Aalto-yliopisto
Aalto-universitetet
Aalto University

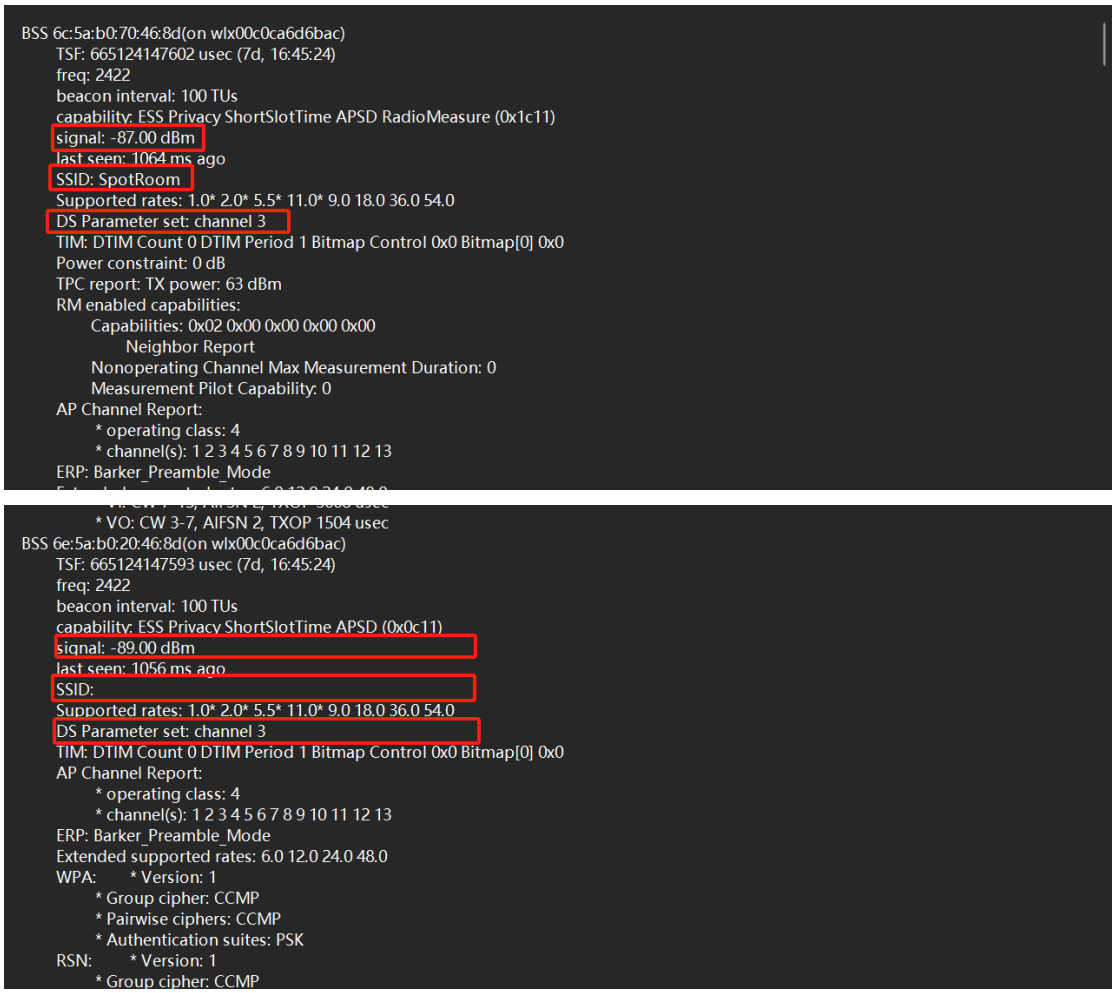
ELEC-E7250-Laboratory Course in Communications Engineering

Measurement report: WLAN measurements

Group Member: Xingji Chen, Zheyuan Liu

1.1.1 Make a table for all transmitters you observe: name, used channel, power level.

According to the file wlanscan.txt, we could get the information from the marked places.



The table of the observation is as follows,

Name	Used Channel	Power Level
SpotRoom	Channel 3	-87.00dBm
None (Hidden)	Channel 3	-89.00dBm

2.4 Laboratory report tasks

2.4.1 Include tables of all measured cases

The tables of all measured cases are as follows,

PacketSize / Throughput	300	500	700	900	1100	1300	1470
802.11g 54M(Mbits/sec)	7.26	11.0	14.3	17.1	19.5	22.5	23.0
802.11n 72M(Mbits/sec)	18.9	31.4	34.9	40.8	46.4	49.4	51.3

The value could be got by this part of the command window,

```
iperf -c 10.0.0.2 -b 54M -u -lPacketSize -t30 # PC1 PacketSize = 300 500 700 900 1100 1300 1470
iperf -s -u -i5  #PC2
------------------------------------------------------------
Server listening on UDP port 5001
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  1] local 10.0.0.2 port 5001 connected with 10.0.0.3 port 48201
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  1] 0.0000-5.0000 sec  4.33 MBytes  7.26 Mbits/sec   0.392 ms 361/15482 (2.3%)
[  1] 5.0000-10.0000 sec  4.31 MBytes  7.24 Mbits/sec   0.335 ms 640/15722 (4.1%)
[  1] 10.0000-15.0000 sec  4.32 MBytes  7.25 Mbits/sec   0.361 ms 585/15680 (3.7%)
[  1] 15.0000-20.0000 sec  4.33 MBytes  7.27 Mbits/sec   0.623 ms 485/15627 (3.1%)
[  1] 20.0000-25.0000 sec  4.35 MBytes  7.29 Mbits/sec   0.350 ms 526/15719 (3.3%)
[  1] 25.0000-30.0000 sec  4.31 MBytes  7.23 Mbits/sec   0.433 ms 630/15689 (4%)
[  1] 0.0000-30.0446 sec  26.0 MBytes  7.26 Mbits/sec   1.902 ms 3227/94053 (3.4%)
[  3] WARNING: ack of last datagram failed.
[  2] local 10.0.0.2 port 5001 connected with 10.0.0.3 port 49739
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  2] 0.0000-5.0000 sec  6.60 MBytes  11.1 Mbits/sec   0.389 ms 635/14475 (4.4%)
[  2] 5.0000-10.0000 sec  6.58 MBytes  11.0 Mbits/sec   0.531 ms 1568/15369 (10%)
```

2.4.2 Plot all the results into one figure

To plot all values into one figure, I use the python code as follows,

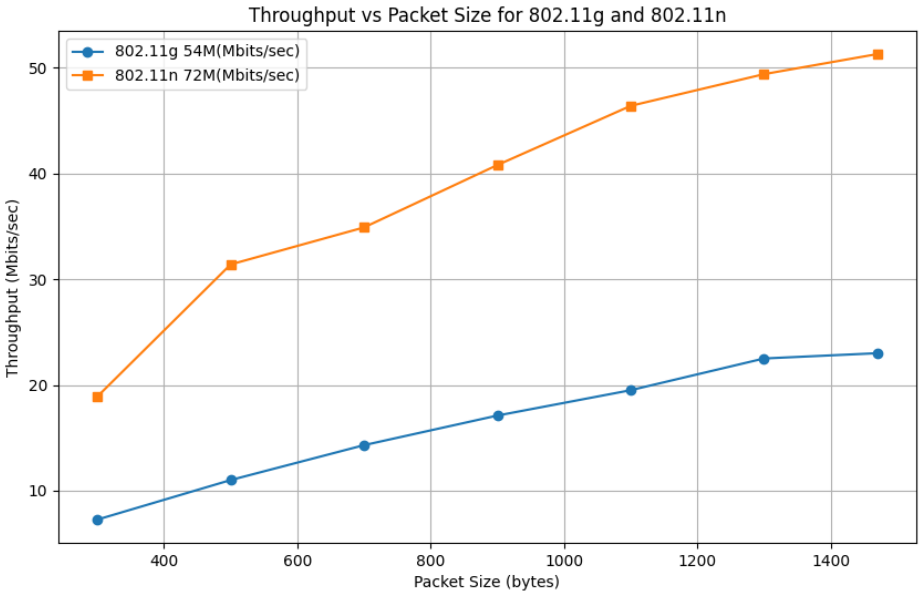
```python
import matplotlib.pyplot as plt

# Data
packet_sizes = [300, 500, 700, 900, 1100, 1300, 1470]
throughput_802_11g = [7.26, 11.0, 14.3, 17.1, 19.5, 22.5, 23.0]
throughput_802_11n = [18.9, 31.4, 34.9, 40.8, 46.4, 49.4, 51.3]


# Plotting
plt.figure(figsize=(10, 6))
plt.plot(packet_sizes, throughput_802_11g, marker='o', label='802.11g 54M(Mbits/sec)')
plt.plot(packet_sizes, throughput_802_11n, marker='s', label='802.11n 72M(Mbits/sec)')

plt.xlabel('Packet Size (bytes)')
plt.ylabel('Throughput (Mbits/sec)')
plt.title('Throughput vs Packet Size for 802.11g and 802.11n')
plt.legend()
plt.grid(True)
plt.show()
```

The plotting figure is as follows,



2.4.3 Comment on the results

Why is the throughput lower with shorter packet sizes?

Each packet sent over a network includes not only the payload (the actual data being transmitted) but also header information which is necessary for routing, error checking, and other networking functionalities. The header size is generally fixed regardless of the payload size. Thus, with smaller packets, the proportion of overhead to payload increases, meaning a larger percentage of the bandwidth is used to transmit non-payload information. This results in lower effective throughput for the actual data.

Many networking protocols require acknowledgments for received packets and may involve handshaking mechanisms to establish connections before data transmission begins. When packets are smaller, more packets are needed to send the same amount of data, leading to more acknowledgments and potentially more handshaking overhead, which can further reduce throughput.

Each packet, regardless of its size, requires processing by networking hardware and software, including routers, switches, and the end devices. This processing includes tasks such as error checking, routing decision-making, and buffering. With smaller packets, the number of packets increases for a given amount of data, leading to higher processing demands which can saturate the processing capabilities of the network

infrastructure and devices, slowing down overall data transmission. The time it takes to transmit a packet on the network includes both a fixed and a variable component. The fixed component is associated with the processing overhead mentioned earlier, while the variable component is related to the size of the packet. For smaller packets, the fixed component becomes a larger fraction of the total transmission time, reducing the efficiency of the network.

Some network protocols have efficiency mechanisms that work better with larger packets. For example, TCP (Transmission Control Protocol) has mechanisms like window scaling, which are more efficiently utilized with larger amounts of data being sent before requiring an acknowledgment. With smaller packets, these mechanisms cannot be utilized to their full potential, potentially leading to reduced throughput.

Both 802.11g 54 Mbps and 802.11n 65/72.2 Mbps modes use 64QAM modulated OFDM subcarriers with 5/6 error coding rate. What is the reason for difference in nominal bitrate? What could explain the even larger difference in practical throughput? [Hint: Study the differences between 802.11n and 802.11g physical layer and MAC layer features.]

Physical Layer Differences

Multiple Input Multiple Output (MIMO): 802.11n utilizes MIMO technology, which allows multiple data streams to be transmitted simultaneously over the same channel, significantly increasing the data rate. In contrast, 802.11g uses a single data stream, limiting its throughput.

Channel Bonding: 802.11n supports channel bonding, which combines two 20 MHz channels into a single 40 MHz channel, doubling the channel width and increasing the maximum data rate. 802.11g operates on a single 20 MHz channel.

Improved Modulation: 802.11n uses higher-order modulation (up to 64-QAM) compared to 802.11g (which uses up to 54-QAM), allowing more bits to be transmitted per symbol and thus increasing the throughput.

Short Guard Interval (SGI): 802.11n can use a shorter guard interval between symbols to reduce overhead and improve throughput. The guard interval in 802.11n can be as short as 400 nanoseconds, compared to 800 nanoseconds in 802.11g.

MAC Layer Differences

Frame Aggregation: 802.11n introduces frame aggregation techniques (A-MPDU and A-MSDU) that allow multiple data frames to be sent in a single transmission, significantly reducing the overhead caused by headers and acknowledgments. This is particularly beneficial for smaller packet sizes, as it mitigates the overhead's impact on throughput.

Block Acknowledgment: 802.11n utilizes a block acknowledgment scheme that allows multiple packets to be acknowledged with a single acknowledgment frame, reducing the overhead and latency associated with acknowledgments and thus improving throughput.

Improved Power Save Mechanisms: 802.11n enhances power-saving mechanisms, which, while not directly affecting throughput, can indirectly benefit performance by managing the energy consumption of devices more efficiently, allowing them to perform at higher levels for longer periods.

Even larger difference in practical throughput:

Environmental Conditions: Physical obstructions, interference from other wireless devices, and multipath propagation can affect signal quality and throughput.

Network Congestion: The more devices that are connected and communicating over the network, the more contention there is for the medium, which can significantly reduce throughput.

Distance from the Access Point: The farther a device is from the access point, the weaker the signal, which can lead to lower data rates or the need for retransmissions.

802.11n's Enhanced Features: The enhancements of 802.11n not only provide higher nominal rates but also make the standard more resilient to interference and signal degradation, leading to a larger difference in practical throughput when compared to 802.11g.

3.4 Laboratory report tasks

3.4.1 Give the tables of the attenuation versus throughput.

The tables of the attenuation versus throughput are as follows,

Throughput\Attenuation	0	5	10	15	17	19	21	23	25
UDP (Mbits/sec)	51.5	51.4	48.8	31.6	30.0	20.8	18.2	13.0	1.92
TCP (Mbits/sec)	41.9	43.9	40.0	26.8	20.6	16.4	10.5	5.56	0.36

The value could be got by this part of the command window,

```
iperf -c 10.0.0.2 -b 72M -u -l1470 -t10 # PC1   #set the attenuator to 0 5 10 15 17 19 21 23 25 dB UDP
iperf -s -u -i5   # PC2 get the throughput , plot the throughput and recieved signal level
------------------------------------------------------------
Server listening on UDP port 5001
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  1] local 10.0.0.2 port 5001 connected with 10.0.0.3 port 42648
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  1] 0.0000-5.0000 sec  30.7 MBytes  51.5 Mbits/sec   0.845 ms 0/21885 (0%)
[  1] 5.0000-10.0000 sec  30.7 MBytes  51.5 Mbits/sec   0.244 ms 0/21905 (0%)
[  1] 0.0000-10.0124 sec  61.5 MBytes  51.5 Mbits/sec   0.836 ms 0/43845 (0%)
[  2] local 10.0.0.2 port 5001 connected with 10.0.0.3 port 35470
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  2] 0.0000-5.0000 sec  30.7 MBytes  51.4 Mbits/sec   0.318 ms 0/21869 (0%)
[  2] 5.0000-10.0000 sec  30.6 MBytes  51.3 Mbits/sec   0.490 ms 0/21829 (0%)
[  2] 0.0000-10.0206 sec  61.4 MBytes  51.4 Mbits/sec   0.940 ms 0/43785 (0%)
[  3] local 10.0.0.2 port 5001 connected with 10.0.0.3 port 58376
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3] 0.0000-5.0000 sec  28.9 MBytes  48.4 Mbits/sec   0.394 ms 0/20597 (0%)
[  3] 5.0000-10.0000 sec  29.3 MBytes  49.2 Mbits/sec   0.353 ms 0/20921 (0%)
[  3] 0.0000-10.0142 sec  58.3 MBytes  48.8 Mbits/sec   1.133 ms 0/41573 (0%)
[  3] WARNING: ack of last datagram failed.
```

```
iperf -c 10.0.0.2 -t5  # PC1   # set the attenuator to 0 5 10 15 17 19 21 23 25 dB TCP
iperf -s -i5  # PC2
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  1] local 10.0.0.2 port 5001 connected with 10.0.0.3 port 42324
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-5.0000 sec  24.9 MBytes  41.8 Mbits/sec
[  1] 0.0000-5.1815 sec  25.9 MBytes  41.9 Mbits/sec
[  2] local 10.0.0.2 port 5001 connected with 10.0.0.3 port 57892
[ ID] Interval       Transfer     Bandwidth
[  2] 0.0000-5.0000 sec  26.2 MBytes  43.9 Mbits/sec
[  2] 0.0000-5.2054 sec  27.3 MBytes  43.9 Mbits/sec
[  3] local 10.0.0.2 port 5001 connected with 10.0.0.3 port 60424
[ ID] Interval       Transfer     Bandwidth
[  3] 0.0000-5.0000 sec  23.8 MBytes  40.0 Mbits/sec
[  3] 0.0000-5.3246 sec  25.4 MBytes  40.0 Mbits/sec
[  4] local 10.0.0.2 port 5001 connected with 10.0.0.3 port 45036
[ ID] Interval       Transfer     Bandwidth
[  4] 0.0000-5.0000 sec  16.0 MBytes  26.8 Mbits/sec
[  4] 0.0000-5.3115 sec  17.0 MBytes  26.8 Mbits/sec
```

3.4.2 Plot the throughput as a function of received signal level.

To plot the throughput as a function of received signal level, we need to first evaluate what were the corresponding received signal levels, as the values we record before, we could plot how the attenuation is mapped to the received signal level, by the python code as follows,

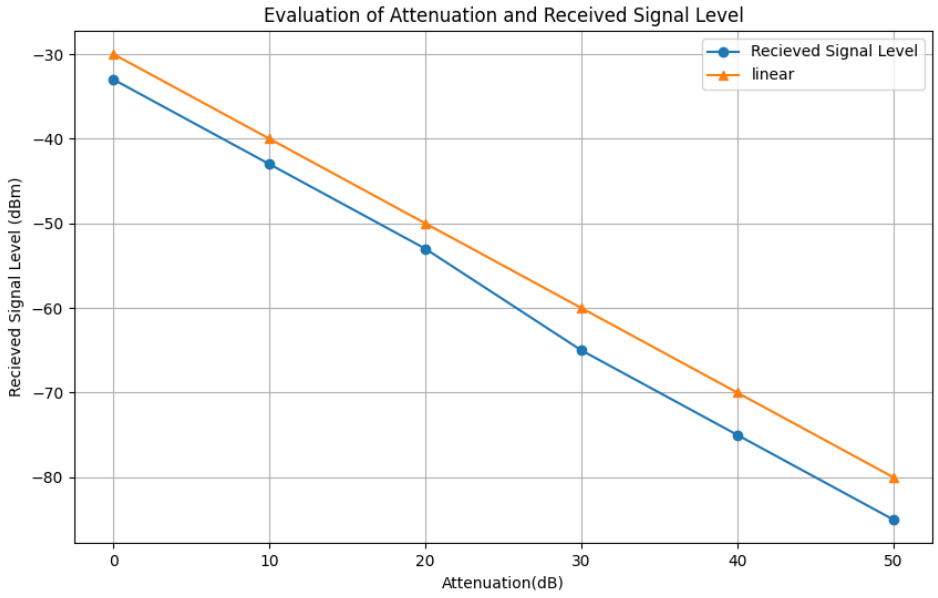
```
import matplotlib.pyplot as plt

# Data
attenuation = [0, 10, 20, 30, 40, 50]
received_signal_level = [-33, -43, -53, -65, -75, -85]
liniear = [-30,-40,-50,-60,-70,-80]

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(attenuation, received_signal_level, marker='o', label='Recieved Signal Level')
plt.plot(attenuation, liniear, marker='^', label='linear')

plt.xlabel('Attenuation(dB)')
plt.ylabel('Recieved Signal Level (dBm)')
plt.title('Evaluation of Attenuation and Received Signal Level')
plt.legend()
plt.grid(True)
plt.show()
```

The plotting figure is as follows,



We can know that the two are approximately linear, but not completely linear. In the period the attenuation from 0 to 25 dB, it could be considered linear.

So, we could create tables of the received signal levels versus throughput,

Throughput\Receivedlv	-33	-38	-43	-48	-50	-52	-54	-56	-58
UDP (Mbits/sec)	51.5	51.4	48.8	31.6	30.0	20.8	18.2	13.0	1.92
TCP (Mbits/sec)	41.9	43.9	40.0	26.8	20.6	16.4	10.5	5.56	0.36

Plot the throughput as a function of received signal level by the python code as follows,

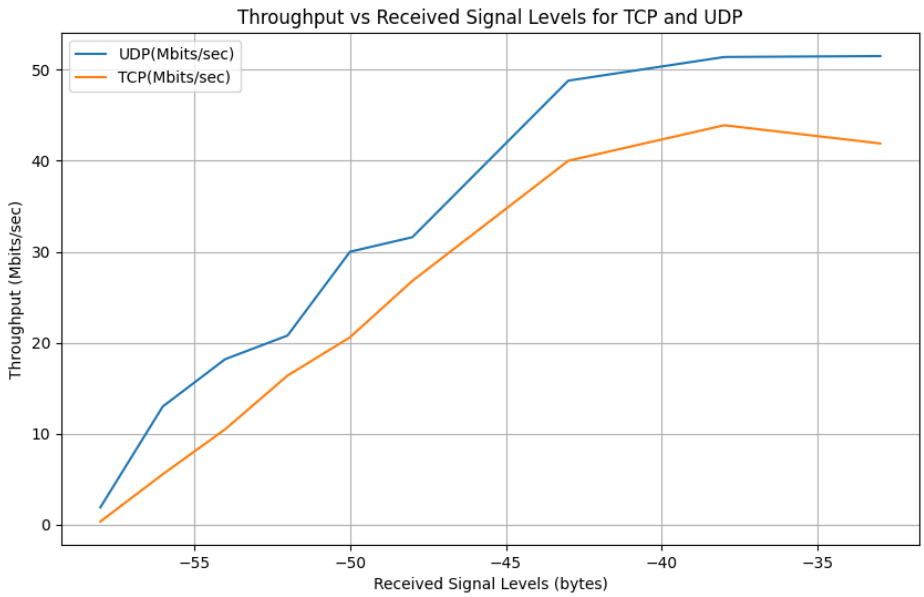
```
import matplotlib.pyplot as plt

# Data
signal_power = [-33, -38, -43, -48, -50, -52, -54, -56, -58]
udp = [51.5, 51.4, 48.8, 31.6, 30.0, 20.8, 18.2, 13.0, 1.92]
tcp = [41.9 , 43.9, 40.0, 26.8, 20.6, 16.4, 10.5, 5.56, 0.36]


# Plotting
plt.figure(figsize=(10, 6))
plt.plot(signal_power, udp, marker='None', label='UDP(Mbits/sec)')
plt.plot(signal_power, tcp, marker='None', label='TCP(Mbits/sec)')

plt.xlabel('Received Signal Levels (bytes)')
plt.ylabel('Throughput (Mbits/sec)')
plt.title('Throughput vs Received Signal Levels for TCP and UDP')
plt.legend()
plt.grid(True)
plt.show()
```

The resulting picture is as follows,



3.4.3 Assuming the same parameters as in part 1.1.1 regarding power levels and losses, what would be the cell size if

3.4.3.1 the attenuation model is $r^{-\alpha}$ and α = 3.5.

The lowest signal power is -58dBm, change the -58dBm to watt, there should be 0.001584 watt ,based on the equation, there should be $0.001584 = r^{-\alpha}$ and α = 3.5, we could get that r equals to 6.31m.

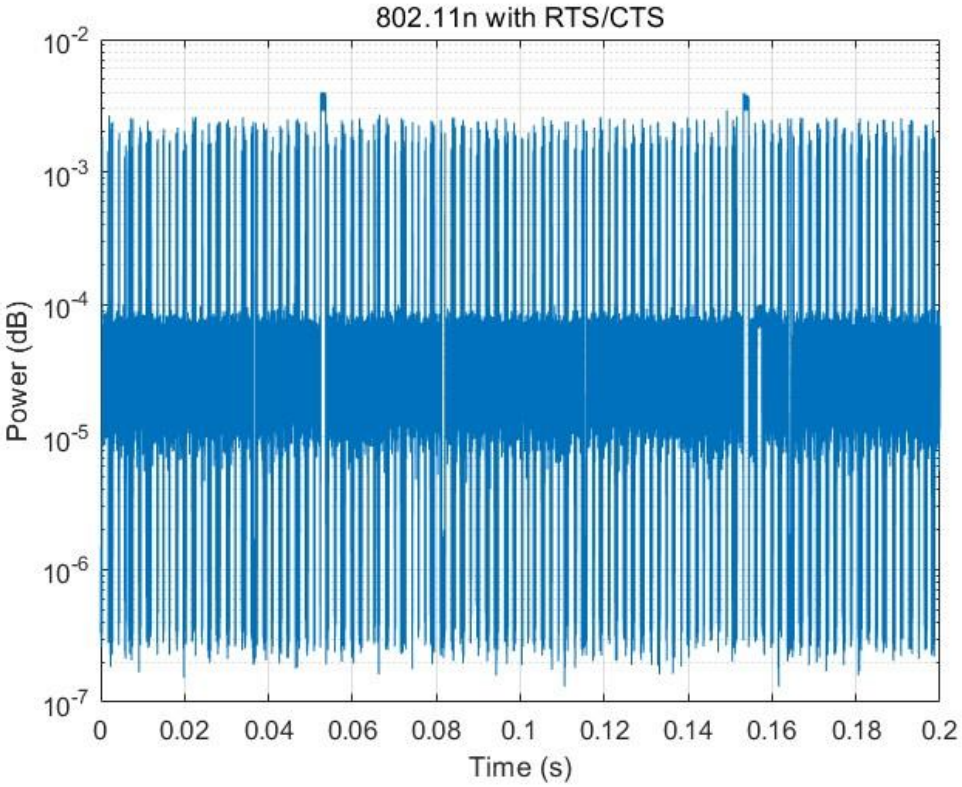
3.4.3.2 the attenuation model is $r^{-\alpha}$ where α = 3.5 and the signal goes through one wall. (-45dBm)

The lowest signal power is -58dBm, think about the wall, there should be -13dBm, change the -13dBm to watt, there should be 5.01 watt ,based on the equation, there should be $5.01 = r^{-\alpha}$ and α = 3.5, we could get that r equals to 0.631m.

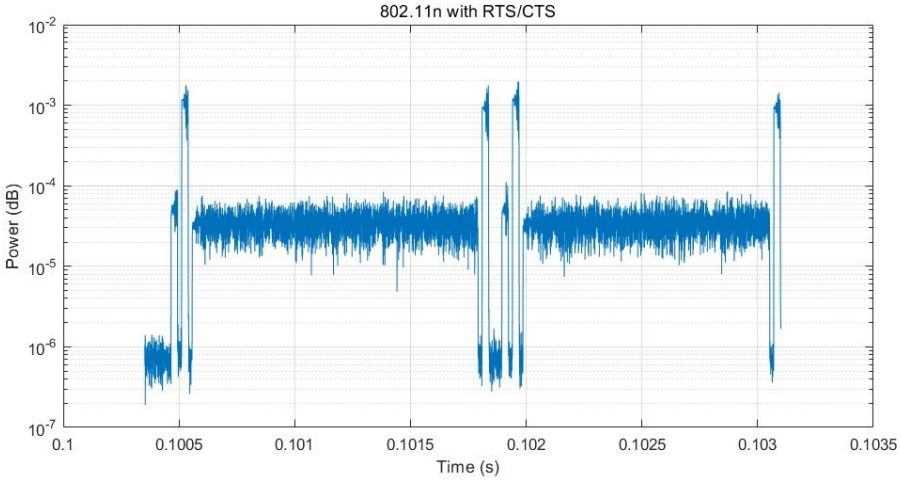
4.4 Laboratory report tasks

4.4.1 In each case plot power over time and zoom into the plot to see couple of data frames. Indicate in the plots the following: dataframe, acknowledgements, RTS and CTS frames (where applicable), short inter-frame space (SIFS), DCF inter-frame space (DIFS).

In the case of 802.11n with RTS/CTS, we could see the power over time plotting as follows:

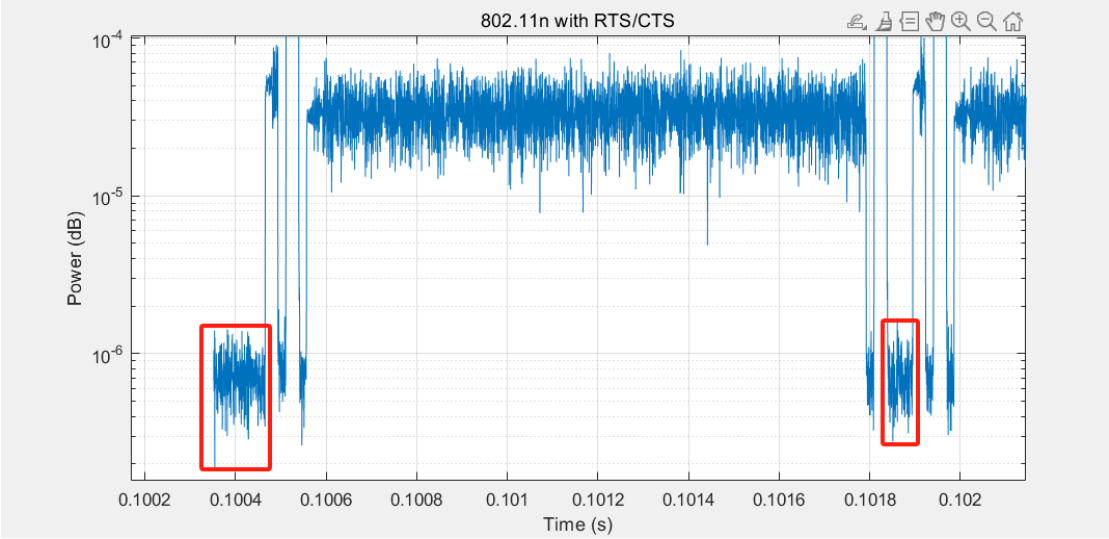


To indicate each period clearly, I plot another figure which include only two cycles:



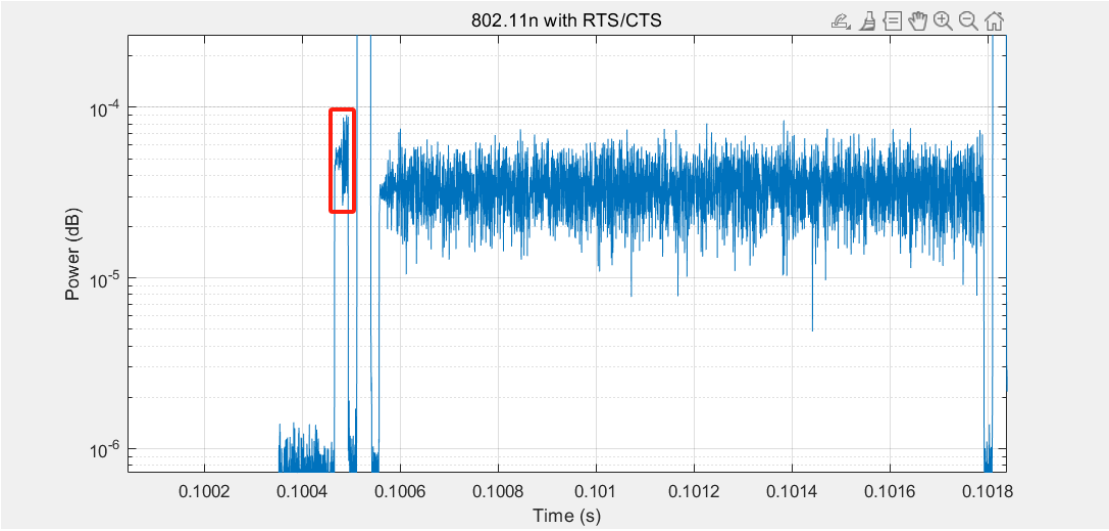
In the case of 802.11n with RTS/CTS, one cycle should be like DIFS → RTS → SIFS → CTS → SIFS → Dataframe → SIFS → ACK.

After zooming the figure, we could point that the DIFS should be the period marked as follows,



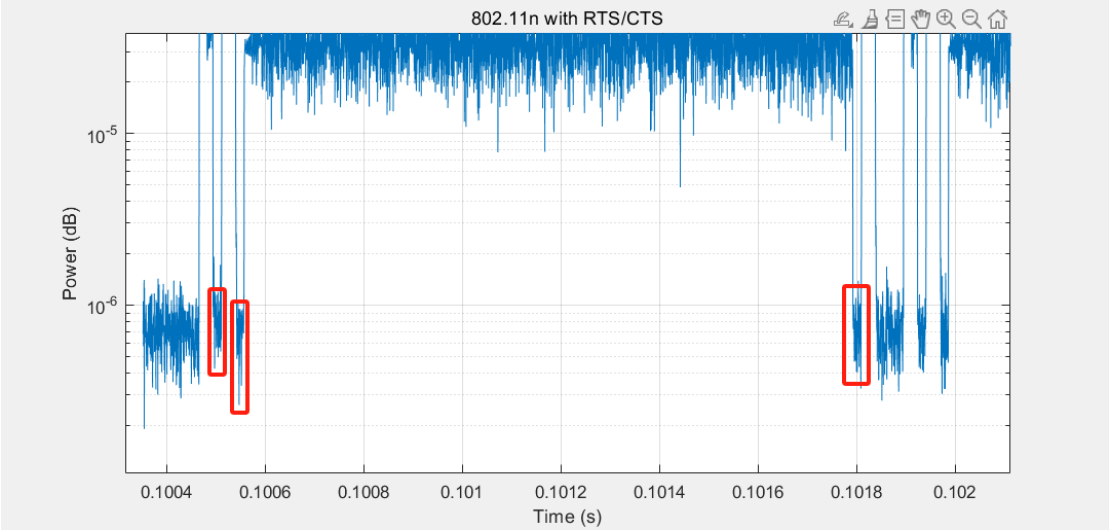
There are two marks in the figure, the first one is the DIFS of the first cycle, and the second one is the next cycle.

The RTS is the period marked as follows,



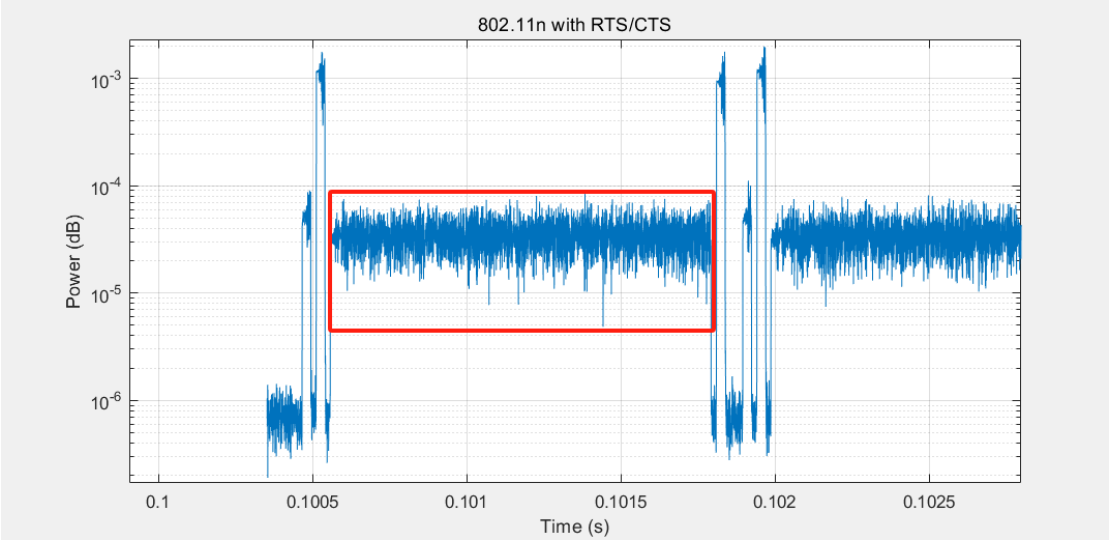
The shock marked in the figure is RTS, the RTS is the period next to the DIFS.

The SIFS is the period marked as follows,



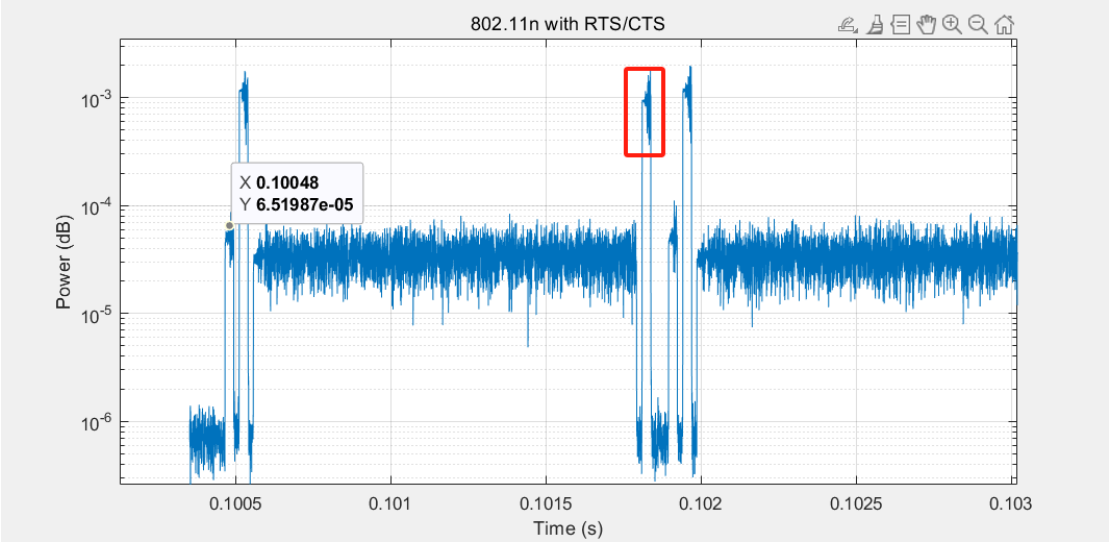
There are three SIFS period in one cycle, respectively after RTS, CTS and the dataframe.

The Dataframe is the period marked as follows,

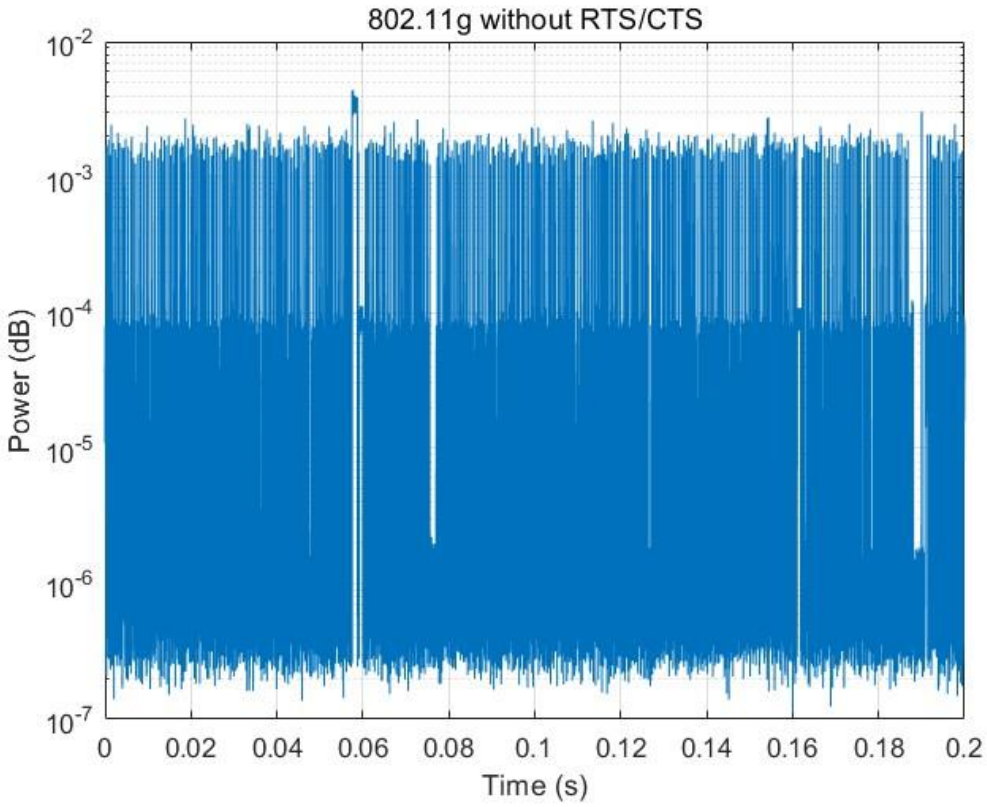


It is obvious that the dataframe is a stable and sustained period, after the dataframe, there is another SIFS as we said before.

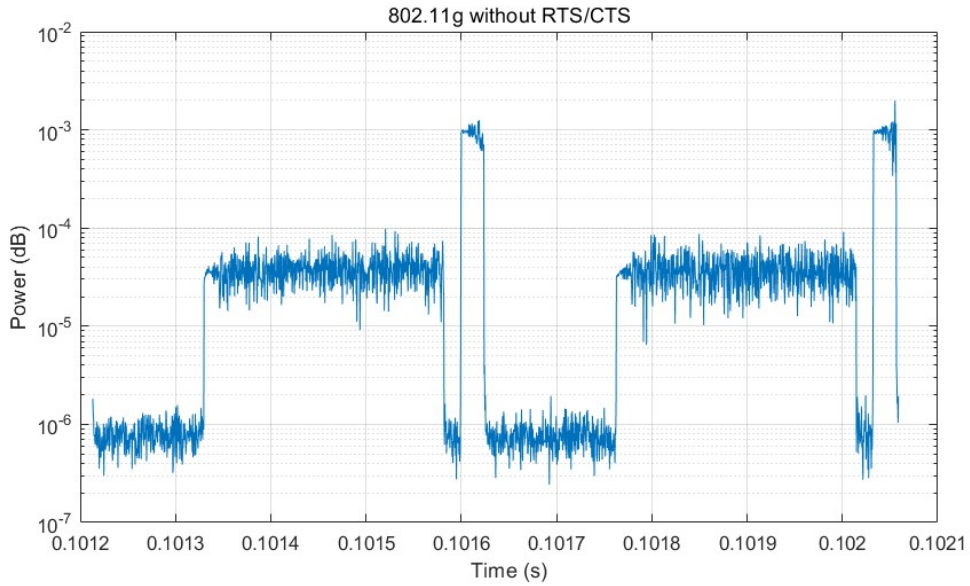
The Acknowledgement is marked as follows,



In the case of 802.11g without RTS/CTS, we could see the power over time plotting as follows:

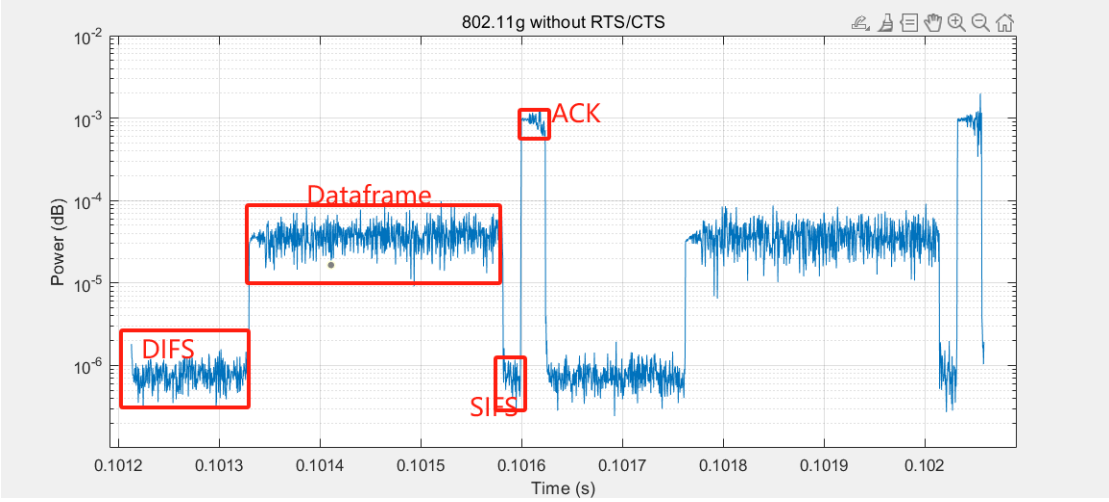


To indicate each period clearly, I plot another figure which include only two cycles:



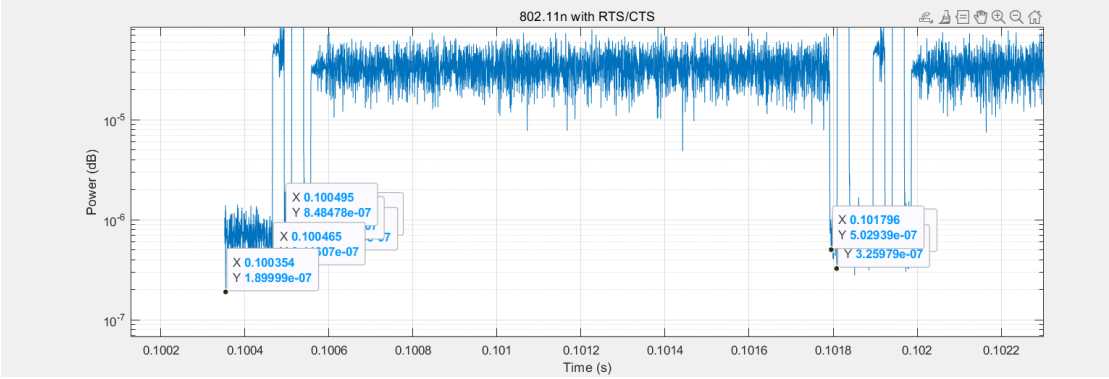
In the case of 802.11g without RTS/CTS, one cycle should be like DIFS → Dataframe → SIFS → ACK. Compare with the case of 802.11n with RTS/CTS, the RTS, CTS and the SIFS after them are not included.

After zooming the figure, we could point that the DIFS, Dataframe, SIFS and ACK marked as follows,



4.4.2 Calculate data frame, acknowledgement, RTS, CTS, SIFS and DIFS lengths.

In the case of 802.11n with RTS/CTS, we could mark the point at the beginning and the end of each period as follows,



Based on the value we get from the figure, we could get that in one cycle from 0.100354s to 0.101842s, it could be divided as follows,

DIFS: 0.100354s to 0.100465s, length: 0.100465-0.100354=0.000111s

RTS: 0.100465s to 0.100495s, length: 0.100495-0100465=0.00003s

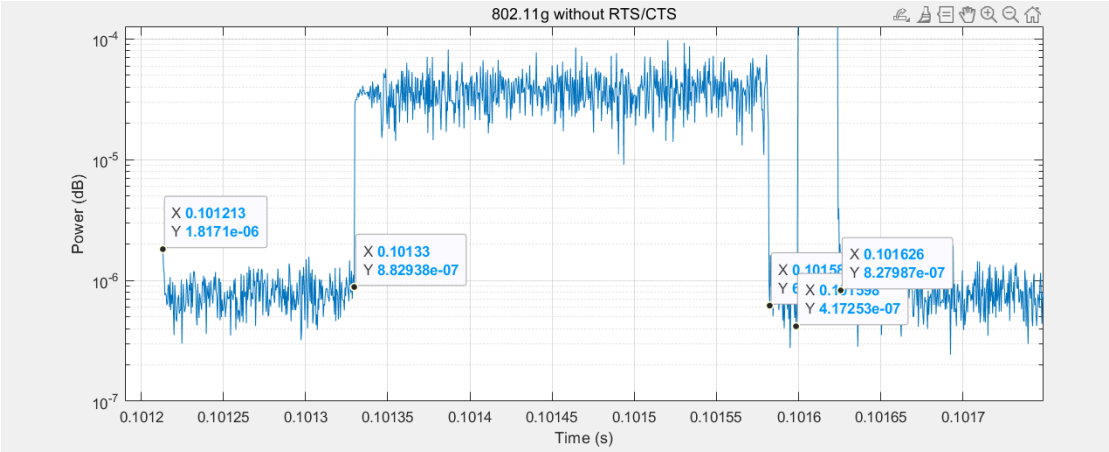
CTS: 0.10051s to 0.100542s, length: 0.100542-0.10051=0.000032s

Dataframe: 0.100556s to 0.101796s, length: 0.101796-0.100556=0.00124s

SIFS: 0.100495s to 0.10051s (between RTS and CTS), 0.100542s to 0.100556s (between CTS and Dataframe), 0.101796s to 0.101808s (between Dataframe and ACK) length: (0.10051-0.100495)+(0.100556-0.100542)+(0.101796-0.101808)=0.000017s

ACK: 0.101808s to 0.101842s, length: 0.101842-0.101808=0.000034s.

In the case of 802.11g without RTS/CTS, we could mark the point at the beginning and the end of each period as follows,



Based on the value we get from the figure, we could get that in one cycle from 0.101213s to 0.101626s, it could be divided as follows,

DIFS: 0.101213s to 0.10133s, length: 0.10133-0.101213=0.000117s

Dataframe: 0.10133s to 0.101582s, length: 0.101582-0.10133=0.000252s

SIFS:0.101582s to 0.101598s, length: 0.101598-0.101582=0.000016s

ACK: 0.101598s to 0.101626s, length: 0.101626-0.101598=0.000028s

4.4.3 How much overhead (in time) does RTS/CTS exchange cause? Compare with measurements in part 2.1 and 2.2.

Compare the two whole period, we could get that the RTS/CTS exchange cause 0.101842 - 0.100354 + 0.101626 - 0.101213 = 0.001901s (Include the SIFS after RTS and CTS)

4.4.4 Compare 802.11n and 802.11g efficiency.

The efficiency of 802.11n could be calculated by:

$$\frac{\text{Dataframe}}{\text{Dataframe+DIFS+RTS+CTS+SIFS+ACK}}$$

Substitute all values we could get that the efficiency is 0.833.

The efficiency of 802.11g could be calculated by:

$$\frac{\text{Dataframe}}{\text{Dataframe+DIFS+SIFS+ACK}}$$

Substitute all values we could get that the efficiency is 0.610.