COMP9336 – Mobile Data Networking

Lab 4 - WIFI API

T2 2022

Name: Yuhua Zhao – **ZID:** z5404443

Date: 23/06/2022

Disclaimer:

- Due to a hardware issue, my laptop cannot obtain the WIFI Frequency (2.4 & 5GHz) from the WIFI API. So, I use Channel to determine whether it's 2.5 or 5GHz. (More discussion is shown below)
- My laptop cannot obtain the signal strength directly, I used the Signal in Percentage to dBm function to do the conversion provided by Tutor Rui.
- Juypter Notebook will be submitted with all the code and output that I performed. (Feel free to have a look)

Task 1 – WIFI API Information extraction:

```
C:\Users\EricZhao>netsh wlan show networks mode=bssid

Interface name : Wi-Fi
There are 43 networks currently visible.

SSID 1 : Optus_A0D197_5GHz
    Network type            : Infrastructure
    Authentication          : WPA2-Personal
    Encryption              : CCMP
    BSSID 1                 : d0:6d:c9:a0:d1:9a
         Signal             : 10%
         Radio type         : 802.11ac
         Channel            : 157
         Basic rates (Mbps) : 6 12 24
         Other rates (Mbps) : 9 18 36 48 54

SSID 2 : VicFreeWiFi
    Network type            : Infrastructure
    Authentication          : Open
    Encryption              : None
    BSSID 1                 : b0:aa:77:95:8e:8e
         Signal             : 15%
         Radio type         : 802.11ac
         Channel            : 157
         Basic rates (Mbps) : 12 24
         Other rates (Mbps) : 18 36 48 54
```

This WIFI API can be extract the neighbour SSID information include:

- SSID name
- Authentication Method of Specific SSID
- Encryption method
- BSSID Information (Signal, Radio Type and so on)
  - Within the same network, if there are multiple Access Points exist, there will be multiple BSSID existing under the same SSID.

Note:

- This scan perform after the coding is completed (same location), some of the SSID may not be shown on the table that I created for task 2.

Task 2 – Counting Surrounding Devices and distance estimation:

Program design:

1) Run the following command to get the Neighbour SSID information.

   (Same as running “netsh wlan show networks mode=BSSID” on Window 10 OS)

```
# Run the command to search the SSID nearby
results = subprocess.check_output(["netsh", "wlan", "show", "network", "mode=BSSID"])
```

2) After retrieving the data, I split the String base on SSID. For example, if there are two SSID discovered, each SSID will be separated as String, basically listing the data structure with two strings.

```
def split_ssids(netsh_Result):
    netsh_decode = netsh_Result.decode('utf-8')
    SplitSSID = netsh_decode.split("\nSSID ")

    # Remove the First Array (Netsh Intro)
    SplitSSID.pop(0)

    # Splited (removed) SSID, need to add back
    for i in range(len(SplitSSID)):
        SplitSSID[i] = "SSID " + SplitSSID[i]

    return SplitSSID
```

3) After splitting the SSIDs, the string of each SSID will feed into the function of "get_SSID_Info()" to store the SSID, BSSID, Signal, Channel information, and so on according. In my case, I use Dictionary to store the SSID information such as SSID name, Authentication method. As mentioned above, if there are multiple APs for the same SSID, the BSSID will be under the SSID section, since I use List to store each BSSID with the corresponding signal, channel, Radio Type, and so on.

```python
def get_SSID_Info(ssid_array_splited):

    # Split the String base on ":" and clean all the spacing
    ssid_Split_Space = ssid_array_splited.split("\n")
    ssid_array = []
    for i in ssid_Split_Space:
        temp = i.split(" : ")
        if len(temp) == 2:
            temp[0] = temp[0].strip()
            temp[1] = temp[1].strip()
            ssid_array.append(temp)


    # Initial a Dictionary to store info
    dic = {}
    ssid_dic={}
    temp_arr = []
    bssid_arr = []

    # Loop each array to store the information accordingly
    for ac in range(len(ssid_array)):
        # Get SSID and store in dic
        if ssid_array[ac][0].find("SSID") != -1 & ssid_array[ac][0].find("BSSID"
            dic["SSID"] = ssid_array[ac][1]

        # Get SSID and store in dic
        elif ssid_array[ac][0].find("Network type") != -1:
            ssid_dic.update({"NetworkType": ssid_array[ac][1]})
            dic["SSID_Info"] = ssid_dic

        # Get SSID and store in dic
        elif ssid_array[ac][0].find("Authentication") != -1:
            ssid_dic.update({"Authentication": ssid_array[ac][1]})
            dic["SSID_Info"] = ssid_dic

        # Get SSID and store in dic
        elif ssid_array[ac][0].find("Encryption") != -1:
            ssid_dic.update({"Encryption": ssid_array[ac][1]})
            dic["SSID_Info"] = ssid_dic

        # Get BSSID info and store as Array
        elif ssid_array[ac][0].find("BSSID") == 0:
            if len(temp) > 0:
                bssid_arr.append(temp_arr)
                temp_arr = []
                temp_arr.append(ssid_array[ac][1])

        # when Counter reach to the end,
        elif ac == len(ssid_array) - 1:
            # Update array List
            temp_bssid = []
            bssid_arr.append(temp_arr)

            # Clean the List
            for l in bssid_arr:
                if len(l) != 0:
                    temp_bssid.append(l)

            # Write the List to Dictionary
            bssid_arr = temp_bssid
            ssid_dic.update({"BSSID_Info": bssid_arr})

        else:
            temp_arr.append(ssid_array[ac][1])

    print(bssid_arr)

    return dic
```

In each BSSID Array

Index 0: BSSID

Index 1: Signal

Index 2: Radio Type

Index 3: Channel

Index 4: Basic Rates (Mbps)

Index 5: Other Rates (Mbps)

4) A couple Function is created for different purposes:

```
lef get_frequency(Channel):
    if int(Channel) <= 14:
        ssid_frequency = "2.4 GHz"
    else:
        ssid_frequency = "5 GHz"

    return ssid_frequency
```

Use the Channel number to determine which frequency is used.

Note: The 2.4 and 5GHz channels are not overlapped according to the Australia Wireless Frequency usage standard.

```
def get_SignalStrength(rssi):
    if (float(rssi<=0)):
        dbm = 100
    elif (float(rssi)>100):
        dbm = -50
    else:
        dbm = float(rssi)/2-100
    return dbm
```

This algorithm is provided by Tutor (Rui) to convert Signal in Percentage to dBm.

```
def get_estDistance(Frequency, Channel, SignalStrength):
    # Info Link: https://en.wikipedia.org/wiki/List_of_WLAN_channels
    # Frequency Table 2.4Ghz & 5GHz
    FT_2_4 = [[1, 2412], [2, 2417], [3, 2422], [4, 2427], [5, 2432], [6, 2437],
    FT_5 = [[32, 5160], [34, 5170], [36, 5180], [38, 5190], [40, 5200],
            [42, 5210], [44, 5220], [46, 5230], [48, 5240], [50, 5250],
            [52, 5260], [54, 5270], [56, 5280], [58, 5290], [60, 5300],
            [62, 5310], [64, 5320], [68, 5340], [96, 5480], [100, 5500],
            [102, 5510], [104, 5520], [106, 5530], [108, 5540], [110, 5550],
            [112, 5560], [114, 5570], [116, 5580], [118, 5590], [120, 5600],
            [122, 5610], [124, 5620], [126, 5630], [128, 5640], [132, 5660],
            [134, 5670], [136, 5680], [138, 5690], [140, 5700], [142, 5710],
            [144, 5720], [149, 5745], [151, 5755], [153, 5765], [155, 5775],
            [157, 5785], [159, 5795], [161, 5805], [163, 5815], [165, 5825],
            [167, 5835], [169, 5845], [171, 5855], [173, 5865], [175, 5875],
            [177, 5885], [182, 5910], [183, 5915], [184, 5920], [187, 5935],
            [188, 5940], [189, 5945], [192, 5960], [196, 5980]]


    SignalStrength = abs(float(SignalStrength))
    Channel_Freq = 0

    if Frequency == "2.4 GHz":
        for i in FT_2_4:
            if i[0] == int(Channel):
                Channel_Freq = i[1]
                break
    else:
        for i in FT_5:
            if i[0] == int(Channel):
                Channel_Freq = i[1]
                break

    result = 10 ** ((27.55 - (20 * math.log10(Channel_Freq)) + SignalStrength)/2
    return result
```

The FT_2_4 and the FT_5 show the exact frequency usage for each channel for 2.4GHz and 5GHz.
(Link: https://en.wikipedia.org/wiki/List_of_WLAN_channels)

The estimated Distance calculation will be using "Free Space Loss Path equation"

5) Perform calculations and output as a Table.

```
table_Array = []
for i in range(len(overall_dic)):
    ssid = overall_dic[i]["SSID"]
    for j in range(len(overall_dic[i]["SSID_Info"]["BSSID_Info"])):
        BSSID = overall_dic[i]["SSID_Info"]["BSSID_Info"][j][0]
        Signal = overall_dic[i]["SSID_Info"]["BSSID_Info"][j][1]
        Channel = overall_dic[i]["SSID_Info"]["BSSID_Info"][j][3]
        ssid_frequency = get_frequency(Channel)
        ssid_SignalStrength = get_SignalStrength(float(Signal.replace("%","")))
        est_Distance = get_estDistance(ssid_frequency, Channel, ssid_SignalStren
        temp = [ssid, BSSID, ssid_frequency, Channel, ssid_SignalStrength, est_D
        table_Array.append(temp)

Table_head = ["SSID", "Frequency", "Channel", "Signal in %", "Signal Strength",
print(tabulate(table_Array, headers=Table_head, tablefmt='orgtbl'))
```

Yuhua Zhao
Z5404443

Result:

SSID	BSSID	Frequency	Channel	Signal Strength	Est. Distance (m)
VicFreeWiFi	b0:aa:77:95:8e:8e	5 GHz	157	-97.5	309.169
VicFreeWiFi	84:b8:02:f4:0f:ae	5 GHz	165	-82.5	54.6014
VicFreeWiFi	bc:16:f5:9e:53:9e	5 GHz	165	-95	230.252
Optus_A0D197_5GHz	d0:6d:c9:a0:d1:9a	5 GHz	157	-97.5	309.169
	8c:85:80:71:17:a5	2.4 GHz	12	-57.5	7.24988
	64:9a:12:21:9a:01	5 GHz	132	-93	188.228
	66:66:24:41:4f:97	5 GHz	40	-92.5	193.418
	9a:42:65:7d:ed:9a	5 GHz	36	-85	81.8786
	46:d4:53:5f:eb:7e	5 GHz	36	-92.5	194.165
	00:00:00:00:00:00	5 GHz	52	-93	202.542
	62:45:b8:10:d4:50	5 GHz	161	-50	1.29927
	9a:42:65:b8:51:22	5 GHz	161	-50	1.29927
	d2:6d:c9:cf:8c:74	5 GHz	149	-90	131.283
Sebel Guest	82:2a:a8:03:8f:44	2.4 GHz	11	-50	3.06346
Sebel Guest	82:2a:a8:03:8d:98	2.4 GHz	6	-50	3.09488
Sebel Guest	24:c9:a1:2d:3d:c8	2.4 GHz	3	-75	55.3765
Sebel Guest	82:2a:a8:04:8f:44	5 GHz	157	-80	41.2284
Sebel Guest	82:2a:a8:04:8d:98	5 GHz	44	-92.5	192.677
Sebel Guest	24:c9:a1:2d:05:08	2.4 GHz	13	-62.5	12.8662
Galaxy Z Fold2 5GCD9F	42:b6:78:4b:ed:58	2.4 GHz	11	-50	3.06346
Sebel Staff	80:2a:a8:03:8f:44	2.4 GHz	11	-50	3.06346
Sebel Staff	80:2a:a8:03:8d:98	2.4 GHz	6	-52.5	4.1271
Sebel Staff	24:c9:a1:6d:3d:c8	2.4 GHz	3	-87.5	233.521
Sebel Staff	80:2a:a8:04:8f:44	5 GHz	157	-80	41.2284
Sebel Staff	80:2a:a8:04:8d:98	5 GHz	44	-92.5	192.677
Optus_5FEB7A	44:d4:53:5f:eb:7c	2.4 GHz	11	-62.5	12.9185
DODO-E0A1	c8:94:bb:96:e0:a8	2.4 GHz	9	-60	9.72701
Optus_B8511E	98:42:65:b8:51:20	2.4 GHz	9	-65	17.2973
Optus_B8511E	98:42:65:b8:51:21	5 GHz	161	-87.5	97.4311
NQDCWIFI	a0:ab:1b:f4:fa:f2	2.4 GHz	7	-50	3.08855
Optus_7DED96	98:42:65:7d:ed:98	2.4 GHz	6	-55	5.50357
WePresent-Hampton	d8:61:62:54:d4:a8	2.4 GHz	4	-90	310.764
WiFi-399BCB	10:27:f5:39:9b:cb	2.4 GHz	3	-67.5	23.3521
WiFi-399BCB	10:27:f5:39:9b:cd	5 GHz	36	-95	258.923
WiFi-058A	00:31:92:1d:05:8a	2.4 GHz	3	-87.5	233.521
NetComm 5978	f8:ca:59:4c:b5:1e	2.4 GHz	1	-60	9.88832
Optus_414F93	64:66:24:41:4f:95	2.4 GHz	1	-62.5	13.1863
WiFi-HT635	6c:ff:ce:38:bd:47	5 GHz	144	-87.5	98.879
TelstraD7225C	d6:35:1d:d7:22:5c	5 GHz	132	-92.5	177.698
iiNet Customer	bc:16:f5:9e:53:9f	5 GHz	165	-97.5	307.046
iiNet Customer	b0:aa:77:95:8e:8f	5 GHz	157	-95	231.844
iiNet Customer	84:b8:02:f4:0f:af	5 GHz	165	-82.5	54.6014
Achelya-5G	2c:30:33:12:d8:da	5 GHz	157	-50	1.30376

(This Screenshot only shows part of the table, more info can be obtain on the Jupyter Notebook)

1) Same SSID shows in a row with a Different BSSIDs, which means that for the same SSID, there are a number of WIFI Access Points.
2) If the SSID is blank, it means that the SSID is hidden but can be able to be discovered by the WIFI API.
3) The Estimated Distance is calculated by using the "Free Space Loss Path equation". As we can see some of the estimated Distances are very large (over 300m). There are two reasons cause that. Firstly, it's because the obstacle blocks the signal transmission, or the Receiver only receives the reflected signal that causes the estimated Distance larger than the actual distance. The secondary is because the Signal Percentage converted to dBm is a rough calculation that may have deviation.

Yuhua Zhao
Z5404443