TP Réseau : Virtual LAN (VLAN)

Travail préparatoire **2**

Installation du TP 2

La maquette 2

Virtual LAN (VLAN) 3

Construction des VLAN 3

Le standard IEEE 802.1Q 3

GNU/Linux 4

Le mode pont 4

VLAN 5

Travail demandé **5**

Switch classique 5

VLAN 6

Les TP d'acquisition des fondamentaux visent à construire un socle de connaissances de base, à appréhender des concepts, des notions et des modèles qui sont fondamentaux. Ce sont des étapes indispensables pour aborder d'autres apprentissages. Les TP sont conduits de manière fortement guidée pour vous placer le plus souvent dans une situation de découverte et d'apprentissage.

Objectifs

Les objectifs de ce TP sont de découvrir le fonctionnement des VLAN sous *GNU/Linux*.

TP Réseau : Virtual LAN (VLAN)

Travail préparatoire

Installation du TP

Le TP3 est disponible dans l'archive `/home/user/sujets-tp/tp4-vlan.tgz` :

```
host> cd /home/user/
host> tar zxvf sujets-tp/tp4-vlan.tgz
host> cd /home/user/tp4-vlan
host> lstart -s
```

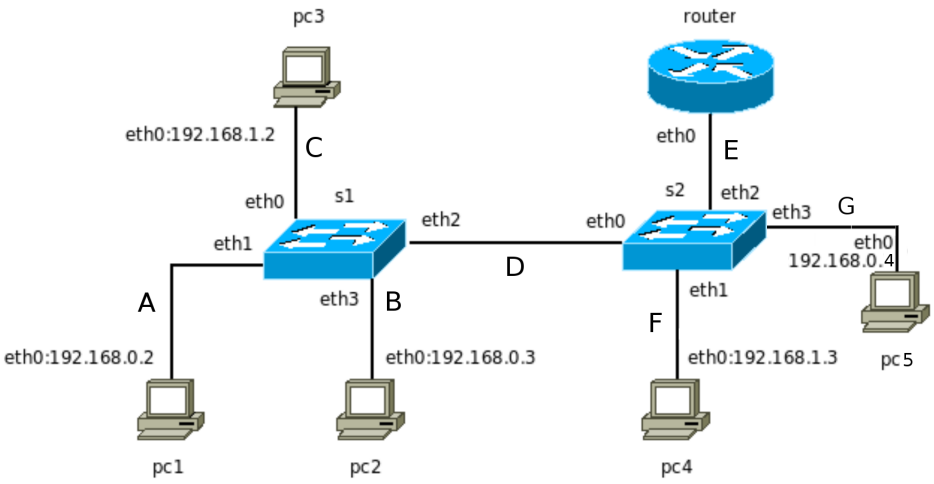
La maquette

Dans ce TP, la maquette NetKit est la suivante :

```
pc1[0]="A"
pc2[0]="B"
pc3[0]="C"
pc4[0]="F"
pc5[0]="F"

s1[0]="C"
s1[1]="A"
s1[2]="D"
s1[3]="B"

s2[0]="D"
s2[1]="F"
s2[2]="E"

router[0]="E"
```

Le fichier lab.conf

Ce qui donne l'architecture suivante :



Virtual LAN (VLAN)

Un VLAN ou réseau local virtuel est un réseau informatique logique indépendant. En configurant un commutateur (*switch*), il est possible de créer des réseaux dits « virtuels » au sein d'un LAN.

Plusieurs VLANs peuvent coexister sur un même commutateur réseau. Pour *Ethernet*, un VLAN est un domaine de diffusion (*broadcast domain*).

Les VLANs permettent la segmentation des réseaux ce qui permettra d'augmenter ou d'améliorer les performances (débit, bande passante, sécurité, ...).

Construction des VLAN

VLAN par port (*Port-based VLAN*) : On affecte chaque port du commutateur à un VLAN. En cas de déplacement d'une machine, il suffit d'affecter (manuellement) son VLAN au nouveau port.

VLAN par adresse MAC (*MAC address-based VLAN*) : Chaque commutateur maintient une table adresse MAC ←→ VLAN. Il faut les initialiser (solution : VLAN par défaut). Le commutateur détermine le VLAN de chaque trame à partir de l'adresse MAC source ou destination. Le déplacement d'une machine est possible et transparent.

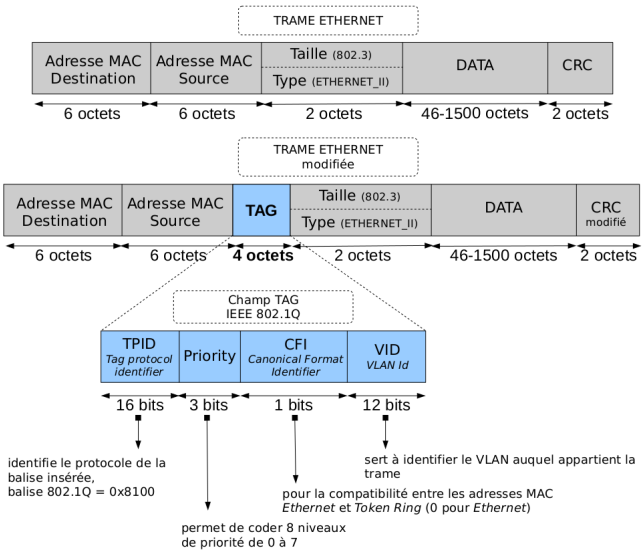
VLAN par adresse de niveau 3 : On affecte une adresse de niveau 3 à un VLAN. L'appartenance d'une trame à un VLAN est alors déterminée par l'adresse de couche 3 (IP par exemple) qu'elle contient (le commutateur doit donc accéder à ces informations). Cela provoque un fonctionnement moins rapide que les VLANs par port ou par MAC. Quand on utilise le protocole IP on parle souvent de VLAN par sous-réseau.

Le standard IEEE 802.1Q

Il permet de modifier la trame *Ethernet* au niveau de la sous-couche MAC (la couche 2 du modèle OSI) afin de fournir un mécanisme d'encapsulation très répandu et implanté dans de nombreux équipements de marques différentes. Il permet de propager plusieurs VLANs sur un même lien physique (*trunk*).

Le terme *trunk* indique un lien de réseau supportant des VLANs multiples entre 2 commutateurs ou entre un commutateur et un routeur.

802.1Q définit le contenu de la balise de VLAN (*VLAN tag*) avec laquelle on complète l'en-tête de la trame *Ethernet*.



GNU/Linux

Le mode pont

Il est possible d'ajouter à une machine *GNU/Linux* la fonction de commutateur *Ethernet* aussi appelée **pont** ou *bridge*. La machine doit possèder plusieurs cartes réseau qui seront utilisées comme des ports du commutateur. Ce pont fonctionnera comme un **commutateur *Ethernet* classique (*switch*)**.

La définition d'un pont sous *GNU/Linux* se fait en utilisant la commande `brctl`.

Il faut commencer par créer une **interface pont** `sw1` :

```
s1:~# brctl addbr sw1
```

L'interface pour le pont apparaît comme une nouvelle interface, un peu comme `eth0` ou `eth1`. Elle n'existe pas physiquement sur l'ordinateur, mais c'est une interface virtuelle qui prend juste les trames depuis une interface physique et de manière transparente les retransmet vers l'autre.

Ensuite, il faut ajouter les interfaces physiques qui vont être « pontées » :

```
s1:~# brctl addif sw1 eth1
s1:~# brctl addif sw1 eth3
```

Et pour finir, on active l'**interface pont** `sw1` :

```
s1:~# ifconfig sw1 up

s1:~# brctl show sw1
bridge name    bridge id              STP enabled    interfaces
sw1            8000.9a496fe4ea42      no             eth1
                                                     eth3
```

Maintenant, les trames sont automatiquement rediffusées sur les autres ports. Il est donc possible de faire un *ping* entre **pc1** et **pc2**.

```
pc1:~# ifconfig
eth0      Link encap:Ethernet HWaddr 6e:5f:98:37:0c:07
          inet addr:192.168.0.2 Bcast:192.168.0.255 Mask:255.255.255.0
pc2:~# ifconfig
eth0      Link encap:Ethernet HWaddr 2e:18:cc:d4:8c:e7
          inet addr:192.168.0.3 Bcast:192.168.0.255 Mask:255.255.255.0

pc1:~# ping -c 1 192.168.0.3
PING 192.168.0.3 (192.168.0.3) 56(84) bytes of data.
64 bytes from 192.168.0.3: icmp_seq=1 ttl=64 time=0.139 ms

--- 192.168.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.139/0.139/0.139/0.000 ms

pc2:~# ping -c 1 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=0.141 ms

--- 192.168.0.2 ping statistics ---
```

```
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.141/0.141/0.141/0.000 ms
```

On peut voir que le mécanisme *bridge* de *GNU/Linux* possède une table d'association port / adresse MAC :

```
s1:~# brctl showmacs sw1
port no mac addr            is local?     ageing timer
  2     2e:18:cc:d4:8c:e7     no              2.88
  1     6e:5f:98:37:0c:07     no              2.88
  2     9a:49:6f:e4:ea:42     yes             0.00
  1     b6:ab:81:26:f7:ed     yes             0.00
```

Il existe d'autres options pour la commande `brctl`. Il est par exemple possible de définir le temps de vieillissement (*ageing time*) de l'adresse MAC en secondes. À l'expiration de ce temps, le pont supprimera cette adresse s'il n'a pas « vu » de trames pour elle. Par exemple :

```
s1:~# brctl setageing sw1 30
```

Pour d'autres options, faire un `man brctl`.

Pour supprimer l'**interface pont** `sw1`, il suffit de la désactiver puis de la supprimer comme ceci :

```
s1:~# ifconfig sw1 down
s1:~# brctl delbr sw1
```

VLAN

Sous *GNU/Linux*, il est possible de créer des VLANs en utilisant la commande `vconfig`.

La syntaxe de la commande pour créer un VLAN est la suivante : `vconfig add interface VLAN-ID`. Cela va créer une interface nommée "`interface.VLAN-ID`" qu'il faudra ensuite activer.

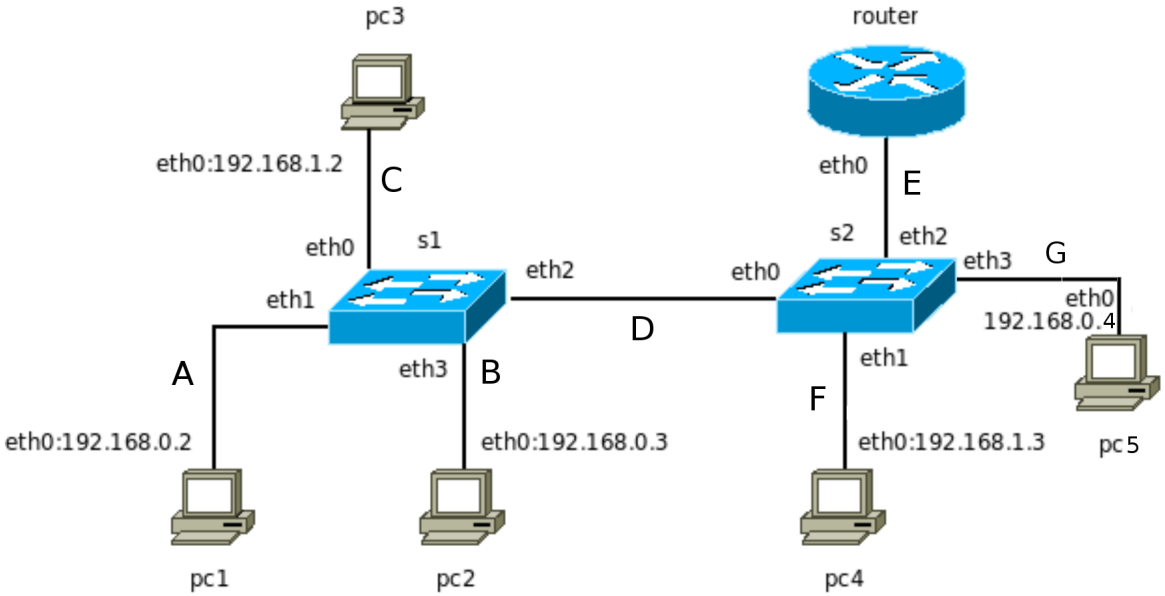
Par exemple :

```
s1:~# vconfig add eth2 100
s1:~# ifconfig eth2.100 up
```

Travail demandé

Switch classique

L'architecture réseau est la suivante :



Vérifier que les paramètres IP des interfaces de l'ensemble des machines sont bien configurés.

Question 1. Créer une interface pont `sw1` sur le switch **s1**.

Question 2. Ajouter au pont `sw1` les interfaces pour relier **pc1**, **pc2** et **s2**. Activer l'interface pont `sw1`.

Question 3. Créer une interface pont `sw2` sur le switch **s2**.

Question 4. Ajouter au pont `sw2` les interfaces pour relier **pc5** et **s1**. Activer l'interface pont `sw2`.

Question 5. Tester la connectivité entre les machines **pc1**, **pc2** et **pc5**.

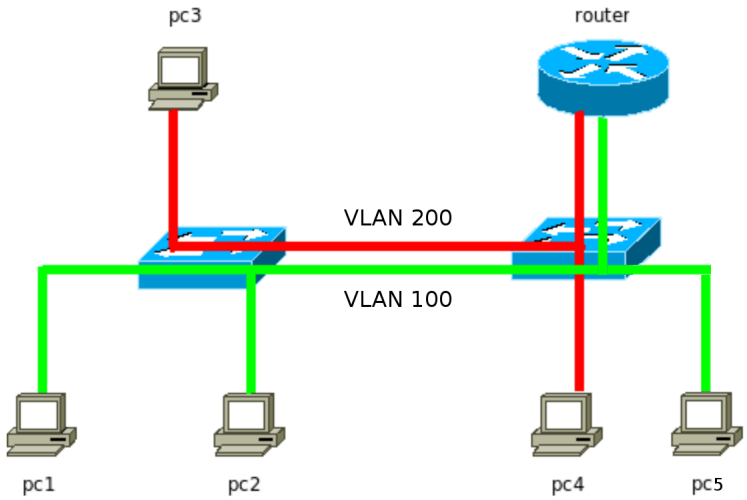
La connectivité avec le **router** n'est pas possible car celui-ci est configuré pour appartenir à un VLAN. En effet, ses trames Ethernet II sont modifiées par le protocole **802.1Q**.

Question 6. Désactiver l'interface pont `sw1` et supprimer la. Désactiver l'interface pont `sw2` et supprimer la.

VLAN

On sépare maintenant les réseaux `192.168.0.0/24` et `192.168.1.0/24`. Nous allons séparer chacun de ces réseaux sur un VLAN : le VLAN numéro 100 pour `192.168.0.0/24` et le VLAN numéro 200 pour `192.168.1.0/24`. Il faudra aussi faire passer les trames par la liaison entre **s1** et **s2** (un *trunk*).

Le schéma ci-dessous présente l'organisation logique que l'on doit obtenir :



Question 7. Créer sur **s1** le VLAN `100` pour l'interface `eth2`. Activer l'interface `eth2.100` sur le switch **s1**.

Question 8. Créer sur **s1** le VLAN `200` pour l'interface `eth2`. Activer l'interface `eth2.200` sur le switch **s1**.

Question 9. Créer une interface pont `vlan100` sur le switch **s1**.

Question 10. Ajouter au pont `vlan100` les interfaces pour relier **pc1**, **pc2** et **s2**. Activer l'interface pont `vlan100` sur le switch **s1**.

Question 11. Créer une interface pont `vlan200` sur le switch **s1**.

Question 12. Ajouter au pont `vlan200` les interfaces pour relier **pc3** et **s2**. Activer l'interface pont `vlan200` sur le switch **s1**.

Question 13. Créer sur **s2** le VLAN `100` pour l'interface `eth0` et `eth2`. Activer l'interface `eth0.100` et `eth2.100` sur le switch **s2**.

Question 14. Créer sur **s2** le VLAN `200` pour l'interface `eth0` et `eth2`. Activer l'interface `eth0.200` et `eth2.200` sur le switch **s2**.

Question 15. Créer une interface pont `vlan100` sur le switch **s2**.

Question 16. Ajouter au pont `vlan100` les interfaces pour relier **pc5**, **routeur** et **s1**. Activer l'interface pont `vlan100` sur le switch **s2**.

Question 17. Créer une interface pont `vlan200` sur le switch **s2**.

Question 18. Ajouter au pont `vlan200` les interfaces pour relier **pc4**, **router** et **s1**. Activer l'interface pont `vlan200` sur le switch **s2**.

Question 19. Tester la connectivité entre toutes les machines (**pc1**, **pc2** ...).

Capturer avec `wireshark` les échanges de trames du domaine `D`.

```
host> vdump D | wireshark -i - -k &
```

Question 20. Identifier les VLAN ID dans les trames *Ethernet*.