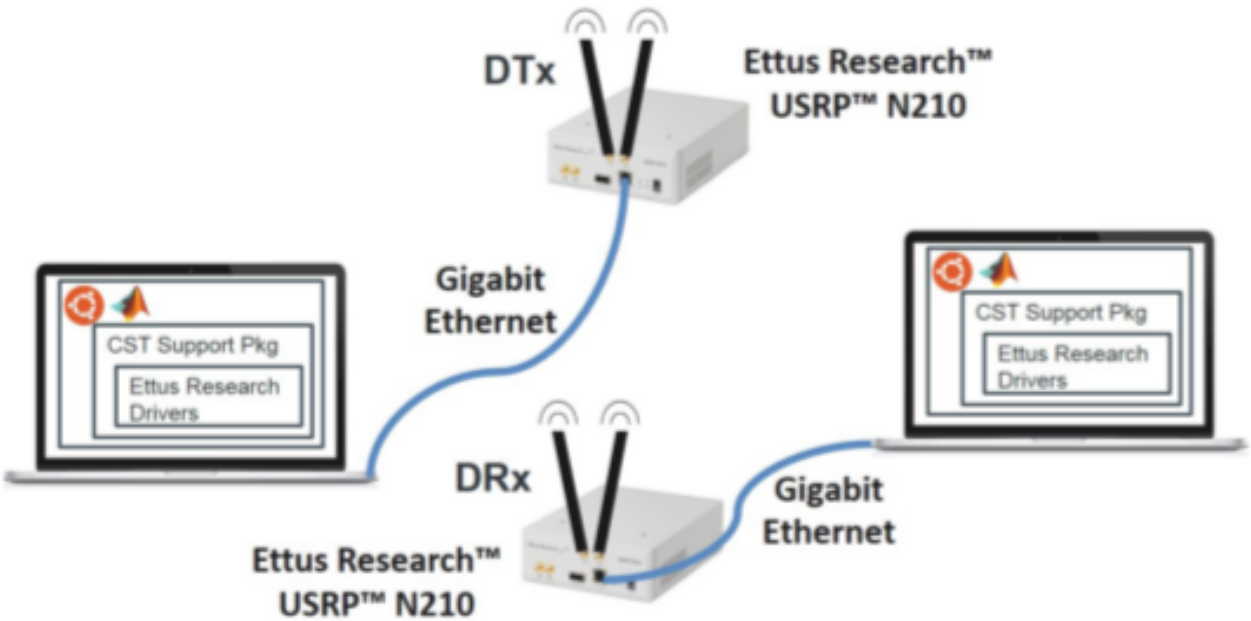

Figure 14: HW Setup

Experimental Setups

- Software Configuration The Communication Toolbox and the SDRu support package are used for the SDR backend software.
- MATLAB Coder The MATLAB code involved in the experiment is compatible to MATLAB coder compilation, therefore the C++ version of all the algorithms can be easily obtained.

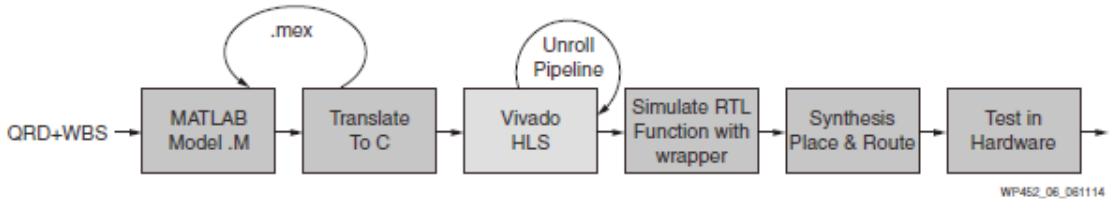

Figure 6: **Adaptive Beamformer Design Flow Using HLS**

Figure 15: MATLAB to C/Verilog Code Generation

Timing Data Packet Reception At DRx

- DTx: 258 USRP Frames
- Permeable: 2 USRP Frames/128 Bits
- DRx: 258-2=256 USRP Frames

The time to process any given frame usually falls below the desired frame time, t_{radio}, and is fairly constant at 2.87 ms. The first set of frames have a higher processing time because they consist of the **MAC header** information that must be resolved (e.g. frame control, MAC address).

RFFE Block Timing

The performance of RFFE running on the MATLAB interpreter and the compiled MEX file (Optimized DLL) is shown in Figure 16. It's obvious that the MEX/DLL boasts a higher performance than the MATLAB interpreter.

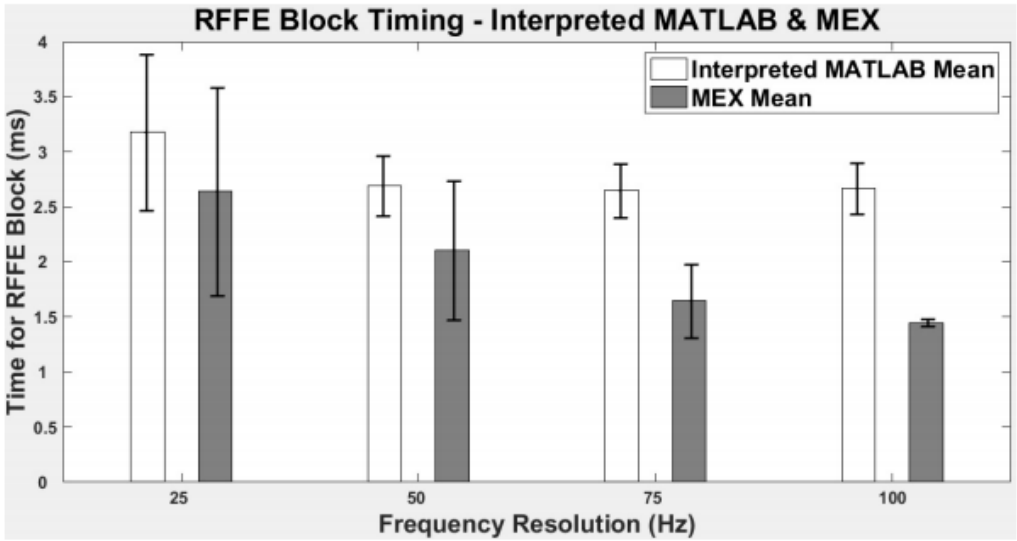

Figure 16: RFFE Block Timing

2 Node Performance: PER and Latency

The experiment was designed to be statistically significant, and hence, 100 packets were transmitted for each of the 5 different transmit gain settings. The results were averaged over 5 runs. The spacing between the nodes is set to be 1 meter approximately. The two criteria below are accounted for in the experiment:

1. **PER** The packet error rate is calculated as the number of errors divided by the number of packets.
2. **Bi-Directional Link Latency** Bi-directional link latency is the average time taken by the DTx between sending a DATA packet and receiving the corresponding ACK packet.

2 Node Performance: PER and Latency

The PER result is shown in Figure 17.

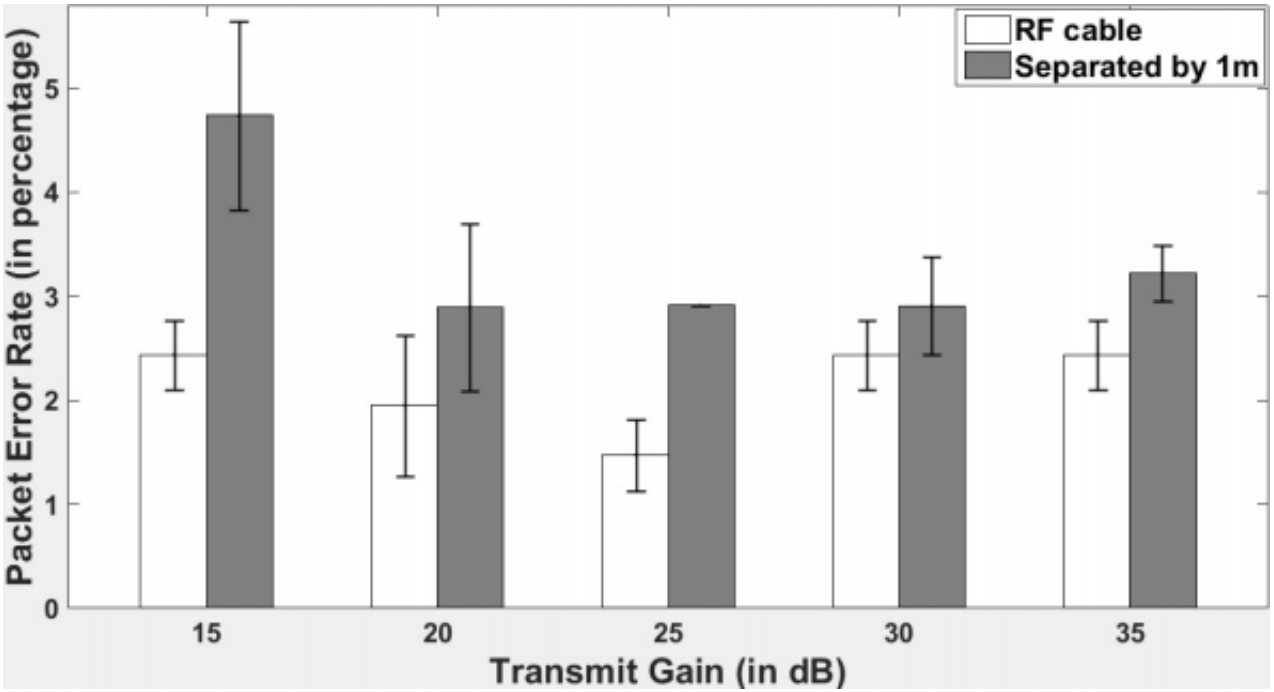

Figure 17: Package Error Rate

2 Node Performance: PER and Latency

Importantly, varying the distance between the 2 nodes **does not** significantly affect performance. Even moving the 2 nodes farther apart while still in line-of-sight (e.g. by 15 meters), the PER and bidirectional link latency stayed consistent. However, the presence of many metallic surfaces, such as in our lab setting, give rise to multipath reflections that can be strong and result in packet errors. The fact that the performance was significantly better when the nodes were connected by RF cables confirms the case

2 Node Performance: PER and Latency

The Latency result is shown in Figure 18.

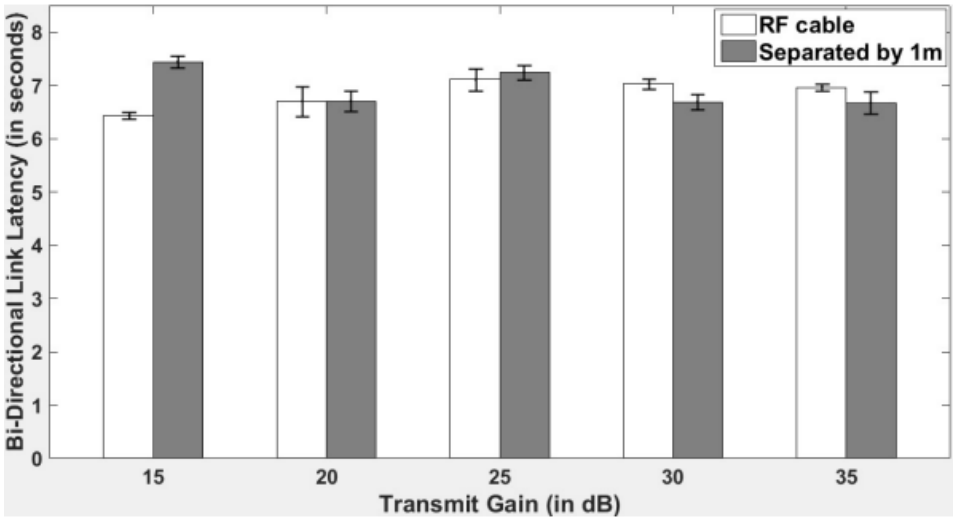

Figure 18: Bi-Directional Link Latency

In the two node system, increasing DIFS and backoff time practically has no effect on the packet error rate due to lack of contention. However, increasing DIFS and backoff time also increases link latency by the same amounts.

2 Node Performance: PER and Latency

The idealized delay of the link in the experiment is (ignoring channel contention, blocking time and retransmission)

$$\begin{aligned} L_{\text{STDLink}} &= \text{DIFS} + \text{TxData} + \text{SIFS} + \text{TxACK} \\ &= 50 + (2072 \times 8) + 10 + (40 \times 8) \\ &= 16596\mu s = 16.596ms \end{aligned} \tag{3}$$

This idealized delay is at the same order of each time slot.

Profile Of Time Elapsed In DTx States

At the DTx, we measured the time elapsed in each state for a DATA-ACK packet exchange. The stacked plots shown in Figure 19(a) and Figure 20(a) show the breakdown of the time spent in each substate.

Profile Of Time Elapsed In DTx States

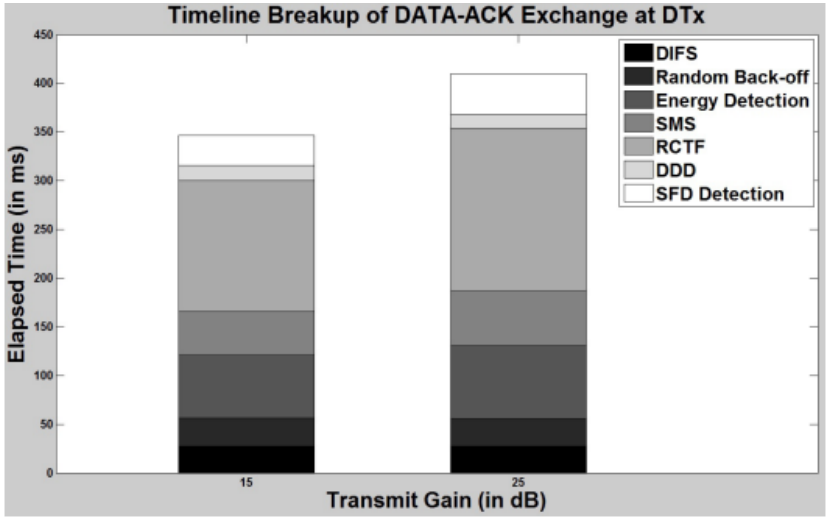

Figure 20: Timeline Breakup:All components

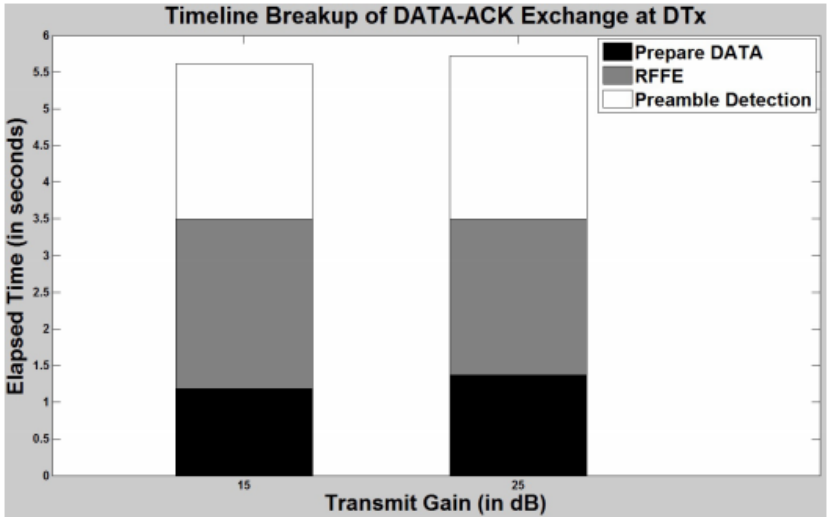

Figure 21: Timeline Breakup:Large

3 Node System Configuration

The configuration of the 3 node transmission test is shown in the Figure 21.
We now have two independent links incident on one shared DRx, and hence, we do not expect to see much difference in the performance of the two links when varying to transmit gains here in the 3 node case.

Main goals

1. Ensure certain transmission performance (Goodput)
2. Fairness of two channel competition (Fairness)

3 Node System Configuration

The 3 node system is shown in Figure 21.

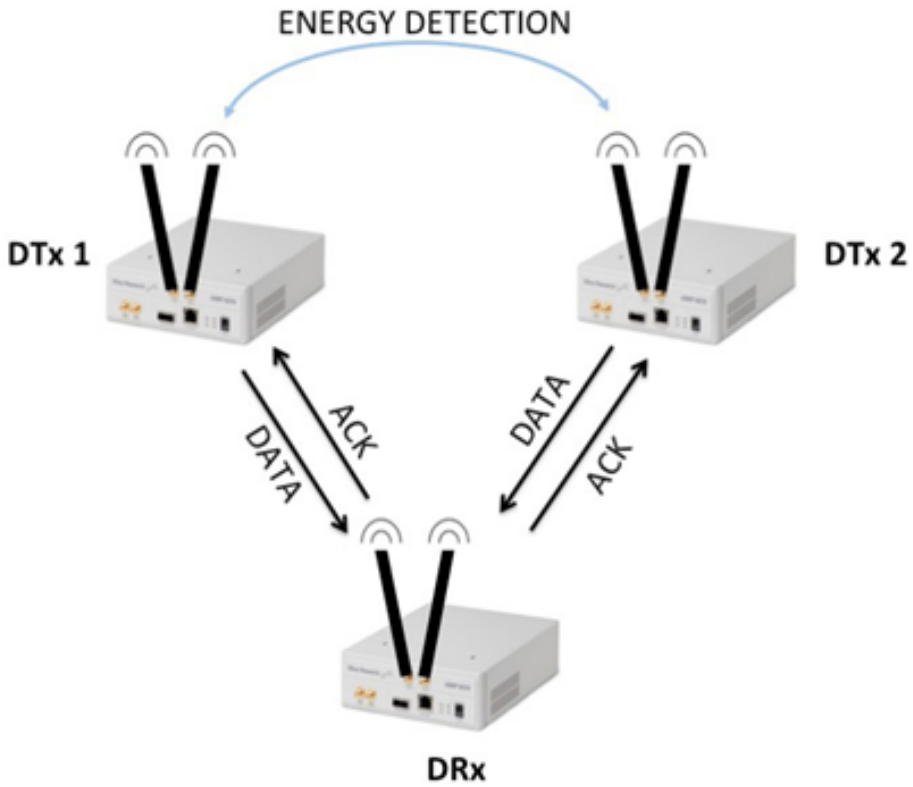

Figure 21: 3 Node System Configuration

Data and ACK Frame Structure

The data and ACK frames are shown in Figure 67.

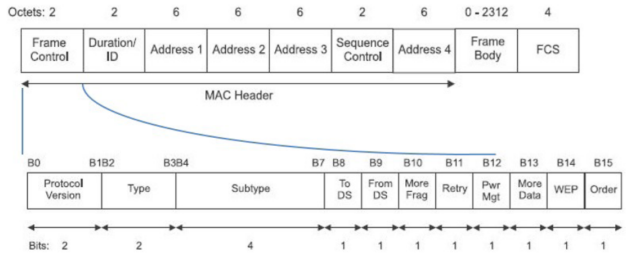

Figure 23: MAC header of Data Frame

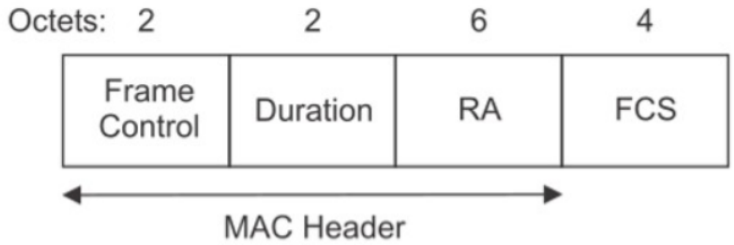

Figure 24: MAC Header of ACK Frame

3 Node Performance: PER and Latency

The result of PER and Latency is shown below in Figure 69.

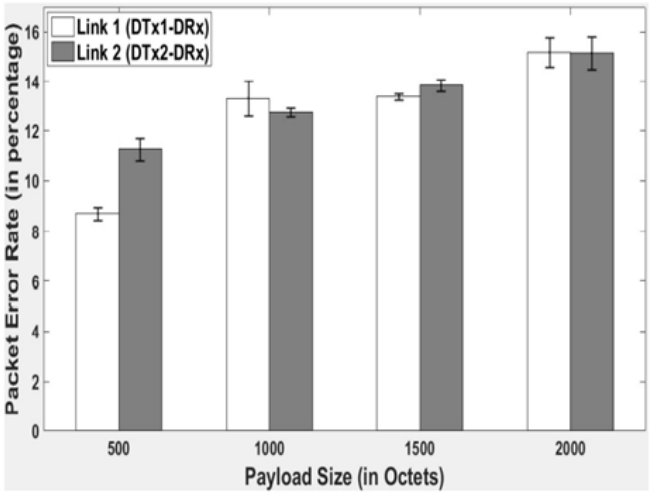

Figure 25: PER Comparison of 3 Node System

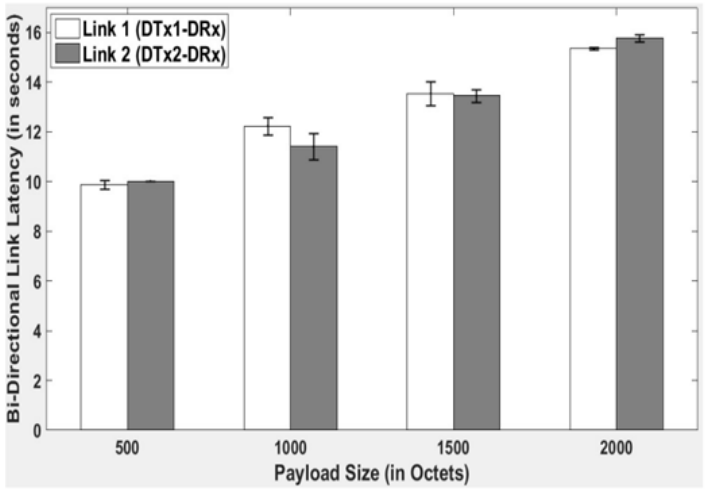

Figure 26: Latency Comparison of 3 Node System

This result indicates that the two TX STAs shares the bandwidth fairly.

Goodput in transmission

Goodput, a performance measure used in computer networks, is the rate at which useful information bits traverse a link. Goodput can be measured using equation

$$G = \frac{\text{Total payload bits correctly decoded}}{\text{Average Bidirectional Latency}} \tag{4}$$

The table below shows the goodput of the 3 node system.

Payload Size (#Octets)	Link 1 Goodput (Kbps)	Link 2 Goodput (Kbps)
500	0.41	0.40
1004	0.66	0.70
1500	0.89	0.89
2004	1.05	1.02

Figure 26: Average Goodput

Conclusion and Analysis

Advantages

1. IEEE 802.11b standard compliant.
2. User could reconfigure the parameter values as needed.
3. Modularity and extensibility.
4. Achieved a high fairness in multi node.

Disadvantages

1. It had trouble realizing slot-synchronized operations.
2. It is difficult to pick the right energy threshold.
3. Parameter setting instability.

THANKS & QUESTION

Thanks for Listening!