

17/03/2018

RSF – Les réseaux Wifi

Compte rendu

*« J'atteste que ce travail est original, qu'il indique de façon appropriée tous les emprunts,
et qu'il fait référence de façon appropriée à chaque source utilisée »*

Thomas Legris & Guilpain Léo
ESIR2 - IOT

Table des matières

Introduction.....	2
Travail préparatoire.....	2
Question n°1.....	2
Question n°2.....	2
Analyse de trafic.....	3
Question n°1.....	3
Question n°2.....	3
Question n°3.....	4
Question n°4.....	5
Question n°5.....	5
Allocation dynamique de canaux	6
Question n°1.....	6
Question n°2.....	7
Conclusion	9

Introduction

Le cours de réseau sans fil a pour objectif d'étudier les réseaux locaux sans fils (norme IEEE 802.11). Les séances de TP vont nous permettre de mettre en pratique l'apport théorique et donc d'étudier les différents échanges à travers la couche physique et liaison. Nous utiliserons pour cela plusieurs outils et programmes pour l'analyse des périphériques. Cela va nous permettre d'identifier les points d'accès les plus efficaces ou bien de regarder les différentes trames spécifiques de la couche liaison de données. En fin de TP, nous allons réfléchir à l'optimisation du réseau d'un point de vue « client » et d'un point de vue « point d'accès ».

Travail préparatoire

Tout d'abord, nous avons utilisé le logiciel Wireshark sur la machine Linux afin d'analyser les différentes trames qui passaient sur le réseau.

Question n°1

Pour identifier le périphérique réseau, il suffit de taper la commande « ifconfig » afin d'obtenir tous les composants réseau, on peut y trouver l'interface wifi wlan0.

On obtient le nom de notre carte :

```
0d:02:00 Ethernet controller: Qualcomm Atheros AR5413/AR5414 Wireless Network Adapter
```

Figure 1 : Nom de la carte

Question n°2

La commande « iwconfig » nous permet d'afficher tous les réseaux Wifi présent sur l'ordinateur. Elle nous permet également de faire passer la carte sans fil en mode monitor ce qui nous permet de lire ce qui se passe dans le réseau.

```
root@localhost:~# iwconfig
lo        no wireless extensions.

eth3      no wireless extensions.

wlan0     IEEE 802.11abg Mode:Monitor Frequency:2.412 GHz Tx-Power=30 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:off

eth2      no wireless extensions.

eth1      no wireless extensions.

eth4      no wireless extensions.

eth0      no wireless extensions.
```

Figure 2 : Commande « iwconfig »

Sur cette figure, nous pouvons observer que seul l'interface « wlan0 » est sans fil. Il est configuré à la fréquence 2.4 Ghz. Pour obtenir la liste des canaux que l'on peut utiliser, nous devons entrer la commande « iwlist » suivi du nom de notre interface.

Analyse de trafic

Question n°1

Les trames *beacons* sont générées par les différents points d'accès en mode broadcast. Ces trames sont envoyées périodiquement permettant d'indiquer aux stations qu'un point d'accès sans fils est présent.

1321	12.664435000	c8:69:cd:9c:ca:1a	Broadcast	802.11	156	Data, SN=431, FN=0, Flags=.pm...F.C
1322	12.664744000	80:a5:89:6c:4a:81	Broadcast	802.11	126	Data, SN=432, FN=0, Flags=.pm...F.C
1323	12.665086000	80:a5:89:6c:4a:81	Broadcast	802.11	126	Data, SN=433, FN=0, Flags=.pm...F.C
1324	12.665443000	e4:b3:18:da:a8:8b	Broadcast	802.11	174	Data, SN=434, FN=0, Flags=.pm...F.C
1325	12.665811000	e4:b3:18:da:a8:8b	Broadcast	802.11	174	Data, SN=435, FN=0, Flags=.pm...F.C
1326	12.666191000	e4:b3:18:da:a8:8b	Broadcast	802.11	174	Data, SN=436, FN=0, Flags=.pm...F.C
1327	12.674055000	00:f2:8b:7c:46:86	Broadcast	802.11	281	Beacon frame, SN=437, FN=0, Flags=.....C, BI=102, SSID=istic-public
1328	12.686491000	Cisco_e1:70:32	Broadcast	802.11	300	Beacon frame, SN=498, FN=0, Flags=.....C, BI=102, SSID=eduroam
1329	12.694350000	ArubaNet_a5:95:61	Broadcast	802.11	134	Beacon frame, SN=66, FN=0, Flags=.....C, BI=100, SSID=eduroam
1330	12.712899000	00:f2:8b:7c:46:83	Broadcast	802.11	303	Beacon frame, SN=438, FN=0, Flags=.....C, BI=102, SSID=univ-rennes1
1331	12.723305000	00:f2:8b:7c:46:85	Broadcast	802.11	296	Beacon frame, SN=439, FN=0, Flags=.....C, BI=102, SSID=istic
1332	12.735634000	Cisco_e1:70:33	Broadcast	802.11	300	Beacon frame, SN=500, FN=0, Flags=.....C, BI=102, SSID=univ-rennes1
1333	12.742080000	00:f2:8b:7c:46:80	Broadcast	802.11	276	Beacon frame, SN=440, FN=0, Flags=.....C, BI=102, SSID=eduspot
1334	12.744449000	ArubaNet_a5:98:82	Broadcast	802.11	134	Beacon frame, SN=41, FN=0, Flags=.....C, BI=100, SSID=eduroam
1335	12.747854000	Cisco_e1:70:35	Broadcast	802.11	293	Beacon frame, SN=501, FN=0, Flags=.....C, BI=102, SSID=istic
1336	12.766153000	Cisco_e1:70:30	Broadcast	802.11	273	Beacon frame, SN=502, FN=0, Flags=.....C, BI=102, SSID=eduspot
1337	12.768590000	00:f2:8b:7c:46:82	Broadcast	802.11	303	Beacon frame, SN=441, FN=0, Flags=.....C, BI=102, SSID=eduroam

Figure 3 : Capture Wireshark des trames beacon

Cette capture Wireshark montre que les différents points d'accès de l'université envoient régulièrement des trames *beacons* en broadcast. Cela leurs permet de se faire connaître aux membres de l'université. Elles sont reçues par notre station.

Question n°2

L'étude d'une trame nous montre les couches suivantes :

▶	Frame 1329: 134 bytes on wire (1072 bits), 134 bytes captured (1072 bits) on interface 0
▶	Radiotap Header v0, Length 26
▶	IEEE 802.11 Beacon frame, Flags:C
▶	IEEE 802.11 wireless LAN management frame

Figure 4 : Différentes couches d'une trame

▼	Frame 1329: 134 bytes on wire (1072 bits), 134 bytes captured (1072 bits) on interface 0
	Interface id: 0 (wlan0)
	Encapsulation type: IEEE 802.11 plus radiotap radio header (23)
	Arrival Time: Mar 12, 2018 08:17:29.194036000 CET
	[Time shift for this packet: 0.000000000 seconds]
	Epoch Time: 1520839049.194036000 seconds
	[Time delta from previous captured frame: 0.007859000 seconds]
	[Time delta from previous displayed frame: 0.007859000 seconds]
	[Time since reference or first frame: 12.694350000 seconds]
	Frame Number: 1329
	Frame Length: 134 bytes (1072 bits)
	Capture Length: 134 bytes (1072 bits)
	[Frame is marked: False]
	[Frame is ignored: False]
	[Protocols in frame: radiotap:wlan]
	[Number of per-protocol-data: 3]
	[IEEE 802.11 wireless LAN, key 1]
	[IEEE 802.11 wireless LAN, key 1]
	[IEEE 802.11 wireless LAN, key 1]

Figure 5 : Frame 1329

Le champ *Frame 1329* nous donne toutes les informations sur la trame. Cette couche contient notamment la date d'arrivée ainsi que la longueur de la trame.

```
IEEE 802.11 Beacon frame, Flags: .....C
Type/Subtype: Beacon frame (0x0008)
Frame Control Field: 0x8000
.000 0000 0000 0000 = Duration: 0 microseconds
Receiver address: Broadcast (ff:ff:ff:ff:ff:ff)
Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
Transmitter address: ArubaNet_a5:95:61 (00:1a:1e:a5:95:61)
Source address: ArubaNet_a5:95:61 (00:1a:1e:a5:95:61)
BSS Id: ArubaNet_a5:95:61 (00:1a:1e:a5:95:61)
Fragment number: 0
Sequence number: 66
Frame check sequence: 0x6f6cda55 [correct]
[Good: True]
[Bad: False]
```

Figure 6 : Couche 802.11

On trouve aussi la couche *802.11 Beacon frame* qui contient d'autres informations propres aux fonctions du Beacon. En effet, cette trame est transmise en broadcast par un point d'accès qui est identifié comme étant la source.

On voit apparaître :

- **Receiver Address** : l'adresse de la station à laquelle cette trame est envoyée (utile lorsque la trame doit transiter par des relais avant d'atteindre sa destination) ;
- **Transmitter Address** : l'adresse de la station expédiant la présente trame (utile lorsque cette station est une station relais).
- **Destination Address** : l'adresse du destinataire des données contenues dans le corps du paquet transmis ;
- **Source Address** : l'adresse de la source des données contenues dans le corps du paquet transmis ;

Question n°3

Dans les nouvelles mises à jour des cartes, il n'y a pas de mode de gestion d'énergie, nous ne pouvons pas identifier les éléments correspondants. On peut voir sur le screen ci-dessous que le mode gestion d'énergie n'est pas supporté.

```
...0 .... = PWR MGT: STA will stay up
```

Figure 7 : Mode économie d'énergie

Question n°4

```
592 2.307703000 00:f2:8b:7c:46:83 Broadcast 802.11 303 Beacon frame, SN=0, FN=0, Flags=.....C, BI=102, SSID=univ-rennes1
593 2.329709000 8c:85:90:26:69:a6 Broadcast 802.11 156 Data, SN=1510, FN=0, Flags=.pm...F.C
594 2.330479000 8c:85:90:26:69:a6 Broadcast 802.11 156 Data, SN=1511, FN=0, Flags=.pm...F.C
595 2.331246000 8c:85:90:26:69:a6 Broadcast 802.11 156 Data, SN=1512, FN=0, Flags=.pm...F.C
596 2.332444000 9c:b6:d0:20:02:d3 Broadcast 802.11 261 Data, SN=1513, FN=0, Flags=.pm...F.C
597 2.334745000 Cisco_e1:70:36 Broadcast 802.11 281 Beacon frame, SN=1514, FN=0, Flags=.....C, BI=102, SSID=istic-publ
598 2.371697000 Cisco_e1:70:33 Broadcast 802.11 303 Beacon frame, SN=1515, FN=0, Flags=.....C, BI=102, SSID=univ-renne
599 2.372940000 8c:85:90:26:69:a6 Broadcast 802.11 280 Data, SN=1516, FN=0, Flags=.pm...F.C
600 2.373450000 e8:2a:44:7d:45:bb Cisco_e1:70:32 802.11 56 QoS Null function (No data), SN=1010, FN=0, Flags=.....TC
601 2.373736000 e8:2a:44:7d:45:bb (RA) 802.11 40 Acknowledgement, Flags=.....C
602 2.374763000 IntelCor_2f:a5:f0 Broadcast 802.11 234 Data, SN=1517, FN=0, Flags=.pm...F.C
603 2.375517000 84:ef:18:db:b3:84 Broadcast 802.11 156 Data, SN=1518, FN=0, Flags=.pm...F.C
604 2.376169000 60:f8:1d:cb:90:66 Broadcast 802.11 129 Data, SN=1519, FN=0, Flags=.pm...F.C
605 2.380074000 Cisco_e1:70:32 (RA) 802.11 40 Acknowledgement, Flags=.....C
606 2.383957000 Cisco_e1:70:35 Broadcast 802.11 296 Beacon frame, SN=1520, FN=0, Flags=.....C, BI=102, SSID=istic
```

Figure 8 : Visualisation des différentes trames

Différentes trames sont analysées. Par exemple, il y a des trames « beacon », des trames « acknowledgment », des trames « Data ».

Question n°5

Lors de la connexion d'une station à un points d'accès, il y a plusieurs étapes permettant l'authentification.

Premièrement, il s'agit d'identifier notre téléphone. Nous avons utilisé le smartphone possédant l'adresse MAC Wifi 94 :65 :2d :7c :89 :71.

```
5896 39.570744000 94:65:2d:7c:89:71 Broadcast 802.11 160 Probe Request, SN=1129, FN=0, Flags=.....C, SSID=Broadcast
6778 44.010835000 94:65:2d:7c:89:71 Cisco_e1:70:30 802.11 163 Probe Request, SN=2105, FN=0, Flags=.....C, SSID=eduspot
6785 44.038573000 94:65:2d:7c:89:71 Cisco_e1:70:30 802.11 60 Authentication, SN=2106, FN=0, Flags=.....C
6786 44.039180000 94:65:2d:7c:89:71 Cisco_e1:70:30 802.11 60 Authentication, SN=2106, FN=0, Flags=....R...C
6787 44.039830000 94:65:2d:7c:89:71 Cisco_e1:70:30 802.11 60 Authentication, SN=2106, FN=0, Flags=.....C
6788 44.040414000 94:65:2d:7c:89:71 Cisco_e1:70:30 802.11 60 Authentication, SN=2106, FN=0, Flags=....R...C
6789 44.040995000 94:65:2d:7c:89:71 Cisco_e1:70:30 802.11 60 Authentication, SN=2106, FN=0, Flags=....R...C
6790 44.041602000 94:65:2d:7c:89:71 Cisco_e1:70:30 802.11 60 Authentication, SN=2106, FN=0, Flags=....R...C
6791 44.042202000 94:65:2d:7c:89:71 Cisco_e1:70:30 802.11 60 Authentication, SN=2106, FN=0, Flags=....R...C
6795 44.047079000 94:65:2d:7c:89:71 Cisco_e1:70:30 802.11 167 Association Request, SN=2107, FN=0, Flags=.....C, SSID=eduspot
1241 9.442454000 Htc_e0:4d:63 Broadcast 802.11 297 Probe Request, SN=3982, FN=0, Flags=...P....C, SSID=Broadcast
1242 9.443304000 ac:2b:6e:17:41:6b (TA) Cisco_e1:70:35 (RA) 802.11 46 Request-to-send, Flags=.....C
1243 9.444668000 Cisco_9c:f1:71 (RA) 802.11 40 Acknowledgement, Flags=...P....C
1244 9.452449000 Htc_e0:4d:63 Broadcast 802.11 297 Probe Request, SN=3983, FN=0, Flags=...P....C, SSID=Broadcast
1245 9.457327000 Cisco_e1:70:35 (RA) 802.11 40 Clear-to-send, Flags=.....C
```

Figure 9 : Résultat de l'authentification

La première étape est « Probe request », elle permet au téléphone de rechercher les différents réseaux aux alentours. Une fois le réseau trouvé (ici tentative de connexion au réseau universitaire « eduspot ») l'appareil fait une demande d'authentification, une fois cette phase terminée, le téléphone cherche alors à s'associer avec le point d'accès.

Allocation dynamique de canaux

Question n°1

Cette partie du TP vise à gérer les RF dynamiquement. Tout d'abord, nous avons créé un programme Python afin de permettre aux clients d'obtenir le meilleur point d'accès.

```
import os

def GetScan():
    f = os.popen('iwlist wlan0 scan')
    ESSIDs = []
    Channels = []
    Qualities = []
    Counter = 0
    for line in f:
        if line and line.strip():
            ch = line.replace(':', ' ').replace('/', ' ').replace('=', ' ').split()

            if ch[0] in ['ESSID']: #si le premier mot est ESSID
                ESSIDs.append(ch[1]) #on ajoute le deuxieme dans le tableau

            if ch[0] in ['Quality']: #si le premier mot est Quality
                Qualities.append(ch[1]) #on ajoute le deuxieme dans le tableau

            if ch[0] in ['Channel']: #si le premier mot est Channel
                Channels.append(ch[1]) #on ajoute le deuxieme dans le tableau

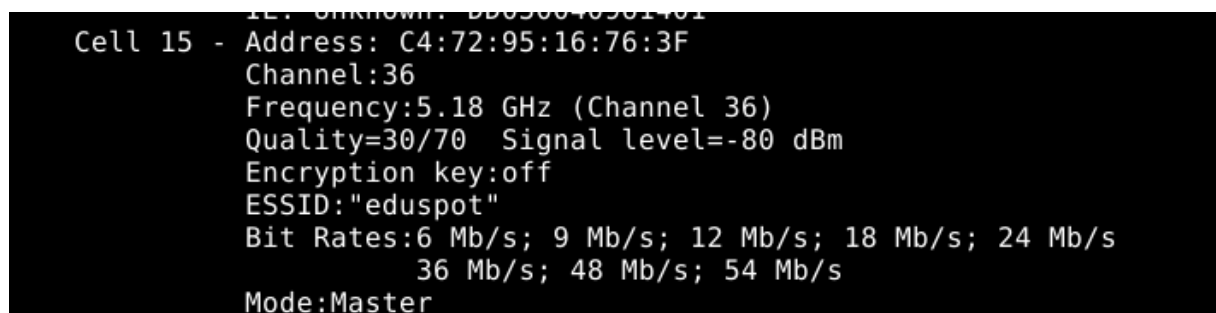
    maxqualite = max(Qualities)
    i=0
    longueur = len(Qualities);
    while i != longueur-1:
        if Qualities[i] == maxqualite:
            print "Le meilleur reseau est : ",ESSIDs[i]
            print "La meilleure qualite est : " , maxqualite # affiche la plus grande qualite
            return i
        else:
            i = i+1

    #print ESSIDs

GetScan()
```

Figure 10 : Programme permettant d'obtenir le nom du réseau

Lorsque l'on fait un scan sur wlan0, on obtient de nombreux éléments concernant les différents réseaux.



```
Cell 15 - Address: C4:72:95:16:76:3F
Channel:36
Frequency:5.18 GHz (Channel 36)
Quality=30/70 Signal level=-80 dBm
Encryption key:off
ESSID:"eduspot"
Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s
          36 Mb/s; 48 Mb/s; 54 Mb/s
Mode:Master
```

Figure 11 : iwlist scan wlan0

Comme on peut le voir, on a accès au canal, à la fréquence, à la qualité et au nom du réseau. Pour choisir le réseau le plus adéquat, il faut trouver celui dont la qualité est maximale. Le programme ci-dessus permet de renvoyer le nom du réseau qui a la qualité maximale ainsi que sa qualité.

Tout d'abord, on définit le fonction « getScan() » en initialisant 3 listes : *ESSIDs*, *Channels* et *Qualities*

Pour parcourir chaque réseau wifi public, on scan l'interface wlan0 et on parcourt celui-ci afin de lister chaque nom de réseau et chaque qualité. Déterminer l'élément le plus grand dans une liste est très facile, il suffit d'utiliser la fonction « max() » sur cette liste, déjà présente dans python, qui nous retourne la meilleure qualité. On refait alors une boucle afin d'associer le réseau qui correspond à cette qualité.

Voici le résultat de notre code.

```
oot@localhost:~/Bureau# python tp1.py
le meilleur reseau est : "univ-rennes1"
la meilleure qualite est : 40
```

Figure 12 : Résultat du code

Question n°2

```
1 import os
2
3 def GetCanal():
4     f = os.popen('iwlist wlan0 scan')
5     ESSIDs = []
6     Channels = []
7     Qualities = []
8
9
10    # On cree 1 tableau par canal avec comme valeur initiale 0
11    a1=[0]
12    a2=[0]
13    a3=[0]
14    a4=[0]
15    a5=[0]
16    a6=[0]
17    a7=[0]
18    a8=[0]
19    a9=[0]
20    a10=[0]
21    a11=[0]
22    a12=[0]
23    a13=[0]
24
25    i =0
26
27    for line in f:
28        if line and line.strip():
29            ch = line.replace(':', ' ').replace('/', ' ').replace('=', ' ').split()
30
31            if ch[0] in ['ESSID']: #si le premier terme est ESSID
32                ESSIDs.append(ch[1]) #on ajoute le deuxieme terme dans le tableau
33
34            if ch[0] in ['Quality']: #si le premier terme est Quality
35                Qualities.append(ch[1]) #on ajoute le deuxieme terme dans le tableau
36
37            if ch[0] in ['Channel']: #si le premier mot est Channel
38                Channels.append(ch[1]) #on ajoute le deuxieme terme dans le tableau
39
40
41    #On trouve la longueur du tableau Qualities
42    longueur = len(Qualities);
43
```



```

79     # On ajoute dans le tableau final, le max de chaque canal. La position dans le tableau correspond au numero du canal
80     tabfinal.append(max(a1))
81     tabfinal.append(max(a2))
82     tabfinal.append(max(a3))
83     tabfinal.append(max(a4))
84     tabfinal.append(max(a5))
85     tabfinal.append(max(a6))
86     tabfinal.append(max(a7))
87     tabfinal.append(max(a8))
88     tabfinal.append(max(a9))
89     tabfinal.append(max(a10))
90     tabfinal.append(max(a11))
91     tabfinal.append(max(a12))
92     tabfinal.append(max(a13))
93
94
95
96     longfinal = len(tabfinal)
97
98     # On remplace chaque 0 dans le tableau par un grand nombre afin de pouvoir retourner la qualite minimale
99     j = 0
100    while j != longfinal:
101        if tabfinal[j] == 0:
102            tabfinal[j] = '9999'
103        j = j+1
104
105
106    minqualite = min(tabfinal)
107    print "La moins bonne qualite est : ",minqualite
108
109    # On parcourt le tableau afin de trouver l'indice correspondant au canal qui a la minqualite
110    n = 0
111    while n != longfinal-1:
112        if tabfinal[n] == minqualite:
113            print "Le meilleur canal est : ",n+1
114            n = n+1
115        else:
116            n=n+1
117    |
118
119
120 GetCanal()

```

```

#On regarde la valeur de chaque canal, en fonction de la valeur, on ajoute la valeur de la qualite
while i != longueur-1:
    if Channels[i] == '1': # si la valeur du canal vaut 1
        a1.append(Qualities[i]) # on ajoute la qualite correspondante dans le tableau correspondant au canal 1
    if Channels[i] == '2':
        a2.append(Qualities[i])
    if Channels[i] == '3':
        a3.append(Qualities[i])
    if Channels[i] == '4':
        a4.append(Qualities[i])
    if Channels[i] == '5':
        a5.append(Qualities[i])
    if Channels[i] == '6':
        a6.append(Qualities[i])
    if Channels[i] == '7':
        a7.append(Qualities[i])
    if Channels[i] == '8':
        a8.append(Qualities[i])
    if Channels[i] == '9':
        a9.append(Qualities[i])
    if Channels[i] == '10':
        a10.append(Qualities[i])
    if Channels[i] == '11':
        a11.append(Qualities[i])
    if Channels[i] == '12':
        a12.append(Qualities[i])
    if Channels[i] == '13':
        a13.append(Qualities[i])
    i = i+1

# On cree un tableau final correspondant au meilleur qualite de chaque canal
tabfinal = []

```

Voici le code pour notre programme. Ce dernier permet de retourner le meilleur canal en fonction de la qualité.

```

root@i207m03:~/Bureau# python canal2.py
La moins bonne qualite est : 20
Le meilleur canal est : 6
root@i207m03:~/Bureau#
root@i207m03:~/Bureau# python canal2.py
La moins bonne qualite est : 21
Le meilleur canal est : 6
root@i207m03:~/Bureau# python canal2.py
La moins bonne qualite est : 30
Le meilleur canal est : 1

```

Figure 13 : Résultat du code

Nous avons également tenté de traiter les entrelacements. Nous avons réalisé ce code mais nous n'avons pas pu le tester.

```
#verification qu'un canal n'est pas impacte par les entrelacements
e = 0
k = 0
while e != longfinal:
    if tabfinal[e] == '9999':
        k = k+1

    if k==6:
        print "le meilleur canal est : ",e-3

    else:
        k=0

    e=e+1
```

Figure 14 : Test entrelacement

Comme on peut le voir dans ce code, nous avons tenté de regarder combien il y avait de canaux à la suite qui ne possédait pas de signal. En effet, pour ne pas subir des irradiations des canaux voisins, il faut se trouver à une distance minimale de 3 canaux de chaque côté. Ainsi, les meilleurs sont le 1, le 6 et le 11. Dans notre code, ces 3 canaux étaient tous le temps utilisés donc le meilleur canal est celui qui possède le plus bas rapport signal/bruit. Si un de ces canal n'est pas utilisé, alors ce sera lui le meilleur.

Conclusion

Le but de ce TP a été d'analyser et de comprendre la norme 802.11 (Wifi). Nous avons analysé les différentes trames échangées entre un client et un point d'accès public. De plus, on a observé les paramètres des points d'accès, la qualité des réseaux, les canaux utilisés et on a pu gérer dynamiquement les réseaux Wifi. On a par exemple défini le meilleur point d'accès en créant un programme Python.

Cette session de travaux pratiques a été bénéfique pour notre promotion. En effet, dans la filière IOT- objets connectés, l'étude des réseaux sans fils est impérative. La mise en pratique et le cours théorique nous permettent de vraiment comprendre le fonctionnement de cette technologie.