



Rapport TP2 WiFi

CSMA/CA



Table des matières

<i>Rapport TP2 WiFi CSMA/CA</i>	1
Introduction	3
1. Création d'une interface virtuelle	4
2. Configuration mode Ad-Hoc	5
3. Les Beacon Frame (trames balises)	7
4. Trames échangés au niveau LLC	9
5. La couche Physique	11
6. Technique CTS/RTS	12
a) Trame RTS	14
b) Trame CTS	14
7. WiFi sous Packet tracer	17
Conclusion.....	19



Introduction

Dans un réseau local Ethernet classique, la méthode d'accès utilisée par les machines est le CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*), pour lequel chaque machine est libre de communiquer à n'importe quel moment. Chaque machine envoyant un message vérifie qu'aucun autre message n'a été envoyé en même temps par une autre machine. Si c'est le cas, les deux machines patientent pendant un temps aléatoire avant de recommencer à émettre.

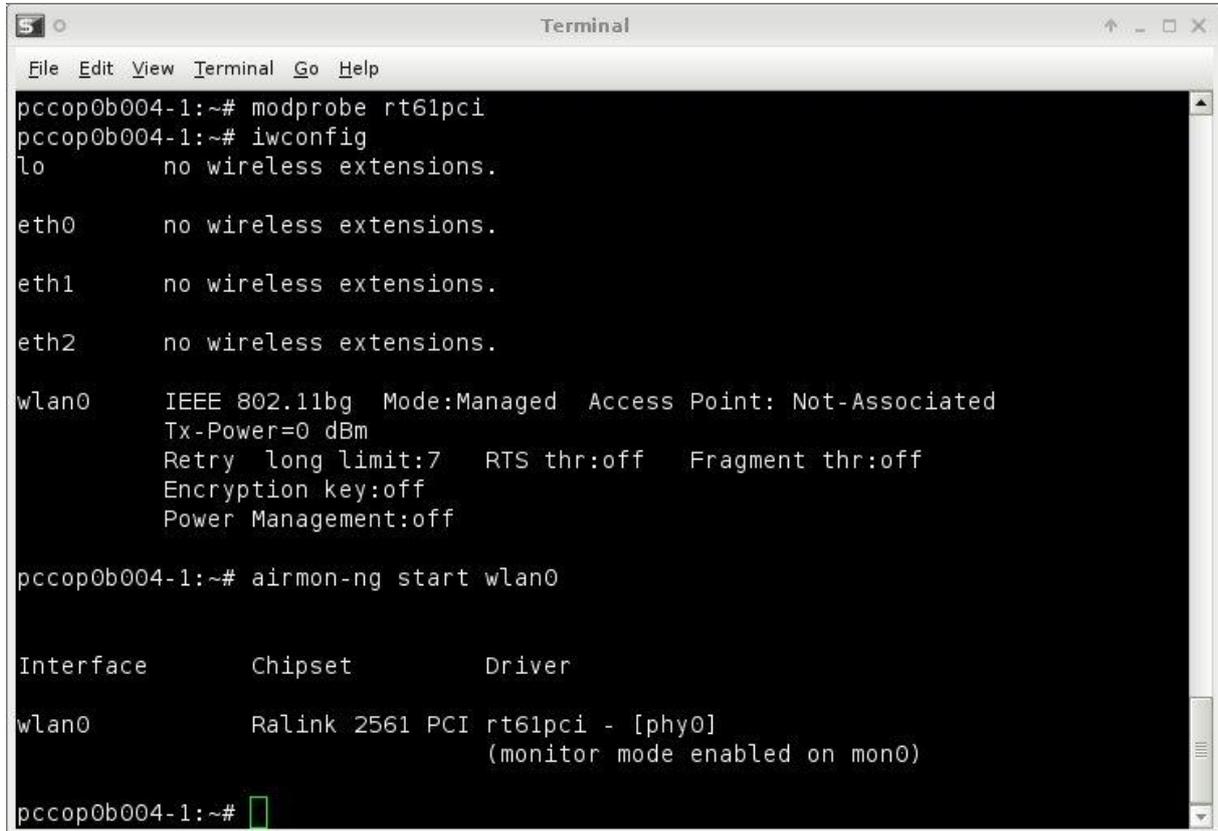
Dans un environnement sans fil ce procédé n'est pas possible dans la mesure où deux stations communiquant avec un récepteur ne s'entendent pas forcément mutuellement en raison de leur rayon de portée. Ainsi la norme 802.11 propose un protocole similaire appelé CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*).

Le protocole CSMA/CA utilise un mécanisme d'esquive de collision basé sur un principe d'accusé de réception réciproque entre l'émetteur et le récepteur.

Ce TP traite de l'étude précise de cette méthode. Pour cela, nous utilisons deux stations en mode ad-hoc, sachant que l'une des stations possède une interface Wifi virtuelle en mode *monitor* afin de sniffer le trafic.

1. Création d'une interface virtuelle

Pour commencer, nous allons créer une interface virtuelle *mon0* en mode monitor pour snifer le trafic.

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Go, Help). The terminal shows the following commands and output:

```
pccop0b004-1:~# modprobe rt61pci
pccop0b004-1:~# iwconfig
lo          no wireless extensions.

eth0       no wireless extensions.

eth1       no wireless extensions.

eth2       no wireless extensions.

wlan0      IEEE 802.11bg  Mode:Managed  Access Point: Not-Associated
           Tx-Power=0 dBm
           Retry  long limit:7   RTS thr:off   Fragment thr:off
           Encryption key:off
           Power Management:off

pccop0b004-1:~# airmon-ng start wlan0

Interface      Chipset      Driver
wlan0          Ralink 2561 PCI rt61pci - [phy0]
                (monitor mode enabled on mon0)

pccop0b004-1:~# █
```

Pour ce faire, il faut charger les modules de la carte réseau (Ralink 2561 dans ce cas) à l'aide de la commande `modprobe rt61pci`. Nous pouvons alors vérifier la présence de l'interface *wlan0* qui est active. La création l'interface virtuelle *mon0* s'opère avec la commande `airmon-ng start wlan0`. On peut alors remarquer que l'interface *wlan0* possède une interface virtuelle qui est activée.

27 mars 2011

On lance une capture wireshark sur l'interface *mon0*.

The screenshot shows the Wireshark interface with a list of captured frames. The first frame is selected, and its details are shown in the lower pane. The frame is an IEEE 802.11 Beacon frame with SSID 'unlv-sf-captif'.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	ArubaNet_6d:cd:50	Broadcast	IEEE 802	Beacon frame, SN=1211, FN=0, Flags=....., BI=100, SSID="unlv-sf-captif"
2	0.001217	ArubaNet_6d:cd:51	Broadcast	IEEE 802	Beacon frame, SN=1212, FN=0, Flags=....., BI=100, SSID="eduroam"
3	0.045013	00:25:d3:44:ba:32	ArubaNet_6f:30:90	IEEE 802	Null function (No data), SN=199, FN=0, Flags=....., T
4	0.058915	ArubaNet_6d:cd:00	Broadcast	IEEE 802	Beacon frame, SN=396, FN=0, Flags=....., BI=100, SSID="unlv-sf-captif"
5	0.060172	ArubaNet_6d:cd:01	Broadcast	IEEE 802	Beacon frame, SN=397, FN=0, Flags=....., BI=100, SSID="eduroam"
6	0.102257	ArubaNet_6d:cd:50	Broadcast	IEEE 802	Beacon frame, SN=1213, FN=0, Flags=....., BI=100, SSID="unlv-sf-captif"
7	0.103614	ArubaNet_6d:cd:51	Broadcast	IEEE 802	Beacon frame, SN=1214, FN=0, Flags=....., BI=100, SSID="eduroam"
8	0.104826	QconTech_5c:c4:43	IPv4mcast_00:00:fd	IEEE 802	Data, SN=1215, FN=0, Flags=p....F
9	0.161213	ArubaNet_6d:cd:00	Broadcast	IEEE 802	Beacon frame, SN=398, FN=0, Flags=....., BI=100, SSID="unlv-sf-captif"
10	0.162569	ArubaNet_6d:cd:01	Broadcast	IEEE 802	Beacon frame, SN=399, FN=0, Flags=....., BI=100, SSID="eduroam"
11	0.204654	ArubaNet_6d:cd:50	Broadcast	IEEE 802	Beacon frame, SN=1216, FN=0, Flags=....., BI=100, SSID="unlv-sf-captif"
12	0.206911	ArubaNet_6d:cd:51	Broadcast	IEEE 802	Beacon frame, SN=1217, FN=0, Flags=....., BI=100, SSID="eduroam"
13	0.205387	ArubaNet_6d:cd:01	Broadcast	IEEE 802	Beacon frame, SN=401, FN=0, Flags=....., BI=100, SSID="eduroam"
14	0.307053	ArubaNet_6d:cd:50	Broadcast	IEEE 802	Beacon frame, SN=1218, FN=0, Flags=....., BI=100, SSID="unlv-sf-captif"
15	0.308410	ArubaNet_6d:cd:51	Broadcast	IEEE 802	Beacon frame, SN=1219, FN=0, Flags=....., BI=100, SSID="eduroam"
16	0.357124	00:24:6c:80:65:62	Broadcast	IEEE 802	Beacon frame, SN=2836, FN=0, Flags=....., BI=100, SSID="eduroam"
17	0.409455	ArubaNet_6d:cd:50	Broadcast	IEEE 802	Beacon frame, SN=1220, FN=0, Flags=....., BI=100, SSID="unlv-sf-captif"
18	0.410807	ArubaNet_6d:cd:51	Broadcast	IEEE 802	Beacon frame, SN=1221, FN=0, Flags=....., BI=100, SSID="eduroam"
19	0.468406	ArubaNet_6d:cd:00	Broadcast	IEEE 802	Beacon frame, SN=404, FN=0, Flags=....., BI=100, SSID="unlv-sf-captif"
20	0.469760	ArubaNet_6d:cd:01	Broadcast	IEEE 802	Beacon frame, SN=405, FN=0, Flags=....., BI=100, SSID="eduroam"
21	0.511851	ArubaNet_6d:cd:50	Broadcast	IEEE 802	Beacon frame, SN=1222, FN=0, Flags=....., BI=100, SSID="unlv-sf-captif"
22	0.513206	ArubaNet_6d:cd:51	Broadcast	IEEE 802	Beacon frame, SN=1223, FN=0, Flags=....., BI=100, SSID="eduroam"
23	0.514498	00:22:fb:90:3b:6a	Broadcast	IEEE 802	Data, SN=1224, FN=0, Flags=p....F
24	0.614250	ArubaNet_6d:cd:50	Broadcast	IEEE 802	Beacon frame, SN=1225, FN=0, Flags=....., BI=100, SSID="unlv-sf-captif"
25	0.615603	ArubaNet_6d:cd:51	Broadcast	IEEE 802	Beacon frame, SN=1226, FN=0, Flags=....., BI=100, SSID="eduroam"
26	0.633721	00:24:6c:80:65:62	c4:17:fe:5e:18:f2	IEEE 802	Probe Response, SN=1715, FN=0, Flags=....., BI=100, SSID="eduroam"
27	0.673198	ArubaNet_6d:cd:00	Broadcast	IEEE 802	Beacon frame, SN=408, FN=0, Flags=....., BI=100, SSID="unlv-sf-captif"
28	0.674553	ArubaNet_6d:cd:01	Broadcast	IEEE 802	Beacon frame, SN=409, FN=0, Flags=....., BI=100, SSID="eduroam"
29	0.718007	ArubaNet_6d:cd:51	Broadcast	IEEE 802	Beacon frame, SN=1228, FN=0, Flags=....., BI=100, SSID="eduroam"

Frame 1 (116 bytes on wire (116 bytes captured))
 Arrival Time: Mar 22, 2011 08:56:04.241962000
 [Time delta from previous captured frame: 0.000000000 seconds]
 [Time delta from previous displayed frame: 0.000000000 seconds]
 [Time since reference or first frame: 0.000000000 seconds]
 Frame Number: 1
 Frame Length: 116 bytes
 Capture Length: 116 bytes
 [Frame is marked: False]
 [Protocols in frame: radiotap.wlan]
 ▶ Radiotap Header v0, Length 24
 ▶ IEEE 802.11 Beacon frame, Flags:

```

0000 00 00 18 00 2e 48 00 00 00 02 6c 09 a0 00 b8 01  ....H.....l.....
0010 00 00 00 00 00 00 00 80 00 00 ff ff ff ff  ....P.....PK
0020 ff ff 00 18 1e 6d cd 50 00 18 1e 6d cd 50 b0 4b  ....P.....PK
0030 81 01 7b 07 00 00 00 64 00 21 04 00 0e 75 6d  ....f.....d.....um
0040 8c 76 2d 73 66 2d 63 61 70 74 69 66 01 08 82 84  ....lv-sf-ca ptif.....

```

Frame (frame), 116 bytes | Packets: 268 Displayed: 268 Marked: 0 Dropped: 0 | Profile: Default

Nous captons toutes les trames provenant des réseaux WiFi avoisinant.

2. Configuration mode Ad-Hoc

Maintenant nous allons positionner l'interface principale *wlan0* en mode ad-hoc et monter l'interface.

Avant d'effectuer cette manipulation on fait tomber l'interface Ethernet par câble et on débranche le câble.

```
# ifconfig eth0 down
```

On configure la carte pour se connecter en ad-hoc.

```
# wlanconfig wlan0 destroy
# wlanconfig wlan0 create wlandev wifi0 wlanmode adhoc
```

27 mars 2011

On détruit la configuration actuelle pour se mettre en mode *ad-hoc*. On lui spécifie un nom de réseau et un *channel* (fréquence à utiliser).

```
# iwconfig wlan0 essid "alexisaurele"  
# iwconfig wlan0 channel 1
```

On vérifie ensuite que l'on utilise bien ce nom de réseau et la fréquence propre au *channel* que l'on a choisi.

```
# iwconfig  
lo          no wireless extensions.  
  
eth0       no wireless extensions.  
  
eth1       no wireless extensions.  
  
wifi0      no wireless extensions.  
  
wlan0      IEEE 802.11g  ESSID:"alexisaurele"  
           Mode:Ad-Hoc  Frequency:2.442 GHz  
           Bit Rate:0 kb/s  
           RTS thr:off  Fragment thr:off  
           Encryption key:off  
  
           Link Quality=0/94  Signal level=-93 dBm  Noise  
level=-93 dBm  
           Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid  
frag:0  
           Tx excessive retries:0  Invalid misc:0  Missed  
beacon:0  
  
# iwlist wlan0 freq  
wlan0      22 channels in total; available frequencies :  
           Channel 01 : 2.412 GHz  
           Channel 02 : 2.417 GHz  
           Channel 03 : 2.422 GHz  
           Channel 04 : 2.427 GHz  
           Channel 05 : 2.432 GHz  
           Channel 06 : 2.437 GHz  
           Channel 07 : 2.442 GHz  
           Channel 08 : 2.447 GHz  
           Channel 09 : 2.452 GHz  
           Channel 10 : 2.457 GHz  
           Channel 11 : 2.462 GHz  
           Current Frequency=2.442 GHz (Channel 1)
```

3. Les Beacon Frame (trames balises)

Nous analysons ensuite les trames (beacon frame) envoyées par cette station.

The screenshot shows the Wireshark interface with a list of captured packets. The filter is set to 'IEEE'. The packet list shows several beacon frames from ArubaNet_6d:cd:50 and acknowledgement frames from Azurewav_61:b0:7b. The detailed view of packet 5009 (Frame 5009) is expanded, showing the IEEE 802.11 Beacon frame structure. The frame control field is 0x0080 (Normal), and the flags field is 0x0. The destination address is Broadcast (ff:ff:ff:ff:ff:ff) and the source address is ArubaNet_6d:cd:50 (00:1a:1e:6d:cd:50). The BSS ID is also ArubaNet_6d:cd:50 (00:1a:1e:6d:cd:50). The fragment number is 0 and the sequence number is 2510. The packet bytes are displayed in hexadecimal and ASCII at the bottom.

No.	Time	Source	Destination	Protocol	Info
5006	31.556672	Azurewav_61:b0:7b	(RA)	IEEE 802	Clear-to-send, Flags=.....
5007	31.560716	Azurewav_61:b0:7b	(RA)	IEEE 802	Clear-to-send, Flags=.....
5008	31.561841	Azurewav_61:b0:7b	(RA)	IEEE 802	Acknowledgement, Flags=.....
5009	31.590179	ArubaNet_6d:cd:50	Broadcast	IEEE 802	Beacon frame, SN=2510, FN=0, Flags=....., BI=100, SSID="umlv-sf-captif"
5010	31.591538	ArubaNet_6d:cd:51	Broadcast	IEEE 802	Beacon frame, SN=2511, FN=0, Flags=....., BI=100, SSID="eduroam"
5011	31.602280	EdimaxTe_a8:5e:06	Broadcast	IEEE 802	Beacon frame, SN=1810, FN=0, Flags=....., BI=100, SSID="alexisaurele"
5012	31.645854	ArubaNet_6d:cd:00	Broadcast	IEEE 802	Beacon frame, SN=873, FN=0, Flags=....., BI=100, SSID="umlv-sf-captif"
5013	31.692579	ArubaNet_6d:cd:50	Broadcast	IEEE 802	Beacon frame, SN=2512, FN=0, Flags=....., BI=100, SSID="umlv-sf-captif"
5014	31.693938	ArubaNet_6d:cd:51	Broadcast	IEEE 802	Beacon frame, SN=2513, FN=0, Flags=....., BI=100, SSID="eduroam"
5015	31.747604	ArubaNet_6d:cd:00	Broadcast	IEEE 802	Beacon frame, SN=875, FN=0, Flags=....., BI=100, SSID="umlv-sf-captif"
5016	31.794979	ArubaNet_6d:cd:50	Broadcast	IEEE 802	Beacon frame, SN=2514, FN=0, Flags=....., BI=100, SSID="umlv-sf-captif"
5017	31.796335	ArubaNet_6d:cd:51	Broadcast	IEEE 802	Beacon frame, SN=2515, FN=0, Flags=....., BI=100, SSID="eduroam"
5018	31.799039	Azurewav_61:b0:7b	(RA)	IEEE 802	Acknowledgement, Flags=.....
5019	31.845891	00:24:6c:80:65:60	Broadcast	IEEE 802	Beacon frame, SN=3053, FN=0, Flags=....., BI=100, SSID="umlv-sf-captif"
5020	31.847251	00:24:6c:80:65:62	Broadcast	IEEE 802	Beacon frame, SN=3052, FN=0, Flags=....., BI=100, SSID="eduroam"
5021	31.849997	ArubaNet_6d:cd:00	Broadcast	IEEE 802	Beacon frame, SN=877, FN=0, Flags=....., BI=100, SSID="umlv-sf-captif"
5022	31.852540	ArubaNet_6d:cd:01	Broadcast	IEEE 802	Beacon frame, SN=878, FN=0, Flags=....., BI=100, SSID="eduroam"

Frame 5009 (116 bytes on wire, 116 bytes captured)
 Radiotap Header v0, Length 24
 IEEE 802.11 Beacon frame, Flags:

- Type/Subtype: Beacon frame (0x08)
- Frame Control: 0x0080 (Normal)
 - Version: 0
 - Type: Management frame (0)
 - Subtype: 8
 - Flags: 0x0
 - DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0 From DS: 0) (0x00)
 - 0... = More Fragments: This is the last fragment
 - 0... = Retry: Frame is not being retransmitted
 - ...0 = PWR MGT: STA will stay up
 - ..0. = More Data: No data buffered
 - .0. = Protected flag: Data is not protected
 - 0... = Order flag: Not strictly ordered
 - Duration: 0
 - Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
 - Source address: ArubaNet_6d:cd:50 (00:1a:1e:6d:cd:50)
 - BSS Id: ArubaNet_6d:cd:50 (00:1a:1e:6d:cd:50)
 - Fragment number: 0
 - Sequence number: 2510

IEEE 802.11 wireless LAN management frame

```

0000 00 00 18 00 2e 48 00 00 00 02 6c 09 a0 00 c0 01  ....H...l....
0010 00 00 00 00 00 00 00 80 00 00 00 ff ff ff ff  ....P...m.P...
0020 ff ff 00 1a 1e 6d cd 50 00 1a 1e 6d cd 50 e0 9c  ....P...d...um
0030 81 11 ce 1a 07 00 00 00 64 00 21 04 00 0e 75 6d  ....d...
0040 6c 76 2d 73 66 2d 63 61 70 74 69 66 01 08 82 84  lv-sf-ca ptif...
  
```

Invalid filter | Packets: 5034 Displayed: 5034 Marked: 0 Dropped: 23 | Profile: Default

Les *beacon frames* servent à annoncer la présence d'une station émettrice dans le réseau. Un routeur écoute les trames balise pour reconnaître les nouveaux périphériques sans fils afin de s'associer et d'obtenir une adresse IP par exemple. On peut remarquer cela en regardant l'adresse destination qui est une adresse de diffusion (broadcast).

Voici la liste des informations contenues dans les *beacon frames* :

FC (*Frame Control*, en français *contrôle de trame*) : ce champ de deux octets est constitué des informations suivantes :

- **Version de protocole** : ce champ de 2 bits permettra de prendre en compte les évolutions de version du standard 802.11. La valeur est égale à zéro pour la première version.
- **Type** et **Sous-type** : ces champs, respectivement de 2 et 4 bits, définissent le type et le sous-type des trames. Le type *gestion* correspond aux demandes d'association ainsi qu'aux messages d'annonce du point d'accès. Le type *contrôle* est utilisé pour l'accès au média afin de demander des autorisations pour émettre. Enfin le type *données* concerne les envois de données (la plus grande partie du trafic).
- **To DS** : ce bit vaut 1 lorsque la trame est destinée au système de distribution (*DS*), il vaut zéro dans les autres cas. Toute trame envoyée par une station à destination d'un point d'accès possède ainsi un champ *To DS* positionné à 1.
- **From DS** : ce bit vaut 1 lorsque la trame provient du système de distribution (*DS*), il vaut zéro dans les autres cas. Ainsi, lorsque les deux champs *To* et *From* sont positionnés à zéro il s'agit d'une communication directe entre deux stations (mode *ad hoc*).
- **More Fragments** (*fragments supplémentaires*) : permet d'indiquer (lorsqu'il vaut 1) qu'il reste des fragments à transmettre
- **Retry** : à 1 ce bit spécifie que le fragment en cours est une retransmission d'un fragment précédemment envoyé (et sûrement perdu)
- **Power Management** (*gestion d'énergie*) : indique, lorsqu'il est à 1, que la station ayant envoyé ce fragment entre en mode de gestion d'énergie
- **More Data** (*gestion d'énergie*) : ce bit, utilisé pour le mode de gestion d'énergie, est utilisé par le point d'accès pour spécifier à une station que des trames supplémentaires sont stockées en attente.
- **WEP** : ce bit indique que l'algorithme de chiffrement WEP a été utilisé pour chiffrer le corps de la trame.
- **Order** (*ordre*) : indique que la trame a été envoyée en utilisant la classe de service strictement ordonnée (*Strictly-Ordered service class*)

Durée / ID : Ce champ indique la durée d'utilisation du canal de transmission.

Champs adresses : une trame peut contenir jusqu'à 3 adresses en plus de l'adresse de 48 bits

Contrôle de séquence : ce champ permet de distinguer les divers fragments d'une même trame. Il est composé de deux sous-champs permettant de réordonner les fragments :

- Le *numéro de fragment*
- Le *numéro de séquence*

4. Trames échangés au niveau LLC

Nous allons ensuite placer une deuxième station en mode ad-hoc avec la précédente.

Pour ce faire, nous configurons les adresses IP des deux stations :

station1 : 192.168.1.2

station2 : 192.168.1.3

```
pccop0b004-1:~# ping -c 1 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=3.27 ms

--- 192.168.1.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.270/3.270/3.270/0.000 ms
```

```
pccop0b004-3:~# ping -c 1 192.168.1.3
PING 192.168.1.3 (192.168.1.3) 56(84) bytes of data.
64 bytes from 192.168.1.3: icmp_seq=1 ttl=64 time=5.90 ms

--- 192.168.1.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 5.905/5.905/5.905/0.000 ms
```

Le ping est un succès, les deux stations communiquent. L'option `-c` indique qu'un seul paquet doit être envoyé.

Voici ce que nous obtenons sur la capture wireshark pendant les échanges.

The screenshot shows the Wireshark interface with the following details:

No.	Time	Source	Destination	Protocol	Info
1	0.000000	EdimaxTe_a8:5e:02	Broadcast	IEEE 802	Beacon frame, SN=1794, FN=0, Flags=....., BI=100, SSID="alexisaurele"
2	0.001181	EdimaxTe_a8:5e:14	Broadcast	IEEE 802	Beacon frame, SN=1808, FN=0, Flags=....., BI=100, SSID="alexisaurele"
3	0.029272	192.168.1.2	192.168.1.3	ICMP	Echo (ping) request
4	0.029299	192.168.1.3	192.168.1.2	ICMP	Echo (ping) reply
5	0.030818	EdimaxTe_a8:5e:04	(RA)	IEEE 802	Acknowledgement, Flags=.....
6	0.072488	ArubaNet_6d:cd:50	Broadcast	IEEE 802	Beacon frame, SN=3356, FN=0, Flags=....., BI=100, SSID="umlv-sf-captif"
7	0.073844	ArubaNet_6d:cd:51	Broadcast	IEEE 802	Beacon frame, SN=3357, FN=0, Flags=....., BI=100, SSID="eduroam"
8	0.077289	ArubaNet_6d:cd:50	(RA)	IEEE 802	Clear-to-send, Flags=.....

Frame 4 details:

- Frame 3 (140 bytes on wire, 140 bytes captured)
- Radiotap Header v0, Length 24
- IEEE 802.11 Data, Flags:
- Logical-Link Control
 - DSAP: SNAP (0xaa)
 - IG Bit: Individual
 - SSAP: SNAP (0xaa)
 - CR Bit: Command
 - Control field: U, func=UI (0x03)
 - 000.00... = Command: Unnumbered Information (0x00)
 -11 = Frame type: Unnumbered frame (0x03)
 - Organization Code: Encapsulated Ethernet (0x000000)
 - Type: IP (0x0800)
 - Internet Protocol, Src: 192.168.1.2 (192.168.1.2), Dst: 192.168.1.3 (192.168.1.3)
 - Internet Control Message Protocol

Packet bytes:

```

0030 aa aa 03 00 00 00 08 00 45 00 00 54 00 00 40 00  ... .. E..T..e.
0040 40 01 b7 52 c0 a8 01 02 c0 a8 01 03 08 00 d8 b6  @..S.....
0050 fc 09 00 26 5c 61 98 4d 4f 67 04 00 08 09 0a 0b  ...&a.M 0g.....
0060 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b  .....
0070 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b  .... !* $%&'()*+
  
```

Les trames au niveau LLC (Logical Link Control) comportent les informations suivantes :

- DSAP (Destination Service Access Point) : adresse destination.
- IG Bit : adresse individuelle ou adresse de groupe.
- SSAP (Service Source Access Point) : adresse source.
- CR Bit : type de LLC, contrôle avec/sans connexion avec/sans acquittement.

5. La couche Physique

Déterminez la valeur de débit à partir de l'entête de la couche physique.

1	0.000000	EdimaxTe_a8:5e:02	Broadcast	IEEE 802 Beacon frame, SN=1794, FN=0, Flags=....., BI=100, SSID="alexisaurele"
2	0.001181	EdimaxTe_a8:5e:14	Broadcast	IEEE 802 Beacon frame, SN=1808, FN=0, Flags=....., BI=100, SSID="alexisaurele"
3	0.029272	192.168.1.2	192.168.1.3	ICMP Echo (ping) request
4	0.029299	192.168.1.3	192.168.1.2	ICMP Echo (ping) reply
5	0.030818	EdimaxTe_a8:5e:04	(RA IEEE 802 Acknowledgement, Flags=.....	
6	0.072488	ArubaNet_6d:cd:50	Broadcast	IEEE 802 Beacon frame, SN=3356, FN=0, Flags=....., BI=100, SSID="unlv-sf-captif"
7	0.073844	ArubaNet_6d:cd:51	Broadcast	IEEE 802 Beacon frame, SN=3357, FN=0, Flags=....., BI=100, SSID="eduroam"
8	0.077289	ArubaNet_6d:cd:50	(RA IEEE 802 Clear-to-send, Flags=.....	


```

▶ Frame 3 (140 bytes on wire, 140 bytes captured)
  ▾ Radiotap Header v0, Length 24
    Header revision: 0
    Header pad: 0
    Header length: 24
    ▶ Present flags: 0x0000482e
    ▶ Flags: 0x00
      Data Rate: 24.0 Mb/s
      Channel frequency: 2412 [BG 1]
      ▶ Channel type: 802.11g (pure-g) (0x00c0)
      SSI Signal: -34 dBm
      Antenna: 1
      802.11 FCS: 0x00000000 [incorrect, should be 0xd72cb171]
    ▶ IEEE 802.11 Data, Flags: .....
    ▶ Logical-Link Control
    ▶ Internet Protocol, Src: 192.168.1.2 (192.168.1.2), Dst: 192.168.1.3 (192.168.1.3)
    ▶ Internet Control Message Protocol
  
```

On peut voir que le débit est de 24 Mbits/s à partir de l'entête de la couche physique dans le champ *Date rate*.

Quelle est la valeur du champ *Retry* ?

```

  ▾ Flags: 0x0
    DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0 From DS: 0) (0x00)
    .... 0.. = More Fragments: This is the last fragment
    .... 0.. = Retry: Frame is not being retransmitted
    ...0 .... = PWR MGT: STA will stay up
    ..0. .... = More Data: No data buffered
    .0.. .... = Protected flag: Data is not protected
    0... .... = Order flag: Not strictly ordered
    Duration: 0
  
```

La valeur du champ *Retry* est 0. Ce qui signifie que si un fragment se perd lors d'une transmission, ce dernier ne sera pas retransmit.

Est-ce que la trame est fragmentée au niveau de la couche 2 ? Quelle est la valeur du champ *Duration* ?

```

  Fragment number: 0
  Sequence number: 2510
  
```

La trame peut être fragmentée au niveau de la couche 2. Dans ce cas la trame n'a pas été fragmentée car le flag MF (*More Fragment*) est à 0, le numéro de fragment (*Fragment number*) est à 0 et le champ *Duration*, qui indique la durée d'utilisation du canal de transmission est également à 0.

6. Technique CTS/RTS

Le principe est simple, la station voulant émettre écoute le réseau. Si le réseau est encombré, la transmission est différée. Dans le cas contraire, si le média est libre pendant un temps donné (appelé *DIFS* pour *Distributed Inter Frame Space*), alors la station peut émettre. La station transmet un message appelé *Ready To Send* (noté *RTS* signifiant *prêt à émettre*) contenant des informations sur le volume des données qu'elle souhaite émettre et sa vitesse de transmission. Le récepteur (généralement un point d'accès) répond un *Clear To Send* (*CTS*, signifiant *Le champ est libre pour émettre*), puis la station commence l'émission des données.

A réception de toutes les données émises par la station, le récepteur envoie un accusé de réception (*ACK*). Toutes les stations avoisinantes patientent alors pendant un temps qu'elle considère être celui nécessaire à la transmission du volume d'information à émettre à la vitesse annoncée.

La technique RTS/CTS est-elle activée par défaut sur les cartes ? Expliquez.

Le RTS/CTS n'est pas activé par défaut, car ces trames sont envoyées à bas débit (1Mbits/s) pour atteindre les périphériques les plus éloignées, or le temps de transmission de ces trames fait baisser le débit utile.

Nous positionnons le RTS à 500.

```
pccop0b004-1:~# iwconfig wlan0 rts 500
pccop0b004-1:~# iwconfig wlan0
wlan0 IEEE 802.11bg ESSID:"alexisaurele"
      Mode:Ad-Hoc Frequency:2.412 GHz Cell: D2:FF:9F:62:AB:01
      Tx-Power=4 dBm
      Retry long limit:7 RTS thr=500 B Fragment thr:off
      Encryption key:off
      Power Management:off
```

Nous effectuons un ping de 500 octets.

```
pccop0b004-1:~# ping -s 500 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 500(528) bytes of data.
508 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=3.90 ms
508 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=2.88 ms
508 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=2.98 ms
508 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=2.15 ms
^C
--- 192.168.1.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 2.156/2.982/3.909/0.626 ms
pccop0b004-1:~#
```

27 mars 2011

```
▶ Frame 3562 (573 bytes on wire, 573 bytes captured)
▶ Radiotap Header v0, Length 13
▼ IEEE 802.11 Data, Flags: .....
  Type/Subtype: Data (0x20)
  ▼ Frame Control: 0x0008 (Normal)
    Version: 0
    Type: Data frame (2)
    Subtype: 0
    ▼ Flags: 0x0
      DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0 From DS: 0) (0x00)
      ....0.. = More Fragments: This is the last fragment
      ....0... = Retry: Frame is not being retransmitted
      ...0 .... = PWR MGT: STA will stay up
      ..0. .... = More Data: No data buffered
      .0.. .... = Protected flag: Data is not protected
      0... .... = Order flag: Not strictly ordered
    Duration: 172
```

Le champ *Duration* est à 172. Ainsi la durée d'utilisation du canal de transmission est plus longue.

Maintenant nous réalisons un ping de 5Mo.

```
pccop0b004-1:~# ping -s 50000 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 50000(50028) bytes of data.
50008 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=117 ms
50008 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=139 ms
50008 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=134 ms
50008 bytes from 192.168.1.2: icmp_seq=5 ttl=64 time=123 ms
^C
--- 192.168.1.2 ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time 4013ms
rtt min/avg/max/mdev = 117.317/128.966/139.705/8.853 ms
pccop0b004-1:~# █
```

La trame est alors fragmentée, et *Duration* passe augmente de manière significative. Le canal de transmission accorde plus de temps pour échanger des données fragmentées.

Les trames RTS et CTS sont utilisées pour la réservation virtuelle des ressources dans le cadre de la procédure d'accès au support physique. Cette technique permet de diminuer les collisions lors des transmissions.

L'entête MAC comporte quelques différences suivant qu'il s'agisse de trames RTS, CTS ou ACK.

a) Trame RTS

```
▼ IEEE 802.11 Request-to-send, Flags: .....
  Type/Subtype: Request-to-send (0x1b)
  ▼ Frame Control: 0x00B4 (Normal)
    Version: 0
    Type: Control frame (1)
    Subtype: 11
    ▶ Flags: 0x0
    Duration: 990
    Receiver address: EdimaxTe_a8:5e:04 (00:0e:2e:a8:5e:04)
    Transmitter address: EdimaxTe_a8:5e:14 (00:0e:2e:a8:5e:14)
```

L'entête de la trame RTS comprend les champs suivant :

- Frame Control : analogue au champ de la trame de données MAC.
- Duration : Durée à réserver.
- RA : Adresse de la station réceptrice.
- TA : Adresse de la station émettrice.

b) Trame CTS

```
▼ IEEE 802.11 Clear-to-send, Flags: .....
  Type/Subtype: Clear-to-send (0x1c)
  ▼ Frame Control: 0x00C4 (Normal)
    Version: 0
    Type: Control frame (1)
    Subtype: 12
    ▶ Flags: 0x0
    Duration: 676
    Receiver address: EdimaxTe_a8:5e:04 (00:0e:2e:a8:5e:04)
```

L'entête de la trame CTS comprend les mêmes champs que celui de RTS, hormis le champ TA. Le champ RA étant recopié à partir du champ TA de la trame RTS reçue.

On peut remarquer que l'adresse MAC source n'est pas présente, en effet la station qui envoie le CTS pour dire « j'écoute une station », peut être n'importe quelle station. On en vient donc au point faible de cette technique, imaginons qu'une machine envoie des CTS de manière rapide et régulière pour prendre toute la bande passante du réseau. Les périphériques sans fils deviennent alors vulnérables aux attaques et les hackers aiment cette technique de « CTS Flood » (inondation de CTS) car l'adresse MAC source n'est pas transmise. C'est une attaque par *déni de service* qui consiste à rendre indisponible un service, dans ce cas le réseau devient saturé.

27 mars 2011

En cas de fragmentation, un RTS/CTS est envoyé avant chaque fragment ? Prouvez ça ou le contraire via une analyse par wireshark.

3063	12.564386	EdimaxTe_a8:5e:14	(TA EdimaxTe_a8:5e:04	(RA) IEEE 802	Request-to-send, Flags=.....
3064	12.565295	192.168.1.2	192.168.1.3	IP	Fragmented IP protocol (proto=ICMP 0x01, off=29600)
3065	12.565668	EdimaxTe_a8:5e:14	(TA EdimaxTe_a8:5e:04	(RA) IEEE 802	Request-to-send, Flags=.....
3066	12.566843	192.168.1.2	192.168.1.3	IP	Fragmented IP protocol (proto=ICMP 0x01, off=31080)
3067	12.567217	EdimaxTe_a8:5e:14	(TA EdimaxTe_a8:5e:04	(RA) IEEE 802	Request-to-send, Flags=.....
3068	12.570069		00:25:d3:c2:ef:81	(RA) IEEE 802	Clear-to-send, Flags=.....
3069	12.571846	192.168.1.2	192.168.1.3	IP	Fragmented IP protocol (proto=ICMP 0x01, off=32560)
3070	12.572220	EdimaxTe_a8:5e:14	(TA EdimaxTe_a8:5e:04	(RA) IEEE 802	Request-to-send, Flags=.....
3071	12.573108	192.168.1.2	192.168.1.3	IP	Fragmented IP protocol (proto=ICMP 0x01, off=34040)
3072	12.573480	EdimaxTe_a8:5e:14	(TA EdimaxTe_a8:5e:04	(RA) IEEE 802	Request-to-send, Flags=.....
3073	12.574450	192.168.1.2	192.168.1.3	IP	Fragmented IP protocol (proto=ICMP 0x01, off=35520)
3074	12.574823	EdimaxTe_a8:5e:14	(TA EdimaxTe_a8:5e:04	(RA) IEEE 802	Request-to-send, Flags=.....
3075	12.575673	192.168.1.2	192.168.1.3	IP	Fragmented IP protocol (proto=ICMP 0x01, off=37000)
3076	12.576045	EdimaxTe_a8:5e:14	(TA EdimaxTe_a8:5e:04	(RA) IEEE 802	Request-to-send, Flags=.....
3077	12.576814	192.168.1.2	192.168.1.3	IP	Fragmented IP protocol (proto=ICMP 0x01, off=38480)
3078	12.577188	EdimaxTe_a8:5e:14	(TA EdimaxTe_a8:5e:04	(RA) IEEE 802	Request-to-send, Flags=.....
3079	12.578115	192.168.1.2	192.168.1.3	IP	Fragmented IP protocol (proto=ICMP 0x01, off=39960)
3080	12.578489	EdimaxTe_a8:5e:14	(TA EdimaxTe_a8:5e:04	(RA) IEEE 802	Request-to-send, Flags=.....
3081	12.579277	192.168.1.2	192.168.1.3	IP	Fragmented IP protocol (proto=ICMP 0x01, off=41440)
3082	12.579650	EdimaxTe_a8:5e:14	(TA EdimaxTe_a8:5e:04	(RA) IEEE 802	Request-to-send, Flags=.....
3083	12.580460	192.168.1.2	192.168.1.3	IP	Fragmented IP protocol (proto=ICMP 0x01, off=42920)
3084	12.580833	EdimaxTe_a8:5e:14	(TA EdimaxTe_a8:5e:04	(RA) IEEE 802	Request-to-send, Flags=.....
3085	12.581803	192.168.1.2	192.168.1.3	IP	Fragmented IP protocol (proto=ICMP 0x01, off=44400)
3086	12.582174	EdimaxTe_a8:5e:14	(TA EdimaxTe_a8:5e:04	(RA) IEEE 802	Request-to-send, Flags=.....
3087	12.583254	192.168.1.2	192.168.1.3	IP	Fragmented IP protocol (proto=ICMP 0x01, off=45880)
3088	12.583628	EdimaxTe_a8:5e:14	(TA EdimaxTe_a8:5e:04	(RA) IEEE 802	Request-to-send, Flags=.....
3089	12.584497	192.168.1.2	192.168.1.3	IP	Fragmented IP protocol (proto=ICMP 0x01, off=47360)
3090	12.584871	EdimaxTe_a8:5e:14	(TA EdimaxTe_a8:5e:04	(RA) IEEE 802	Request-to-send, Flags=.....
3091	12.585537	192.168.1.2	192.168.1.3	ICMP	Echo (ping) reply
3092	12.590944		00:25:d3:c2:ef:81	(RA) IEEE 802	Clear-to-send, Flags=.....

D'après la capture wireshark ci-dessus, il s'avère qu'un RTS est envoyé avant chaque fragment justement pour éviter les collisions de transmissions. Cependant les trames CTS sont envoyées moins souvent mais de manière périodique, probablement proportionnelle au champ *Duration*.

En changeant le débit de la carte, montrez que ce débit est transmis au niveau de l'entête de la couche physique.

Nous changeons le débit de l'interface wlan0 à 11 Mb/s.

```
pccop0b004-1 :~# iwconfig wlan0 rate 11M
```

```

▶ Frame 3 (140 bytes on wire, 140 bytes captured)
▼ Radiotap Header v0, Length 24
  Header revision: 0
  Header pad: 0
  Header length: 24
  ▶ Present flags: 0x0000482e
  ▶ Flags: 0x00
  Data Rate: 11.0 Mb/s
  Channel frequency: 2412 [BG 1]
  ▶ Channel type: 802.11g (pure-g) (0x00c0)
  SSI Signal: -34 dBm
  Antenna: 1
  802.11 FCS: 0x00000000 [incorrect, should be 0xd72cb171]

```

Ainsi, le champ *Data Rate* est bien transmis dans l'entête de la couche physique.

27 mars 2011

Quelle est la valeur par défaut de nombre de retransmissions de votre carte ?

```
pccop0b004-1:~# iwconfig wlan0
wlan0 IEEE 802.11bg ESSID:"alexisaurele"
      Mode:Ad-Hoc Frequency:2.412 GHz Cell: D2:FF:9F:62:AB:01
      Tx-Power=4 dBm
      Retry long limit:7 RTS thr=500 B Fragment thr:off
      Encryption key:off
      Power Management:off
```

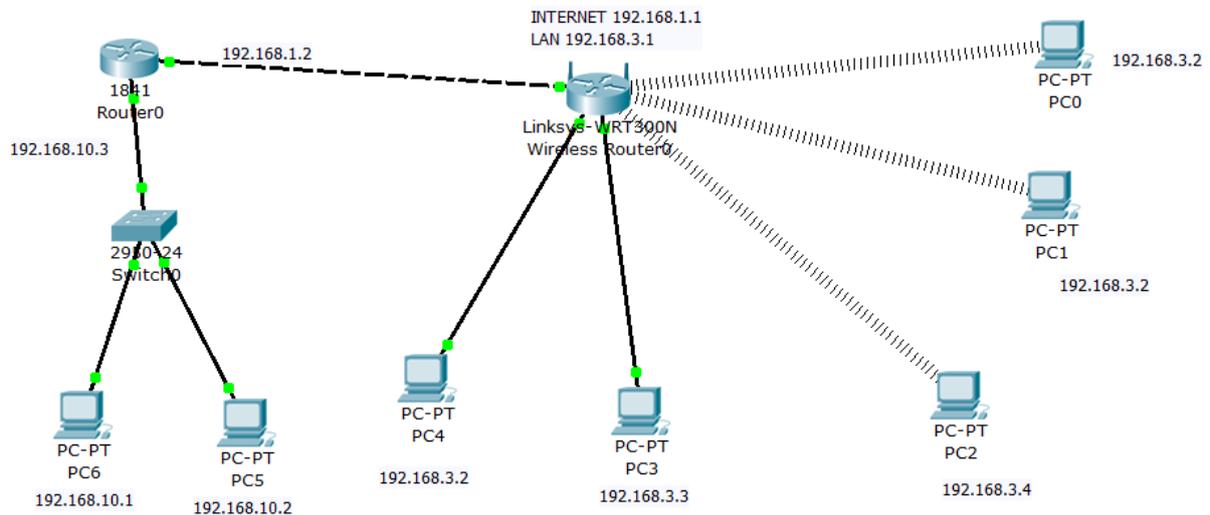
Avec la commande *iwconfig* (lorsque RTS est activé), le champ *Retry long limit* indique le nombre de retransmissions de l'interface *wlan0*.

Comment peut-on modifier cette valeur ?

Cette valeur est modifiable avec ma commande `iwconfig wlan0 retry 10` pour passer à 10 retransmissions maximum.

7. WiFi sous Packet tracer

Packet tracer est un logiciel de simulation de réseaux virtuels créée par cisco. Nous avons configuré PC0, PC1, PC2, PC3 et PC4 en IP dynamique, PC5 et PC6 possèdent des adresses statiques.



La connexion fonctionne.

Attention !



Veillez à mettre sur ON les ports ethernet du switch.

Nous effectuons un ping à partir de PC5 vers PC2.

```
PC>ping 192.168.3.4

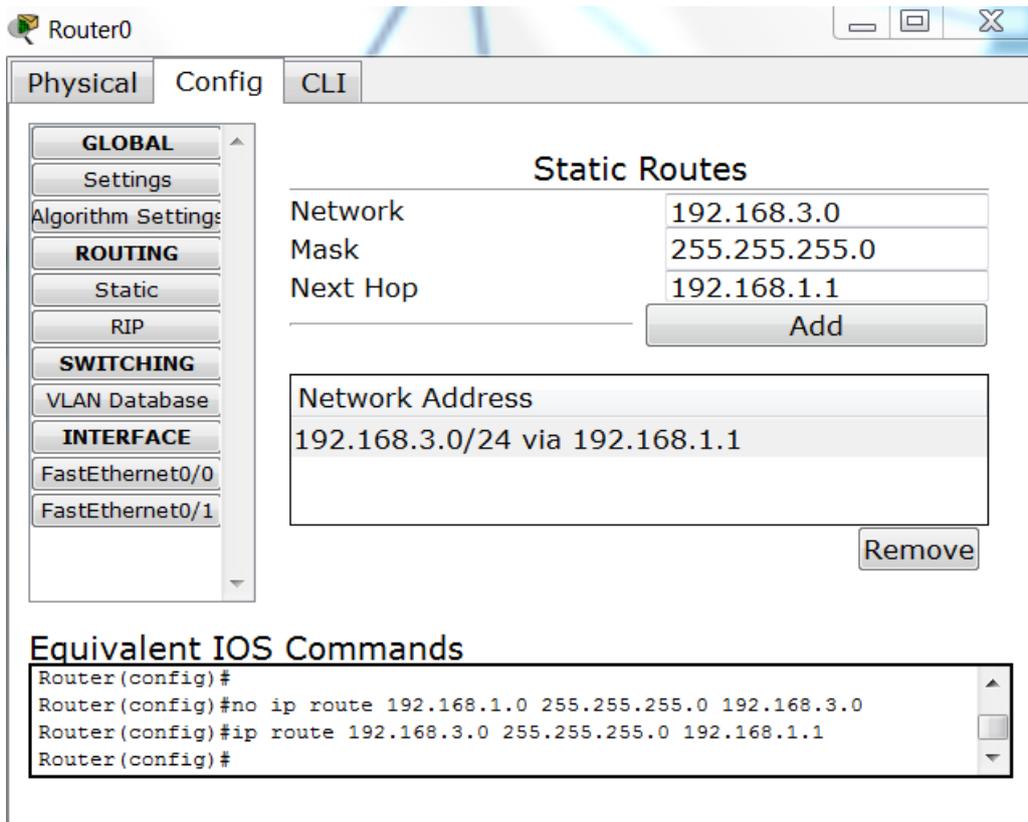
Pinging 192.168.3.4 with 32 bytes of data:

Reply from 192.168.10.3: Destination host unreachable.

Ping statistics for 192.168.3.4:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Sans succès, la machine est inaccessible car elle n'appartient pas au même réseau que la machine source (source : 192.168.10.3, destination : 192.168.3.4).

Pour remédier à cela nous ajoutons une route sur le routeur.



The screenshot shows the Router0 configuration window with the CLI tab selected. The left sidebar contains a tree view with categories: GLOBAL, ROUTING, SWITCHING, and INTERFACE. Under ROUTING, the 'Static' option is selected. The main area is titled 'Static Routes' and contains a table with the following entries:

Network	192.168.3.0
Mask	255.255.255.0
Next Hop	192.168.1.1

Below the table is an 'Add' button. Underneath, a text box displays the resulting configuration: 'Network Address 192.168.3.0/24 via 192.168.1.1', with a 'Remove' button to its right. At the bottom, the 'Equivalent IOS Commands' section shows the following commands in a text area:

```
Router(config)#  
Router(config)#no ip route 192.168.1.0 255.255.255.0 192.168.3.0  
Router(config)#ip route 192.168.3.0 255.255.255.0 192.168.1.1  
Router(config)#
```

De cette manière les paquets en direction du routeur sont redirigés sur le réseau local appartenant au routeur sans fil.

```
PC>ping 192.168.3.4  
  
Pinging 192.168.3.4 with 32 bytes of data:  
  
Reply from 192.168.1.1: bytes=32 time=26ms TTL=126  
Reply from 192.168.1.1: bytes=32 time=22ms TTL=126  
Reply from 192.168.1.1: bytes=32 time=26ms TTL=126  
Reply from 192.168.1.1: bytes=32 time=17ms TTL=126  
  
Ping statistics for 192.168.3.4:  
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:  
    Minimum = 17ms, Maximum = 26ms, Average = 22ms
```

Et les deux stations peuvent maintenant communiquer correctement.

Conclusion

Nous arrivons à la fin de ce TP avec de meilleures connaissances dans le monde du WiFi. Nous avons étudié en détails le principe CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) composé de trames RTS (*Ready-To-Send*) et CTS (*Clear-To-Send*) et d'acquittement (ACK) permettant de diminuer le risque de collisions des paquets transmis. D'autre part nous avons pu découvrir Packet Tracer qui se montre être une application très utile et efficace pour simuler des configurations de réseaux divers et variés.