# SCAPY ET LE MODÈLE TCP/IP


- Scapy est une bibliothèque Python permettant la création, la manipulation et la réception/envoi de paquets réseau.
- Il peut aussi servir à faire des tâches comme du scanning, tracerouting, probing, et même des attaques.
- Scapy supporte un grand nombre de protocoles, et vous permet d'ajouter le vôtre.

**Installation :**

- Scapy est disponible via pip:

```
pip install scapy
```

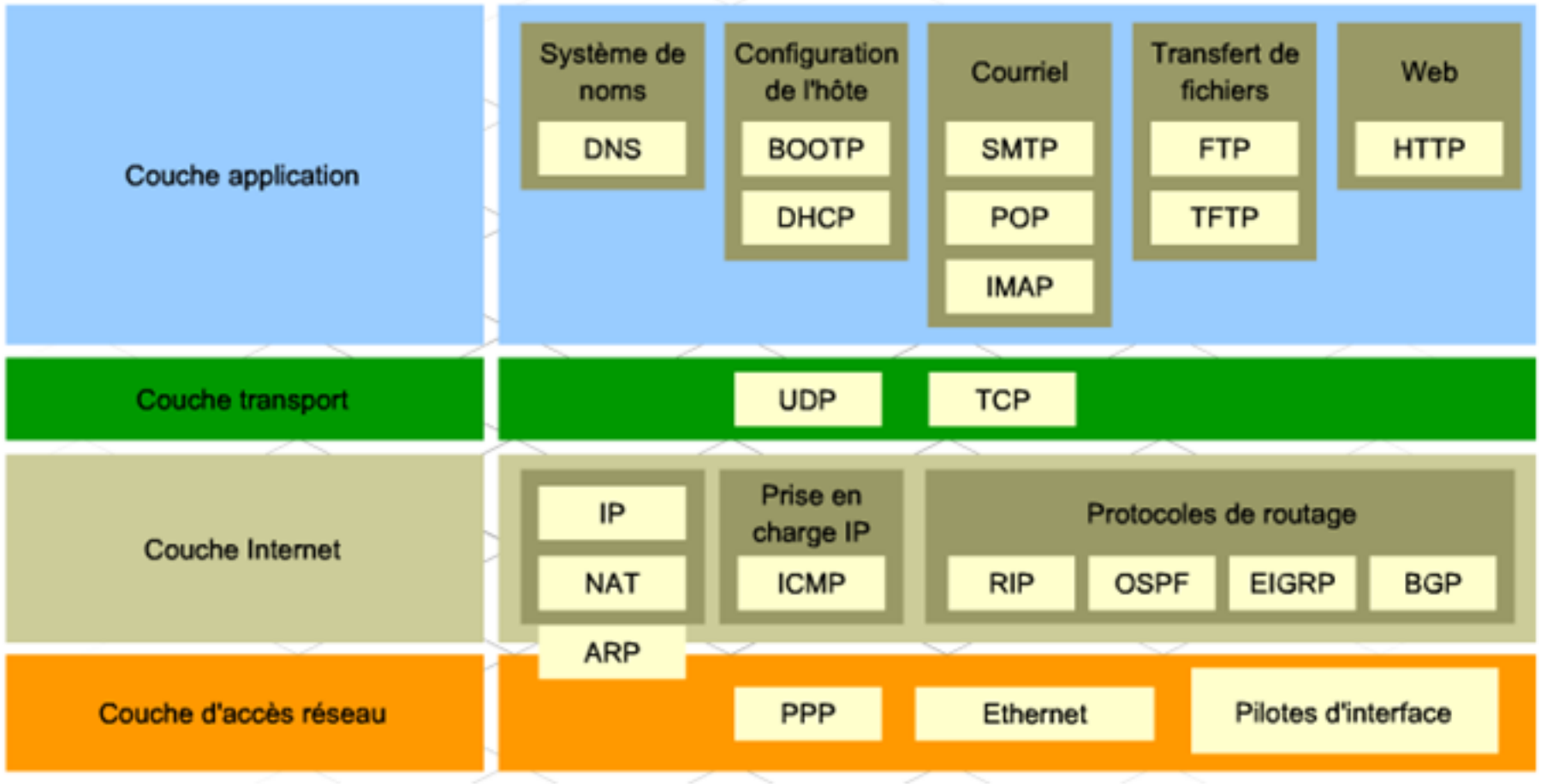
- On peut aussi l'installer depuis le gestionnaire de paquet apt :

```
apt install python3-scapy
```

# LES MODÈLES

| | Modèle OSI | Périphérique / Description | Modèle TCP/IP |
|---|---|---|---|
| 7 | Application |  Services applicatifs au plus proche des utilisateurs | Application |
| 6 | Présentation |  Encode, chiffre, compresse les données utiles | |
| 5 | Session |  Etablit des sessions entre des applications | |
| 4 | Transport |  Etablit, maintien et termine des sessions entre des périphériques terminaux | Transport |
| 3 | Réseau |  Adresse les interfaces globalement et détermine les meilleurs chemins à travers un inter-réseau | Internet |
| 2 | Liaison de Données |  Adresse localement les interfaces, livre les informations localement, méthode MAC | Accès au Réseau |
| 1 | Physique |   Encodage du signal, câblage et connecteurs, spécifications physiques | |

# LES PROTOCOLES ET LE MODÈLE TCP/IP



# LA COUCHE 1 : ACCÈS AU RÉSEAU

```
from scapy.all import *

Layer1=Ether()
Layer1.show()
```

```
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 78:4f:43:6c:f5:bb
  type      = 0x9000
```

```
from scapy.all import *

Layer1=Ether(src="01:02:03:04:05:06")
Layer1.show()
```

```
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 01:02:03:04:05:06
  type      = 0x9000
```

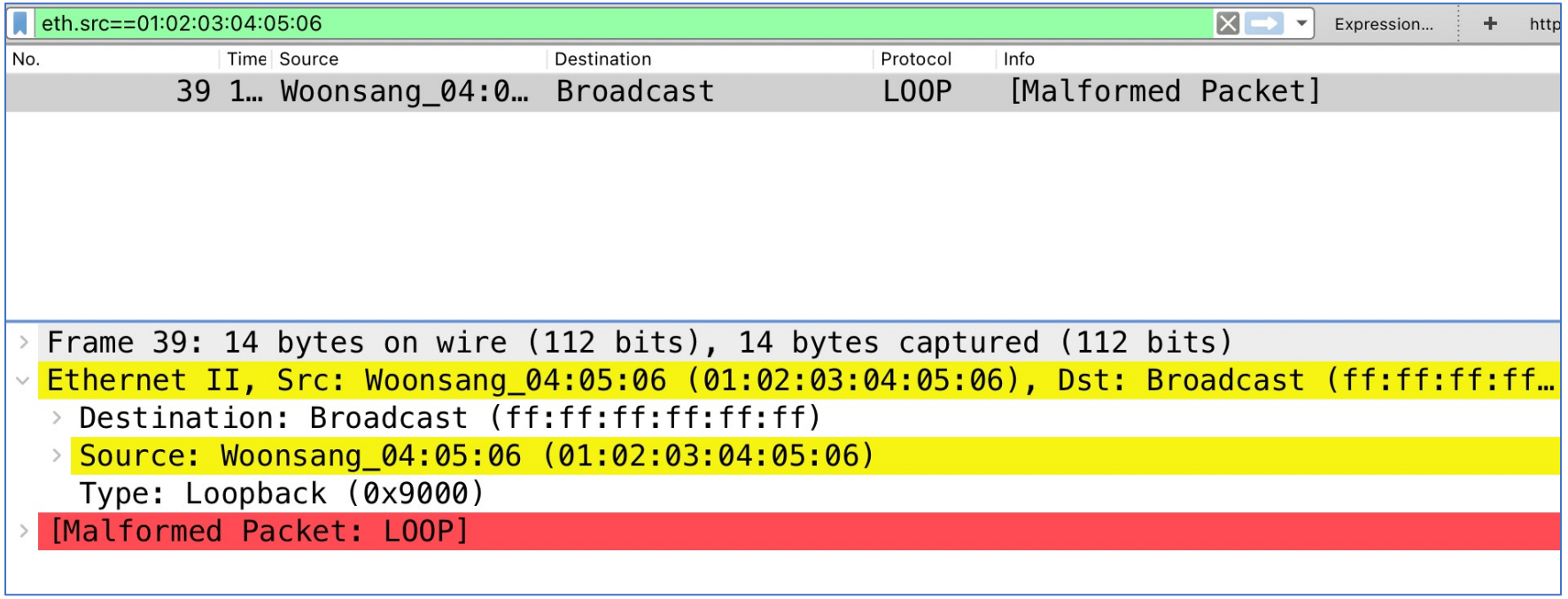
```
inet 192.168.1.49 netmask 0xffffff00 broadcast 192.168.1.255
```

Accès au Réseau

# TRANSMISSION DE LA COUCHE 1

```
from scapy.all import *

Layer1=Ether(src="01:02:03:04:05:06")
Sendp(Layer1)
```

Accès au Réseau



# LA COUCHE 2 : INTERNET

```
from scapy.all import *

Layer2=IP()
Layer2.show()
```

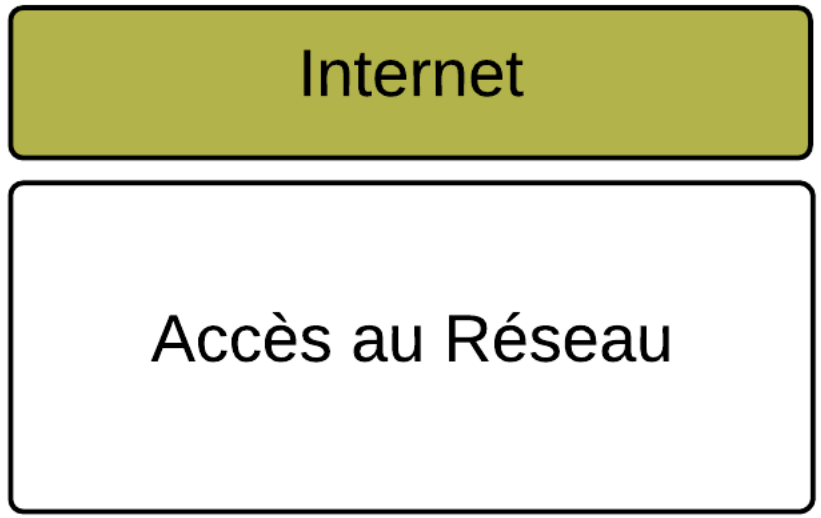
```
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = hopopt
  chksum    = None
  src       = 127.0.0.1
  dst       = 127.0.0.1
  \options   \
```


```
from scapy.all import *

Layer2=IP(dst="192.168.1.254")
Layer2.show()
```

```
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = ip
  chksum    = None
  src       = 192.168.1.49
  dst       = 192.168.1.254
  \options   \
```

# TRANSMISSION DE LA COUCHE 2

```
from scapy.all import *

Layer1=Ether(src="01:02:03:04:05:06")
Layer2=IP(dst="192.168.1.254")
Sendp(Layer1/Layer2)
```




```
      5937 7… 192.168.1.49    192.168.1.254      IPv4
```

```
> Frame 5937: 34 bytes on wire (272 bits), 34 bytes captured (272 bits)
v Ethernet II, Src: Woonsang_04:05:06 (01:02:03:04:05:06), Dst: e4:9e:12:7a:34:3a (e4:…
  > Destination: e4:9e:12:7a:34:3a (e4:9e:12:7a:34:3a)
  > Source: Woonsang_04:05:06 (01:02:03:04:05:06)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 192.168.1.49, Dst: 192.168.1.254
```

# LA COUCHE 3 : TRANSPORT

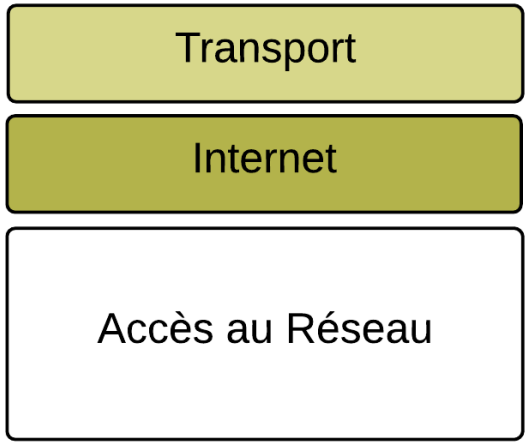
```
from scapy.all import *

Layer3=TCP()
Layer3.show()
```

```
from scapy.all import *

Layer1=Ether(src="01:02:03:04:05:06")
Layer2=IP(dst="192.168.1.254")
Layer3=TCP()
Sendp(Layer1/Layer2/Layer3)
```

```
###[ TCP ]###
  sport     = ftp_data
  dport     = http
  seq       = 0
  ack       = 0
  dataofs   = None
  reserved  = 0
  flags     = S
  window    = 8192
  chksum    = None
  urgptr    = 0
  options   = ''
```

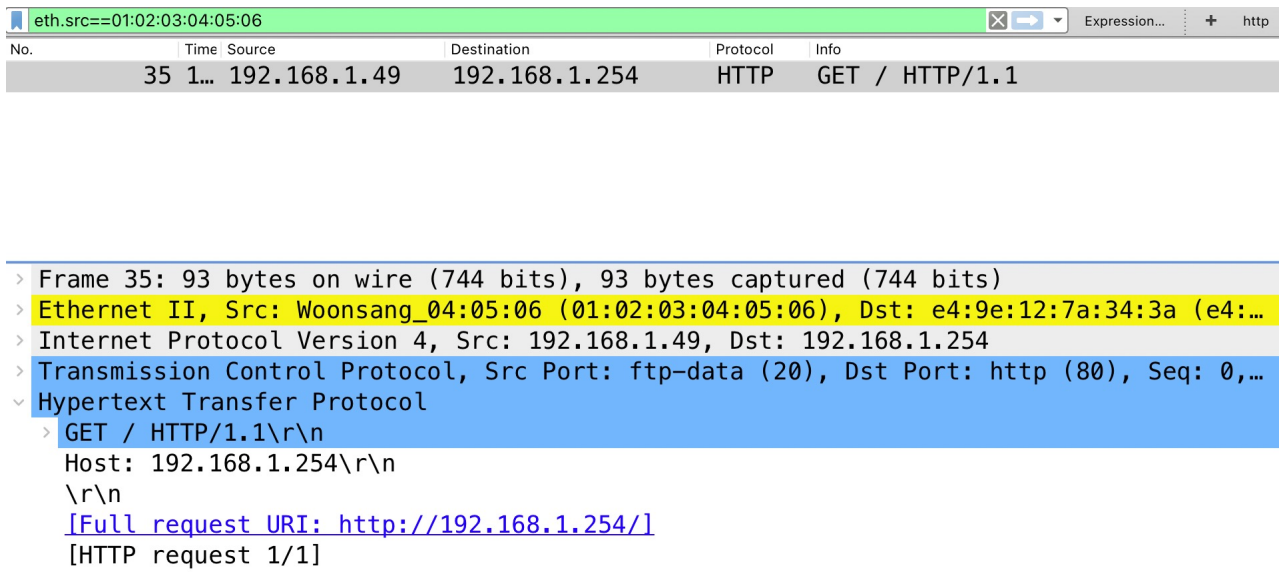
| No. | Time | Source | Destination | Protocol | Info |
|---|---|---|---|---|---|
| 3222 | 2… | 192.168.1.49 | 192.168.1.254 | TCP | ftp-data → http [SYN] Seq=0 Wi… |

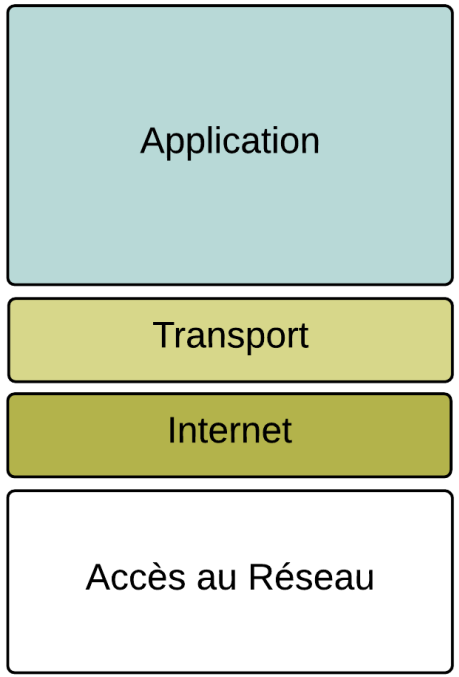
```
> Frame 3222: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
v Ethernet II, Src: Woonsang_04:05:06 (01:02:03:04:05:06), Dst: e4:9e:12:7a:34:3a (e4:…
  > Destination: e4:9e:12:7a:34:3a (e4:9e:12:7a:34:3a)
  > Source: Woonsang_04:05:06 (01:02:03:04:05:06)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 192.168.1.49, Dst: 192.168.1.254
> Transmission Control Protocol, Src Port: ftp-data (20), Dst Port: http (80), Seq: 0,…
```



# LA COUCHE 4 : APPLICATION

```
from scapy.all import *

Layer1=Ether(src="01:02:03:04:05:06")
Layer2=IP(dst="192.168.1.254")
Layer3=TCP()
Layer4=Raw(load="GET / HTTP/1.1\r\nHost: 192.168.1.254\r\n\r\n")

Sendp(Layer1/Layer2/Layer3/Layer4)
```



# ACCÉDER AUX PROPRIÉTÉS DE LA COUCHE BASSE ETHERNET

**Capture de trames Ethernet :**


Vous pouvez utiliser Scapy pour capturer des trames Ethernet sur votre réseau:

```
from scapy.all import sniff

packets = sniff(count=10, filter="ether", iface="en0")
packets.summary()
```

Ce code capture 10 trames Ethernet sur l'interface eth0.

**Analyse d'une trame capturée :**

```
frame = packets[0]
frame.show()
```

Cela affichera tous les détails de la première trame capturée.

On peut ainsi récupérer l’adresse MAC source de la première trame capturée :

```
mac_address  = packets[0].src
print(mac_address)
```

# ACCÉDER AUX PROPRIÉTÉS DE LA COUCHE IP

```
from scapy.all import sniff,IP

# Capture des paquets
packets = sniff(count=10, filter="ether", iface="en0")

# Afficher un résumé des paquets capturés
packets.summary()

# Sélectionner le premier paquet et l'afficher
frame = packets[0]
frame.show()

# Extraire l'adresse IP source et l'afficher

# Vérifier si le paquet contient une couche IP et extraire l'adresse IP
source
if IP in frame:
    ip_address = frame[IP].src
    print("Adresse IP Source:", ip_address)
```

**Création d'une trame Ethernet :**

```
from scapy.all import Ether

eth_frame = Ether() #Ether est une classe et Ether() est une instance de cette classe
print(eth_frame.summary())
```

summary(): c'est une méthode qui renvoie un résumé court du paquet.
On aurait pu utiliser la méthode show(). Cette méthode affiche un résumé lisible du paquet.

```
sendp(packet, iface="en0")
```

Cette fonction envoie la trame précédente.
**sendp** est utilisé pour envoyer des trames au niveau de la couche de liaison de données (couche 2).
Par défaut, Scapy remplira les champs avec des valeurs appropriées, comme l'adresse MAC source de votre carte réseau. Si vous souhaitez spécifier des valeurs, vous pouvez le faire lors de la création:

```
eth_frame = Ether(dst="ff:ff:ff:ff:ff:ff", src="00:01:02:03:04:05", type=0x0800)
```

Les principales propriétés d'Ether sont dst, src et type. On peut les relever avec **eth_frame.dst, eth_frame.src et eth_frame.type**

## Encapsulation d'autres protocoles avec Ethernet :

Vous pouvez combiner Ether avec d'autres protocoles pour simuler l'encapsulation:


```
from scapy.all import Ether, IP, ICMP

packet = Ether()/IP(dst="8.8.8.8")/ICMP()
print(packet.summary())
```

Ce code crée une trame Ethernet qui contient un paquet IP qui, à son tour, contient un paquet ICMP (ping).

L'opérateur / est utilisé pour l'encapsulation. En utilisant la notation packet[Protocol], vous pouvez accéder directement à la couche spécifique du paquet que vous souhaitez examiner ou manipuler.

Si vous capturez une réponse avec sr1() et stockez le résultat dans response, vous pouvez également accéder à ses différentes couches en utilisant cette notation :

```
icmp_layer = response[ICMP]
ip_layer = response[IP]
```

# CODER UN PING AVEC SCAPY

Pour faire un ping, cela revient à envoyer un paquet ICMP Echo Request à une adresse cible et d'attendre une réponse ICMP Echo Reply. ICMP est un protocole qui fonctionne avec IP sur le niveau 2.


Création du Paquet ICMP Echo Request :

- Créez un objet IP en spécifiant l'adresse IP de destination.
- Créez un objet ICMP. Par défaut, Scapy crée un paquet ICMP Echo Request.

```
destination_ip = "8.8.8.8"  # Exemple avec l'adresse IP de Google DNS
packet = IP(dst=destination_ip)/ICMP()
```

Envoi du Paquet et Réception de la Réponse :

- Utilisez la fonction sr1() pour envoyer le paquet et attendre la première réponse. Cette fonction est utile pour les requêtes où une réponse est attendue, comme le ping.
- sr1() envoie le paquet et attend une réponse correspondante. Dans ce cas, elle enverra le paquet ICMP Echo Request et attendra un Echo Reply.

```
response = sr1(packet, timeout=2)  # timeout de 2s pour la réponse
```

Analyse de la Réponse :

- Vérifiez si vous avez reçu une réponse et, si c'est le cas, affichez les détails pertinents.

```
if response is not None:
    response.show()  # Afficher tous les détails de la réponse
    if response.type == 0:  # Le type 0 correspond à un Echo Reply
        print(f"Réponse reçue de {response.src} : Le hôte est en ligne.")
else:
    print("Aucune réponse reçue : L’hôte hors ligne ou bloque les pings.")
```