

# Réseaux Avancés

*TP NETKIT*

## 4. Protocole ARP

### Q4.1

Le routeur r1 a dans son cache ARP les adresses de r2 et de pc1 et le routeur r2 a dans son cache ARP les adresses de r1 et de pc2. En effet la commande ping a permis aux routeurs de stocker ces adresses.

### Q4.2

#### Séquence de messages capturés par tcpdump dans r2 :

```
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
fe:fd:c8:01:01:07 (oui Unknown) > Broadcast, ethertype ARP (0x0806),
length 42: arp who-has 200.1.1.1 tell 200.1.1.7
fe:fd:c8:01:01:01 (oui Unknown) > fe:fd:c8:01:01:07 (oui Unknown),
ethertype ARP (0x0806), length 42: arp reply 200.1.1.1 is-at
fe:fd:c8:01:01:01 (oui Unknown)
fe:fd:c8:01:01:07 (oui Unknown) > fe:fd:c8:01:01:01 (oui Unknown),
ethertype IPv4 (0x0800), length 98: 200.1.1.7 > 195.11.14.5: ICMP
echo request, id 1282, seq 1, length 64
fe:fd:c8:01:01:01 (oui Unknown) > fe:fd:c8:01:01:07 (oui Unknown),
ethertype IPv4 (0x0800), length 98: 195.11.14.5 > 200.1.1.7: ICMP
echo reply, id 1282, seq 1, length 64
```

#### Séquence de messages capturés par tcpdump dans r1 :

```
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
12:3e:e2:7d:e3:87 (oui Unknown) > Broadcast, ethertype ARP (0x0806),
length 42: arp who-has 100.0.0.9 tell 100.0.0.10
fa:de:dc:30:96:57 (oui Unknown) > 12:3e:e2:7d:e3:87 (oui Unknown),
ethertype ARP (0x0806), length 42: arp reply 100.0.0.9 is-at
fa:de:dc:30:96:57 (oui Unknown)
```

```
12:3e:e2:7d:e3:87 (oui Unknown) > fa:de:dc:30:96:57 (oui Unknown),  
ethertype IPv4 (0x0800), length 98: 200.1.1.7 > 195.11.14.5: ICMP  
echo request, id 1282, seq 1, length 64  
fa:de:dc:30:96:57 (oui Unknown) > 12:3e:e2:7d:e3:87 (oui Unknown),  
ethertype IPv4 (0x0800), length 98: 195.11.14.5 > 200.1.1.7: ICMP  
echo reply, id 1282, seq 1, length 64
```

### **Séquence de messages capturés par tcpdump dans pc1 :**

```
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes  
fe:fd:c3:0b:0e:01 (oui Unknown) > Broadcast, ethertype ARP (0x0806),  
length 42: arp who-has 195.11.14.5 tell 195.11.14.1  
6e:5f:98:37:0c:07 (oui Unknown) > fe:fd:c3:0b:0e:01 (oui Unknown),  
ethertype ARP (0x0806), length 42: arp reply 195.11.14.5 is-at  
6e:5f:98:37:0c:07 (oui Unknown)  
fe:fd:c3:0b:0e:01 (oui Unknown) > 6e:5f:98:37:0c:07 (oui Unknown),  
ethertype IPv4 (0x0800), length 98: 200.1.1.7 > 195.11.14.5: ICMP  
echo request, id 1282, seq 1, length 64  
6e:5f:98:37:0c:07 (oui Unknown) > fe:fd:c3:0b:0e:01 (oui Unknown),  
ethertype IPv4 (0x0800), length 98: 195.11.14.5 > 200.1.1.7: ICMP  
echo reply, id 1282, seq 1, length 64
```

## 5. Protocole RIPv2

### Q5.1

#### Résultat :

```
r4:~# ping 100.1.0.13
PING 100.1.0.13 (100.1.0.13) 56(84) bytes of data.
64 bytes from 100.1.0.13: icmp_seq=1 ttl=64 time=7.70 ms
^C
--- 100.1.0.13 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 7.700/7.700/7.700/0.000 ms
r4:~# ping 100.1.2.1
connect: Network is unreachable
```

Pourquoi vous avez eu ce résultat ?

#### Réponse :

Nous avons ce résultat car le routeur r4 a essayé de pingé l'interface eth2 du routeur r2 or r4 ne connaît pas le chemin pour aller à cette interface.

La table de routage de r4 est la suivante :

```
r4:~# route
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use
Iface
100.1.0.16       *              255.255.255.252 U         0      0      0 eth2
100.2.0.0        *              255.255.255.252 U         0      0      0 eth0
100.1.0.12       *              255.255.255.252 U         0      0      0 eth3
100.1.4.0        *              255.255.255.0   U         0      0      0 eth1
```

Nous voyons en effet qu'il n'y a pas d'entrée pour le réseau 100.1.2.0/24.

### Q5.2

#### Résultat :

```
# /etc/init.d/zebra start
Loading capability module if not yet done.
Starting Quagga daemons (prio:10): zebra ripd.
```

### Q5.3

```
r4:~# ping 100.1.2.1
PING 100.1.2.1 (100.1.2.1) 56(84) bytes of data.
64 bytes from 100.1.2.1: icmp_seq=1 ttl=63 time=11.6 ms
```

Maintenant, le routeur r4 arrive à pinger l'interface eth2 de r2 (adresse IP : 100.1.2.1).

Tableau de routage de r4 :

```
r4:~# route
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use
Iface						
100.1.0.16	*	255.255.255.252	U	0	0	0 eth2
100.1.0.0	100.1.0.13	255.255.255.252	UG	2	0	0 eth3
100.1.0.4	100.1.0.17	255.255.255.252	UG	2	0	0 eth2
100.2.0.0	*	255.255.255.252	U	0	0	0 eth0
100.1.0.8	100.1.0.17	255.255.255.252	UG	2	0	0 eth2
100.1.0.12	*	255.255.255.252	U	0	0	0 eth3
100.1.4.0	*	255.255.255.0	U	0	0	0 eth1
<b>100.1.2.0</b>	<b>100.1.0.17</b>	<b>255.255.255.0</b>	<b>UG</b>	<b>3</b>	<b>0</b>	<b>0 eth2</b>
100.1.3.0	100.1.0.17	255.255.255.0	UG	2	0	0 eth2
100.1.1.0	100.1.0.13	255.255.255.0	UG	2	0	0 eth3

Maintenant, il y a une entrée pour le réseau 100.1.2.0/24.

## Q5.4

### Résultat :

```
r4:~# tcpdump -i eth2 -v -n -s 1518
tcpdump: listening on eth2, link-type EN10MB (Ethernet), capture size
1518 bytes
08:51:12.550753 IP (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto
UDP (17), length 172) 100.1.0.17.520 > 224.0.0.9.520:
    RIPv2, Response, length: 144, routes: 7
    AFI: IPv4:      100.1.0.0/30, tag 0x0000, metric: 2, next-hop: self
    AFI: IPv4:      100.1.0.4/30, tag 0x0000, metric: 1, next-hop: self
    AFI: IPv4:      100.1.0.8/30, tag 0x0000, metric: 1, next-hop: self
    AFI: IPv4:      100.1.0.12/30, tag 0x0000, metric: 2, next-hop: self
    AFI: IPv4:      100.1.1.0/24, tag 0x0000, metric: 2, next-hop: self
    AFI: IPv4:      100.1.2.0/24, tag 0x0000, metric: 2, next-hop: self
    AFI: IPv4:      100.1.3.0/24, tag 0x0000, metric: 1, next-hop: self
^C
1 packets captured
1 packets received by filter
0 packets dropped by kernel
```

```
r4:~# traceroute 100.1.2.1
traceroute to 100.1.2.1 (100.1.2.1), 64 hops max, 40 byte packets
 1  100.1.0.17 (100.1.0.17)  4 ms  1 ms  1 ms
 2  100.1.2.1 (100.1.2.1)   8 ms  8 ms  0 ms
```

## Q5.5

### Réponse, en r5 :

```
$route add default gw 100.2.0.1
```

```
r5:~# ping 100.1.2.1
PING 100.1.2.1 (100.1.2.1) 56(84) bytes of data.
64 bytes from 100.1.2.1: icmp_seq=1 ttl=62 time=21.0 ms
64 bytes from 100.1.2.1: icmp_seq=2 ttl=62 time=1.99 ms
^C
--- 100.1.2.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 1.999/11.532/21.065/9.533 ms
```

```
r5:~# traceroute 100.1.2.1
traceroute to 100.1.2.1 (100.1.2.1), 64 hops max, 40 byte packets
 1  100.2.0.1 (100.2.0.1)  2 ms  0 ms  0 ms
 2  100.1.0.17 (100.1.0.17)  1 ms  0 ms  1 ms
 3  100.1.2.1 (100.1.2.1)  1 ms  1 ms  1 ms
```

## Q5.6

Solution, en r4 :

```
$route add default gw 100.2.0.2
```

## Q5.7

```
$ifconfig eth1 down
```

```
r1:~# traceroute 100.1.0.10
traceroute to 100.1.0.10 (100.1.0.10), 64 hops max, 40 byte packets
 1  100.1.0.2 (100.1.0.2)  1 ms  1 ms  0 ms
 2  100.1.0.10 (100.1.0.10)  0 ms  0 ms  0 ms
```

**Réponse :** Les paquets passent par r2 (adresse 100.1.0.2) dans cette phase transitoire.

## 6. Bridging

### Q6.1

```
switch1:~# brctl showmacs br0
```

port	no	mac	addr	is local?	ageing timer
1	00:00:00:00:01:00	yes		0.00	
2	00:00:00:00:01:01	yes		0.00	

```
switch2:~# brctl showmacs br0
```

port	no	mac	addr	is local?	ageing timer
2	00:00:00:00:01:01	no		1.31	
1	00:00:00:00:02:00	yes		0.00	
2	00:00:00:00:02:01	yes		0.00	

### Q6.2

```
pc3:~# tcpdump -e -q
```

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol  
decode
```

```
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
```

```
09:32:41.499433 00:00:00:00:02:00 (oui Ethernet) > 01:80:c2:00:00:00
```

```
(oui Unknown), 802.3, length 52: LLC, dsap STP (0x42) Individual,  
ssap STP (0x42) Command, ctrl 0x03: STP 802.1d, Config, Flags [none],  
bridge-id 8000.00:00:00:00:02:00.8001, length 35
```

```
09:32:41.531497 00:00:00:00:20:00 (oui Ethernet) > Broadcast, ARP,  
length 42: arp who-has 10.0.0.3 tell 10.0.0.2
```

```
09:32:41.531539 00:00:00:00:30:00 (oui Ethernet) > 00:00:00:00:20:00  
(oui Ethernet), ARP, length 42: arp reply 10.0.0.3 is-at  
00:00:00:00:30:00 (oui Ethernet)
```

```
09:32:41.532027 00:00:00:00:20:00 (oui Ethernet) > 00:00:00:00:30:00  
(oui Ethernet), IPv4, length 98: 10.0.0.2 > 10.0.0.3: ICMP echo  
request, id 1282, seq 1, length 64
```

```
09:32:41.532085 00:00:00:00:30:00 (oui Ethernet) > 00:00:00:00:20:00  
(oui Ethernet), IPv4, length 98: 10.0.0.3 > 10.0.0.2: ICMP echo  
reply, id 1282, seq 1, length 64
```

```
^C
```

```
7 packets captured
```

```
7 packets received by filter
```

0 packets dropped by kernel

### Q6.3

```
switch1:~# brctl showmacs br0
```

port	no	mac addr		is local?	ageing timer
1	00:00:00:00:01:00	yes		0.00	
2	00:00:00:00:01:01	yes		0.00	
2	00:00:00:00:20:00	no		100.14	

```
switch2:~# brctl showmacs br0
```

port	no	mac addr		is local?	ageing timer
2	00:00:00:00:01:01	no		0.53	
1	00:00:00:00:02:00	yes		0.00	
2	00:00:00:00:02:01	yes		0.00	
1	00:00:00:00:20:00	no		107.51	
1	00:00:00:00:30:00	no		107.51	

switch1 connaît la position de pc2 (MAC ADDRESS : 00:00:00:00:20:00).

switch2 connaît la position de pc2 (MAC ADDRESS : 00:00:00:00:20:00) et pc3 (MAC ADDRESS : 00:00:00:00:30:00).

### Q6.4

```
pc1:~# tcpdump -e -q
```

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode  
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
```

```
09:42:45.555992 00:00:00:00:01:00 (oui Ethernet) > 01:80:c2:00:00:00 (oui Unknown), 802.3,  
length 52: LLC, dsap STP (0x42) Individual, ssap STP (0x42) Command, ctrl 0x03: STP 802.1d,  
Config, Flags [none], bridge-id 8000.00:00:00:00:01:00.8001, length 35
```

```
09:42:46.456172 00:00:00:00:20:00 (oui Ethernet) > Broadcast, ARP, length 42: arp who-has  
10.0.0.3 tell 10.0.0.2
```

```
09:42:47.555764 00:00:00:00:01:00 (oui Ethernet) > 01:80:c2:00:00:00 (oui Unknown), 802.3,  
length 52: LLC, dsap STP (0x42) Individual, ssap STP (0x42) Command, ctrl 0x03: STP 802.1d,  
Config, Flags [none], bridge-id 8000.00:00:00:00:01:00.8001, length 35
```

```
09:42:49.555779 00:00:00:00:01:00 (oui Ethernet) > 01:80:c2:00:00:00 (oui Unknown), 802.3,  
length 52: LLC, dsap STP (0x42) Individual, ssap STP (0x42) Command, ctrl 0x03: STP 802.1d,  
Config, Flags [none], bridge-id 8000.00:00:00:00:01:00.8001, length 35
```

Les paquets que l'on voit atteindre en pc1 sont les paquets requêtes allant de pc2 vers pc3 (10.0.0.2 > 10.0.0.3).



## Q6.5

```
pc1:~# tcpdump -e -q
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes

09:48:39.555691 00:00:00:00:01:00 (oui Ethernet) > 01:80:c2:00:00:00 (oui Unknown), 802.3,
length 52: LLC, dsap STP (0x42) Individual, ssap STP (0x42) Command, ctrl 0x03: STP 802.1d,
Config, Flags [none], bridge-id 8000.00:00:00:00:01:00.8001, length 35

09:48:39.999967 00:00:00:00:20:00 (oui Ethernet) > 00:00:00:00:30:00 (oui Ethernet), IPv4,
length 98: 10.0.0.2 > 10.0.0.3: ICMP echo request, id 2306, seq 2, length 64
```

Les paquets que l'on voit atteindre en pc1 sont les paquets requêtes allant de pc2 vers pc3 (10.0.0.2 > 10.0.0.3) et les réponses ICMP allant de pc3 vers pc2 (10.0.0.3 > 10.0.0.2).

## 7. Web Server and Browser

### Q7.1

```
<html><body><h1>It works!</h1></body></html>
```

### Q7.2

It works!

### Q7.3

```
10.0.0.2 - - [12/Nov/2014:18:53:40 +0000] "GET / HTTP/1.1" 200 56 "-"
"Links (2.2; Linux 2.6.26.5-netkit-K2.8 i686; 127x64)"
```

### Q7.4

```
[Wed Nov 12 18:51:39 2014] [notice] Apache/2.2.9 (Debian) configured
-- resuming normal operations
```

### Q7.5

```
server:~# apache2 -l
```

Compiled in modules:

```
core.c
mod_log_config.c
mod_logio.c
worker.c
http_core.c
mod_so.c
```

```
server:~# a2enmod rewrite
```

Enabling module rewrite.

Run '/etc/init.d/apache2 restart' to activate new configuration!

## Q7.6

```
server:~# a2enmod
Which module(s) do you want to enable (wildcards ok)?
userdir
Enabling module userdir.
Run '/etc/init.d/apache2 restart' to activate new configuration!
```

## Q7.7

```
$mkdir public_html
$nano index.html
```

## Q7.8

La directive `DirectoryIndex custom_file.html` permet de faire en sorte que lorsque le client arrive sur l'url <http://10.0.0.1/~guest>, le fichier html `custom_file.html` se charge.

## 8. Common Gateway Interface

### Q8.1

#### Scénario 1 :

Il faut sélectionner le lien “This link”, le script `/cgi-bin/test.sh` est alors lancé avec les arguments “foo”, “bar” et “baz”. Nous obtenons alors cette page :

```
I have received the parameters as command line arguments :  
Number of arguments : 3  
Argument : foo  
Argument : bar  
Argument : baz
```

#### Scénario 2 :

L'utilisateur saisi des informations dans un formulaire et la réponse est envoyé via la méthode **GET**, nous obtenons alors cette page :

```
I have received the parameters through the QUERY_STRING environment  
variable :
```

```
Variable contents : comment=my+comment&vote=yes
```

#### Scénario 3 :

L'utilisateur saisi des informations dans un formulaire et la réponse est envoyé via la méthode **POST**, nous obtenons alors cette page :

```
I have received the parameters as standard input.
```

```
comment=my+comment&vote=no
```