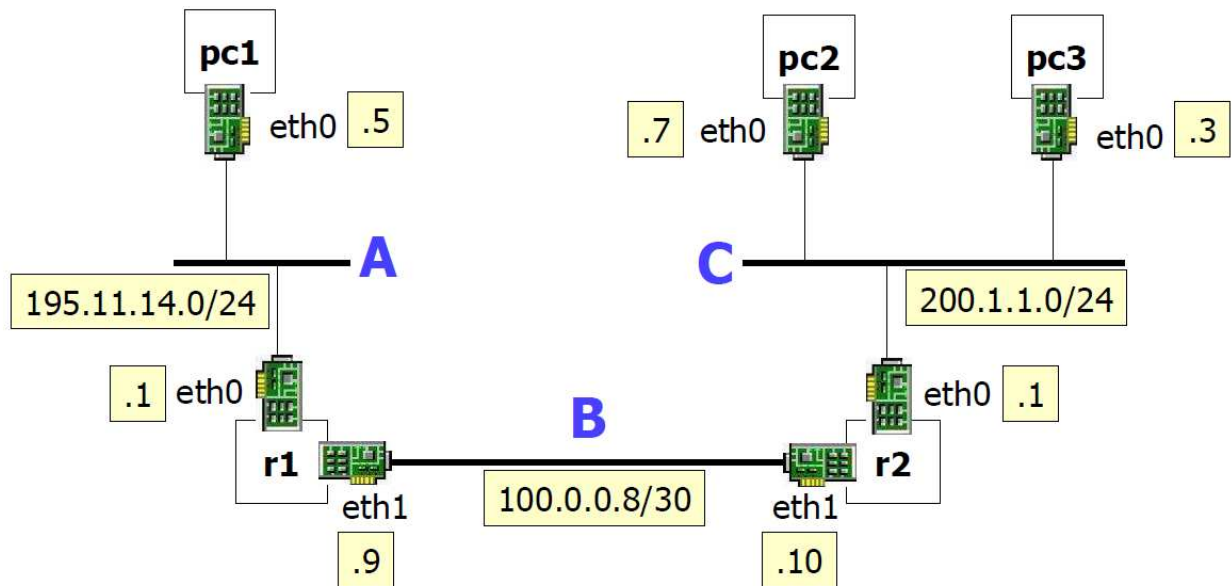
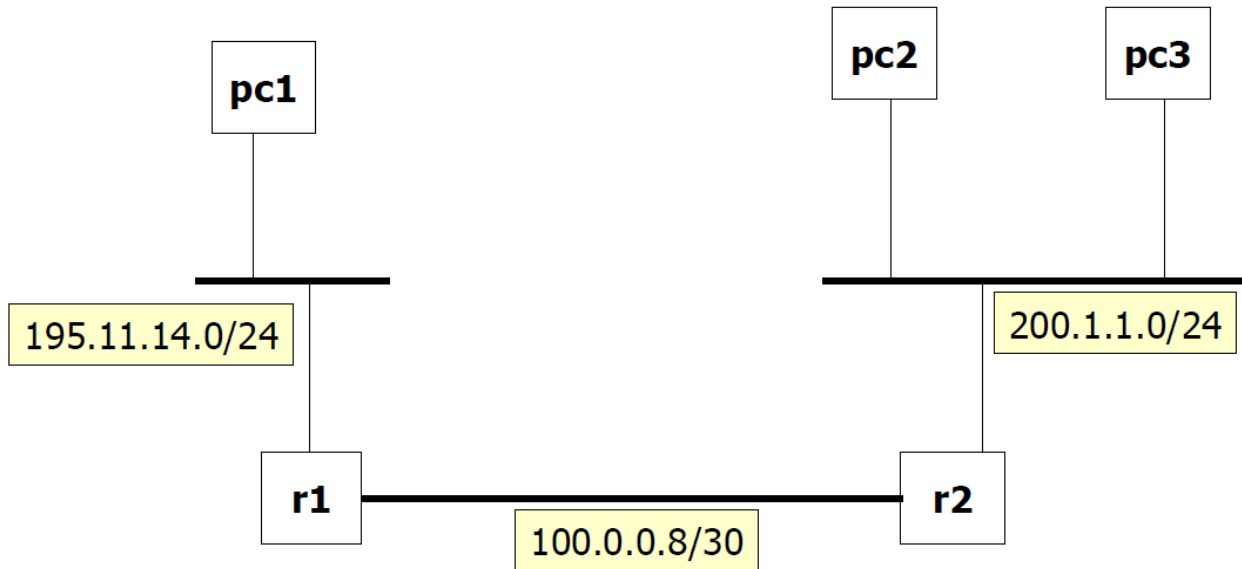


## 4. Protocole ARP

[http://www.netkit.org/netkit-labs/netkit-labs\\_basic-topics/netkit-lab\\_arp/netkit-lab\\_arp.pdf](http://www.netkit.org/netkit-labs/netkit-labs_basic-topics/netkit-lab_arp/netkit-lab_arp.pdf)

Nous voulons créer le scénario suivant (la configuration détaillée des interfaces est dans la 2ème figure) :



Nous allons utiliser le laboratoire disponible ici :

[http://www.netkit.org/netkit-labs/netkit-labs\\_basic-topics/netkit-lab\\_arp/netkit-lab\\_arp.tar.gz](http://www.netkit.org/netkit-labs/netkit-labs_basic-topics/netkit-lab_arp/netkit-lab_arp.tar.gz)

Sur la machine réelle utiliser les commandes :

```
cd netkit-lab_arp
```

```
lstart
```

Nous allons utiliser la commande "arp", qui permet de manipuler l'*arp cache* du système.

Nous allons tout d'abord nous mettre dans pc3, et envoyer du trafic vers pc2 (ping). Il s'agit donc de trafic "local" (qui ne traverse pas de routeur).

- 1) exécuter la commande "arp" dans pc3 avant d'exécuter la commande "ping"
- 2) exécuter la commande "ping 200.1.1.7"
- 3) exécuter à nouveau la commande "arp" dans pc3, et observer la différence.
- 4) Se mettre dans pc2 et exécuter aussi la commande "arp"

Maintenant, on va générer du trafic entre pc2 et pc1 (ce trafic traverse donc les deux routeurs).

- 1) Dans pc2, exécuter "ping 195.11.14.5"
- 2) Vérifier l'état de l'arp cache dans pc2

**Q4.1** : Qu'est-ce qui se passe dans les routeurs r1 et r2 ? Vérifiez l'état de leur arp cache aussi.

Pour observer l'échange de messages arp request/reply avec tcpdump, on va faire redémarrer le laboratoire.

Dans la machine réelle on utilise donc les commandes lcrash et à nouveau lstart :

```
user@localhost:~/netkit-lab_arp$ lcrash
```

```
user@localhost:~/netkit-lab_arp$ lstart
```

Ensuite nous allons utiliser la commande : tcpdump -e -t -i eth0

e = montre les adresses MAC

t = supprime les timestamps

- 1a) Utiliser la commande tcpdump -e -t -i eth0 dans r2 (on reste à l'écoute)
- 1b) Utiliser la commande tcpdump -e -t -i eth1 dans r1 (on reste à l'écoute)
- 1c) Utiliser la commande tcpdump -e -t -i eth0 dans pc1 (on reste à l'écoute)
- 2) En même temps, on se met à nouveau dans pc2 et on fait ping vers pc1

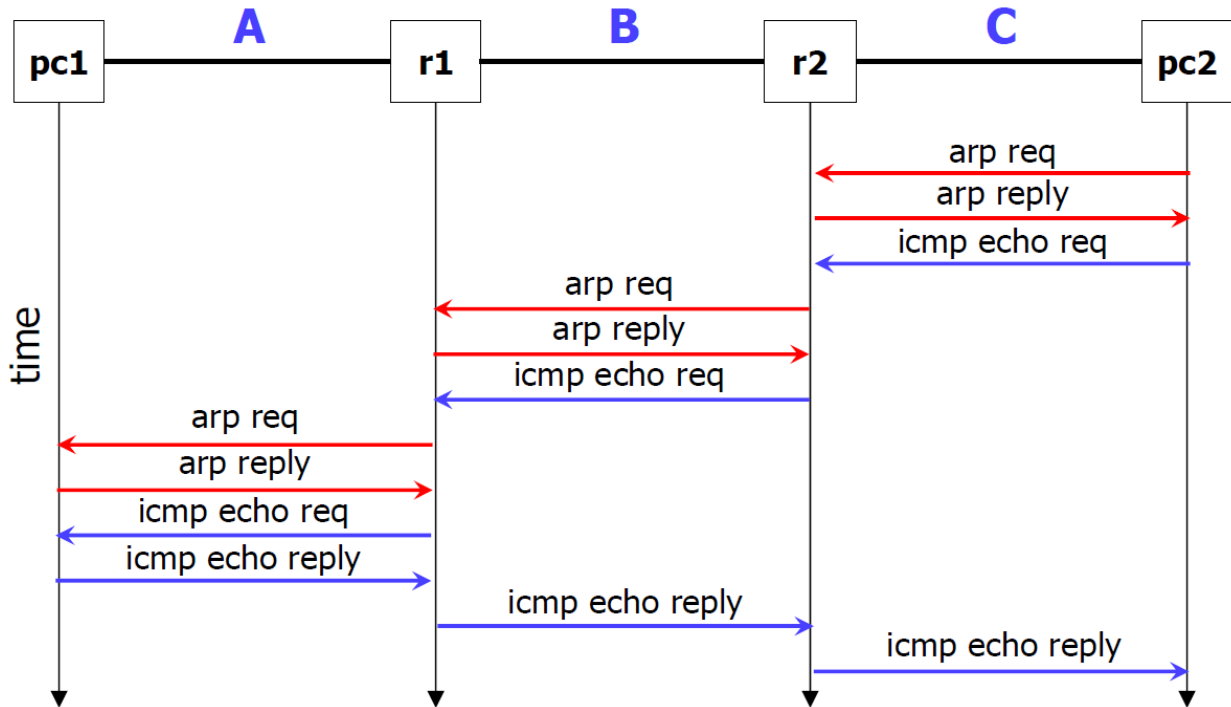
**Q4.2** : Observer la séquence de messages capturés par tcpdump dans r2, r1 et pc1

...

...

...  
...

Voici ce qui se passe dans le réseau.



## 5. Protocole RIPv2

[http://www.netkit.org/netkit-labs/netkit-labs\\_basic-topics/netkit-lab\\_rip/netkit-lab\\_rip.pdf](http://www.netkit.org/netkit-labs/netkit-labs_basic-topics/netkit-lab_rip/netkit-lab_rip.pdf)

RIPv2 : Protocole de routage de la famille Distance Vector. Nous allons utiliser Zebra (a GNU project).

Exemple de configuration Zebra (ripd.conf) :

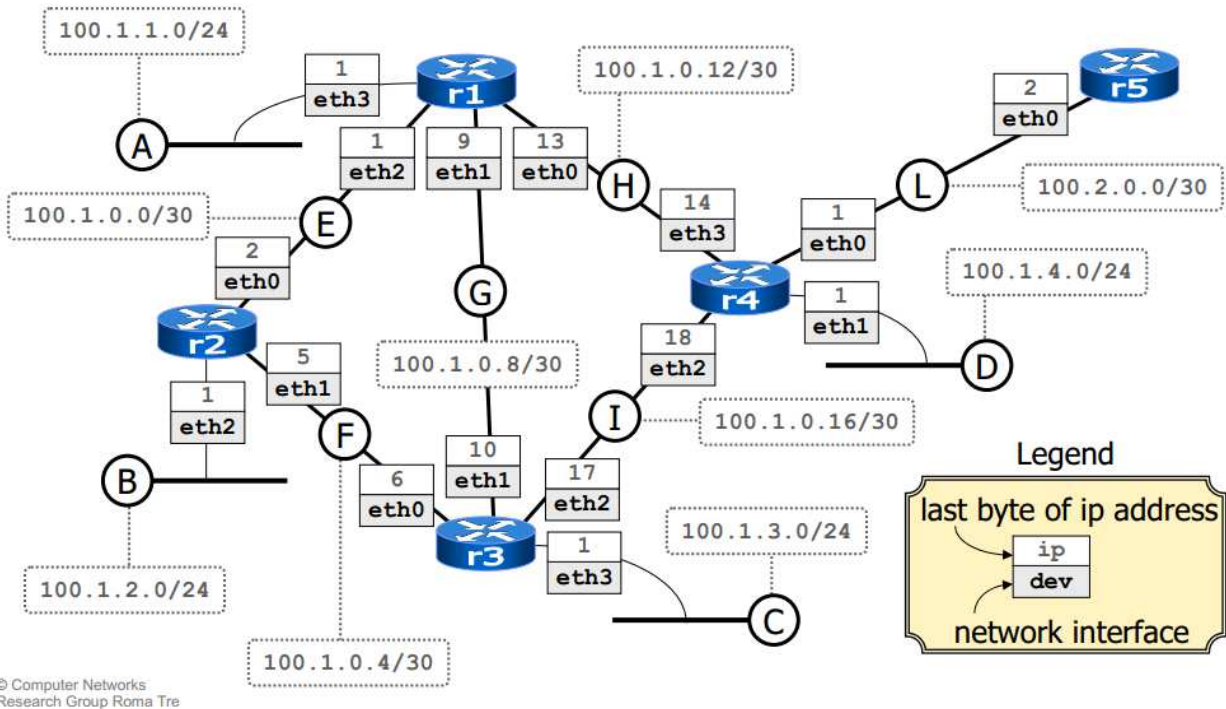
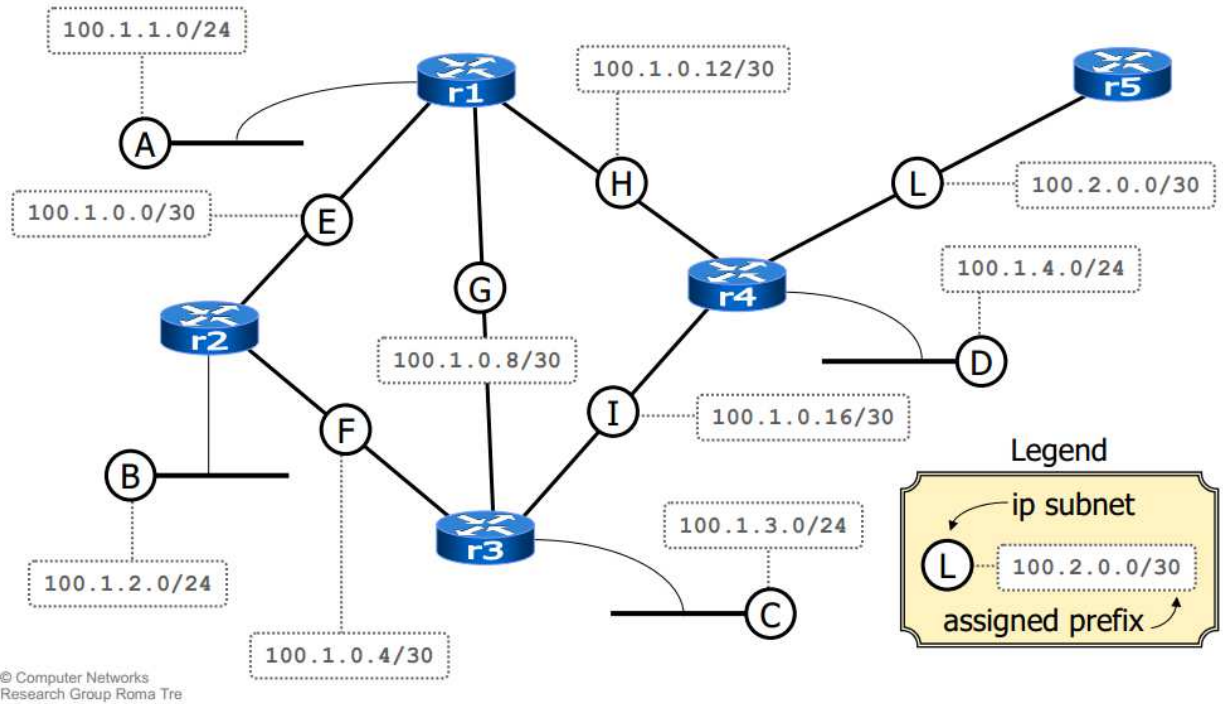
```
virtual machine
pc1:/etc/zebra# cat ripd.conf
!
hostname ripd
password root
enable password root
!
router rip
redistribute connected
network 100.1.0.0/16
!
log file /var/log/zebra/ripd.log
pc1:/etc/zebra#
```

talk rip on some interface

redistribute to rip neighbors information about all directly connected subnets

send rip multicast packets to interfaces falling into this prefix

Nous allons créer le scénario suivant (la configuration des subnets est dans la 1ère figure, celle des interfaces dans la 2ème figure) :



Nous allons utiliser le laboratoire disponible ici :

[http://www.netkit.org/netkit-labs/netkit-labs\\_basic-topics/netkit-lab\\_rip/netkit-lab\\_rip.tar.gz](http://www.netkit.org/netkit-labs/netkit-labs_basic-topics/netkit-lab_rip/netkit-lab_rip.tar.gz)

Sur la machine réelle utiliser les commandes :

```
cd netkit-lab_rip
```

```
lstart
```

La configuration de ce laboratoire est telle que :

- 5 virtual host (les 5 routeurs r1 ... r5) sont créés et connectés aux domaines de collision indiqués dans les figures en haut.
- Pour chaque routeur:
  1. Les interfaces réseau sont configurées automatiquement
  2. Les fichiers de configurations /etc/zebra/daemons, /etc/zebra/zebra.conf, et /etc/zebra/ripd.conf sont à jour.
- Le démon de routage Zebra **n'est pas** démarré automatiquement.

On se met dans r4, et on exécute la commande **ping 100.1.0.13**, pour tester la connectivité.

Toujours dans r4, essayez maintenant **ping 100.1.2.1**

**Q5.1 : Résultat : ...**

Pourquoi vous avez eu ce résultat ? Examinez le tableau de routage dans r4.

**Réponse : ...**

Nous allons donc faire démarrer les démons de routage sur chaque nœud, excepté r5.

Sur chaque routeur r1 ... r4, on exécute donc :

```
# /etc/init.d. init.d/zebra start /zebra start
```

**Q5.2 : Résultat: ...**

On se met à nouveau dans r4, et on exécute la commande **ping 100.1.2.1**

**Q5.3 : Qu'est-ce qu'on peut observer maintenant ?**

...

Pourquoi ? Examinez à nouveau le tableau de routage dans r4.

Pour voir le format des paquets RIPv2 qui sont échangés dans le réseau, il suffit d'utiliser tcpdump.

Toujours en r4, exécuter la commande : `tcpdump -i eth2 -v -n -s 1518`

Options :

-v = display packet details (enable full protocol decoding)

-n = don't resolve numbers to names

-s 1518 = sniff entire Ethernet packets (by default, only the first 68 bytes are captured)

Commande traceroute :

Toujours en r4, exécuter la commande : `traceroute --> r4:~# traceroute 100.1.2.1`

**Q5.4 : Résultat :**

...

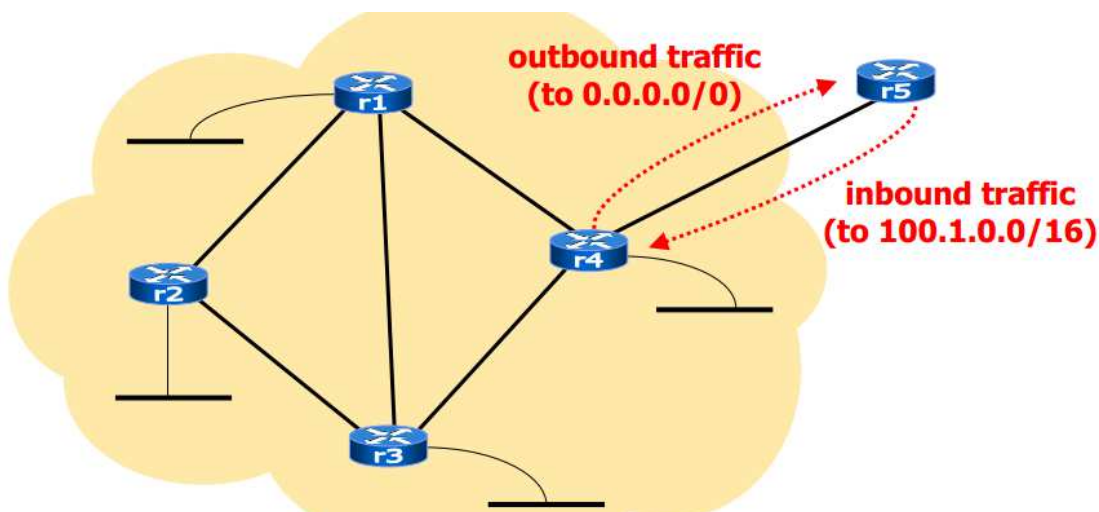
Nous voulons maintenant inspecter le tableau de routage RIP. Pour cela, nous allons nous mettre en r4, et nous exécutons la commande : `telnet localhost ripd`

Le mot de passe est : **zebra**

Avec la commande `show ip rip` nous pouvons regarder le tableau de routage RIP (chaque ligne avec les informations Network, Next Hop, Metric, From et Time)

Notre réseau est dit un « Stub Network », c'est-à-dire qu'il a seulement une connexion vers un routeur extérieur (r5). Par conséquent, une route statique est suffisante pour se connecter au reste d'Internet.

**Q5.5 :** Configurez donc une route statique en r5 (à l'aide de la commande "route") pour permettre au trafic « inbound » (voire la figure suivante) d'entrer dans notre réseau.



Réponse : en r5, ...

Testez, toujours en r5, avec un ping, et ensuite avec un traceroute, vers 100.1.2.1

Il faut maintenant configurer les autres routeurs (r1... r4) pour que le trafic direct vers Internet (« outbound traffic », dans la figure en haut) puisse sortir à travers r5.

**Q5.6** : Pour cela, nous nous mettons en r4 et ajoutons tout d'abord une route par default vers r5.

Solution : ...

Ensuite, il faut propager, à l'aide du protocole RIPv2, cette information vers les autres 3 routeurs (r1, r2 et r3), pour qu'ils puissent automatiquement découvrir comment se connecter à Internet.

Il faut donc se mettre en r4 et se connecter au démon ripd : **telnet localhost ripd** (Le mot de passe est toujours : **zebra**)

Ensuite on va le configurer : il faut utiliser la commande **enable** pour gagner les privilèges d'administrateurs sur ripd (Le mot de passe est toujours : **zebra**).

Ensuite :

```
ripd# configure terminal
```

```
ripd(config)# router rip
```

```
ripd(config-router)# route 0.0.0.0/0
```

```
ripd(config-router)# quit
```

```
ripd(config)# quit
```

```
ripd# disable
```

```
ripd> exit
```

Si on attend un petit peu, la nouvelle route aura été injectée dans le réseau. Par exemple, se mettre dans r1 et utiliser la commande **route** pour vérifier.

Essayez, par exemple en r1, un ping vers n'importe quelle destination (même de destinations « inexistantes ») **ping 193.204.161.1**, et vérifiez que r5 est bien en train de recevoir les ICMP echo requests, avec la commande **tcpdump -i eth0 i eth0 -n -s 1518**

Maintenant, on va voir un scénario dynamique, où un lien de notre réseau « tombe en panne ». Pour cela, mettez-vous en r1 et exécutez tout d'abord **traceroute 100.1.0.10**. Le trafic passe à travers eth1.

**Q5.7** : Maintenant on va éteindre cette interface (toujours en r1), avec la commande ifconfig : ...

Ré-essayez tout de suite, toujours en r1, la commande **traceroute 100.1.0.10**

Ou est-ce que les paquets passent-ils dans cette phase transitoire ?

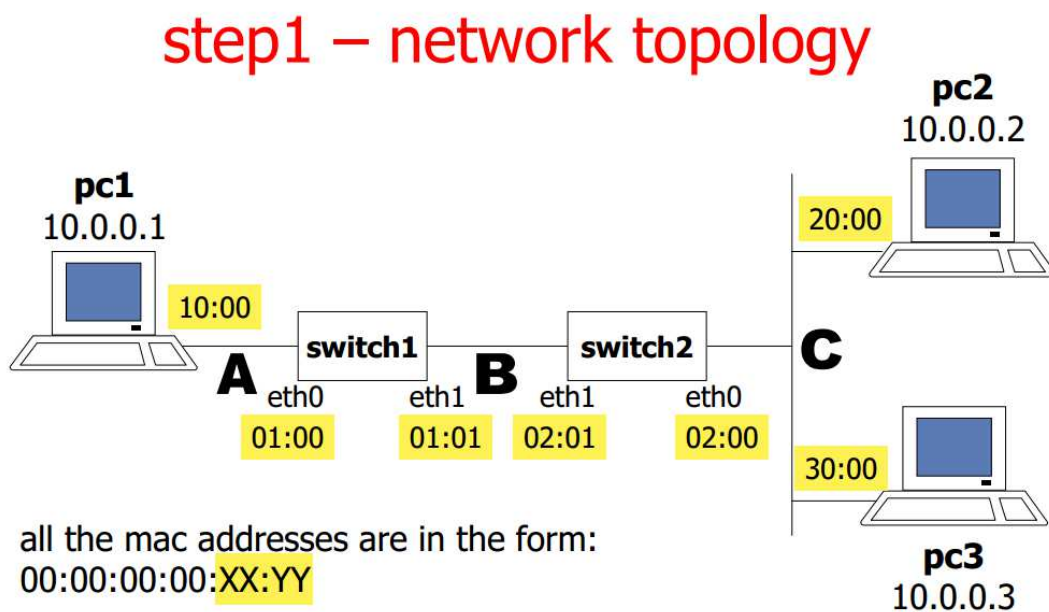
Réponse : ...

Si on attend un petit peu, le réseau (grâce au protocole RIPv2) aura découvert des nouveaux chemins utilisables. Essayez à nouveau en r1 **traceroute 100.1.0.10**, et regardez le nouveau tableau de routage en r1.

## 6. Bridging

[http://www.netkit.org/netkit-labs/netkit-labs\\_advanced-topics/netkit-labs\\_bridging/netkit-lab\\_two-switches.pdf](http://www.netkit.org/netkit-labs/netkit-labs_advanced-topics/netkit-labs_bridging/netkit-lab_two-switches.pdf)

Topologie du réseau :



Nous allons utiliser le laboratoire disponible ici :

[http://www.netkit.org/netkit-labs/netkit-labs\\_advanced-topics/netkit-labs\\_bridging/netkit-lab\\_two-switches.tar.gz](http://www.netkit.org/netkit-labs/netkit-labs_advanced-topics/netkit-labs_bridging/netkit-lab_two-switches.tar.gz)

Sur la machine réelle utiliser les commandes :

```
cd netkit-lab_two-switches
```

```
lstart
```



Dans ce laboratoire :

- 3 machines virtuelles : pc1, pc2 et pc3
- 2 machines virtuelles : les 2 switch, configurés automatiquement en tant que switch.
- Toutes les interfaces réseau sont déjà configurées.

Rappel :

Toutes les interfaces réseau Ethernet ont une adresse MAC de 6 octets :

- Les 3 premiers représentent le "Organizationally Unique Identifier (OUI)", qui identifie le vendeur de la carte (par exemple, Asustek)
- Les 3 derniers octets représentent le serial number de l'interface.

Dans Netkit, les adresses MAC sont générées automatiquement pour les interfaces.

L'adresse MAC d'une interface (virtuelle, dans ce cas) peut être changée avec la commande `ifconfig`.

Par exemple, mettez-vous dans la machine « switch1 » :

```
switch1:~# ifconfig eth0 up
```

```
switch1:~# ifconfig eth0 hw ether 00:00:00:00:01:00
```

```
switch1:~# ifconfig eth0
```

Regardez le champ « HWaddr » avant et après la commande `ifconfig`.

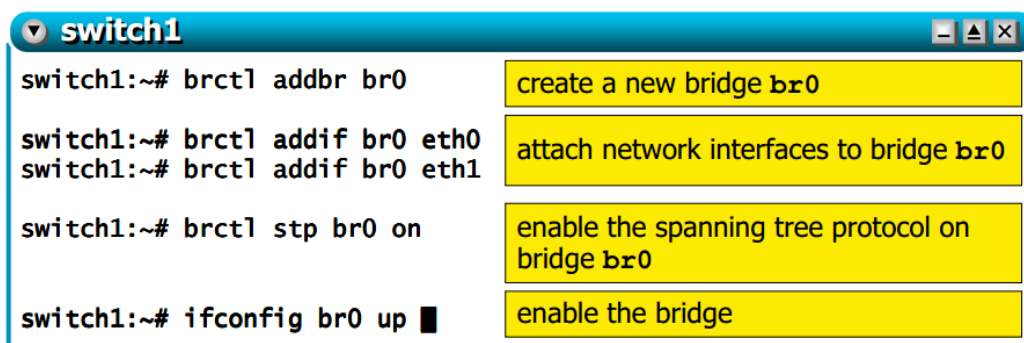
Note importante :

- L'adresse MAC de l'interface doit être forcée après la commande **`ifconfig eth0 up`**, car par défaut cette commande fait un reset des paramètres par défaut.
- Comme le switch est un device de niveau 2, il n'y a aucun besoin de configurer des adresses IP pour ces interfaces

La commande **`brctl`** permet de configurer et vérifier la configuration d'un switch.

Toujours en switch1, exécutez la commande **`brctl show`**.

Ensuite, exécutez les commandes suivantes (sur les 2 switches !) :



```
switch1:~# brctl addbr br0
switch1:~# brctl addif br0 eth0
switch1:~# brctl addif br0 eth1
switch1:~# brctl stp br0 on
switch1:~# ifconfig br0 up
```

create a new bridge <code>br0</code>
attach network interfaces to bridge <code>br0</code>
enable the spanning tree protocol on bridge <code>br0</code>
enable the bridge

- Il faut noter qu'une machine virtuelle peut avoir plusieurs processus "bridge" actifs (sur différentes interfaces réseau)
- Une fois défini, le bridge (ici, br0) est visible comme une interface réseau qui doit être configurée pour fonctionner.

Regardons maintenant les tableaux de ces switch (Source Address Tables).

Si les 3 pcs ne génèrent pas de trafic, alors ces tableaux contiennent seulement des informations locales.

Exécutez la commande : **brctl showmacs br0** sur switch1 et switch2

**Q6.1** : Vérifiez le résultat.

Les portes (= interfaces) sont numérotées selon le standard 802.1d.

Exécutez la commande : **brctl showstp br0** sur switch1 et switch2

Evolution des *address tables* :

Exécutez dans pc3 : **tcpdump -e -q**

... et générez entretemps du trafic entre pc2 et pc3 à l'aide de la commande ping.

**Q6.2** : Observez les résultats de la commande tcpdump.

Exécutez à nouveau la commande : **brctl showmacs br0** sur switch1 et switch2

**Q6.3** : Vérifiez le résultat et répondez aux questions suivantes :

Est-ce que switch1 connaît la position de pc1 ? de pc2 ? et de pc3?

Est-ce que switch2 connaît la position de pc1 ? de pc2 ? et de pc3?

Le filtering :

Nous allons vider les tableaux d'addressing en définissant le lifetime (ageing) de ces entries à 10 seconds :

Dans les 2 switch, utiliser la commande : **brctl setageing br0 10**

Après 10 secondes de "silence", seulement les interfaces locales de chaque switch seront connues.

Nous allons répéter le même test du Step 6 (avec cette fois un tcpdump en pc1, et un ping entre pc2 et pc3), mais cette fois avec 3 secondes d'espace entre les ping : **ping -i 3 10.0.0.3**

**Q6.4** : Quels paquets vous voyez atteindre pc1 ? Pourquoi ?

Répétez à nouveau le même test, mais avant, configurez les 2 switch avec la commande : **brctl setageing br0 1**

**Q6.5** : Quels paquets vous voyez arriver à pc1 ? Pourquoi ?