université de BORDEAUX

TP8 - VLAN

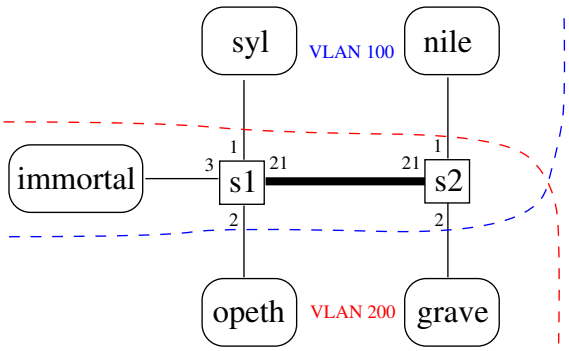
1 Avant de commencer

1. Lancez le script de démarrage

   ```
   /net/ens/qemunet/qemunet.sh -x -t /net/cremi/sathibau/Reseau/TPvlan/topology -y -v
   ```

   L'option `-y` sert à lancer les consoles d'aministration des switch. L'option `-v` quant à elle sert à activer le support des VLAN.

2. Regardez le contenu de `/net/stockage/aguermou/AR/TP/4/topology` et observez la topologie de notre réseau. Elle est présentée dans la figure ci-dessous :



   qui indique notamment les numéros de ports sur lesquels sont branchés les différentes machines et les switches entre eux.

   Vous pouvez disposer les nombreuses fenêtres qui s'ouvre de la même façon que cette figure pour vous y retrouver. En plus des 5 fenêtres Linux habituelle, vous avez les 2 fenêtres correspondant à la console d'administration des switches s1 et s2 du réseau virtuel. Pour l'instant, rien n'est configuré, seuls les câbles sont branchés entre les machines et les switches.

   Note : si par mégarde vous fermez la fenêtre d'un switch, on peut la relancer à la main, il suffit de relancer la commande vdeterm qui s'est affichée dans le terminal de départ, du genre :

   ```
   rxvt -bg Black -fg White -title s1 -e vdeterm /tmp/qemunet-$USER-*/s1.mgmt
   ```

2 Premiers pas

Nous allons nous concentrer dans un premier temps sur les machines `immortal`, `opeth` et `syl`. En consultant la description de la topologie, nous pouvons constater que l'objectif

est que la machine `syl` (resp. `opeth`) fasse partie du VLAN 100 (resp. 200), et que la machine `immortal` ait accès aux deux VLANs. Nous allons donc configurer un premier réseau IP contenant `immortal` et `syl` et un second contenant `immortal` et `opeth`, `immortal` faisant office de passerelle entre les deux réseaux/VLAN.

1. Configurer `syl` et `opeth` de telle sorte qu'elles soient dans deux réseaux IP différents.
2. Lancer un `tcpdump` sur `opeth`, puis lancer sur `syl` un ping vers une autre adresse de son propre réseau (même s'il n'y a pas encore de machine à cette adresse, on verra passer les ARP). Même si les réseaux sont différents, les deux machines étant pour le moment sur le même réseau Ethernet, `opeth` voit passer les trames de `syl` ! (du moins les broadcasts utilisés pour ARP).
3. On va donc commencer par isoler ces deux machines dans des VLANs différents 100 et 200. On utilise pour cela la console d'administration du switch `s1` pour configurer ces VLANs. Si vous souhaitez avoir la liste des commandes disponibles au niveau de la console d'administration, il suffit de taper la commande `help`. Dans un premier temps, il faut créer les VLAN 100 et 200 à l'aide de la commande `vlan/create` :

   ```
   vlan/create     Num_VLAN (créer le VLAN numéro Num_VLAN)
   ```

   Il faut maintenant préciser que l'on veut placer les ports de nos deux machines à l'intérieur des VLANs. Les numéros de ports sur lesquels sont branchées les machines sont indiqués sur la figure plus haut, ou bien visibles avec `port/print`

   ```
   port/setvlan    Num_PORT Num_VLAN (associer Num_PORT à Num_VLAN en mode untagge
   ```

   Si l'on s'est trompé de numéro de vlan, on peut revenir en arrière en utilisant `port/setvlan Num_PORT 0`

   Relancez le ping, observez que `opeth` ne voit pas passer les trames.
4. Utilisez la commande `vlan/print`, on remarque que les ports sont en mode *untagged* : le trafic de ces ports ne contient pas le tag 802.1q, c'est de l'Ethernet classique. Le switch s'occupe d'ajouter le tag à la volée pour le trafic qui y entre et de l'enlever pour le trafic qui en sort.
5. La configuration d'`immortal` va être différente de celle des autres machines dans le sens où elle va devoir faire partie des deux réseaux IP/VLAN à la fois ! Pour ce faire, il est nécessaire de :
   — Dire à l'interface réseau d'`immortal` qu'elle fait partie des deux VLANs. Ceci peut être fait en exécutant les commandes suivantes :

   ```
   ifconfig eth0 up
   vconfig add eth0 100 # pour ajouter le vlan 100
   vconfig add eth0 200 # pour ajouter le vlan 100
   ```

   (oui, il prévient que c'est une commande obsolète mais pour l'instant elle est courante, et facile à utiliser) Ceci a pour effet de créer deux interfaces réseaux `eth0.100` et `eth0.200` faisant partie des VLANs 100 et 200, que vous pouvez voir dans `ifconfig -a` ou `ip a ls`. `eth0` verra passer toutes les trames avec les tags, tandis que `eth0.100` et `eth0.200` ne verront passer que les trames des VLANs correspondants, sans les tags.

— Configurer les nouvelles interfaces réseau pour qu'elles fassent partie des bons réseaux IP.
`ifconfig eth0.100 ...`
`ifconfig eth0.200 ...`

Si vous voulez que la création et la configuration des interfaces réseaux associées aux VLAN soit faite automatiquement à chaque démarrage, il suffit d'éditer le fichier `/etc/network/interfaces` et d'y ajouter les sections suivantes :

```
#VLAN 100
auto eth0.100
iface eth0.100 inet static
address .../24

#VLAN 200
auto eth0.200
iface eth0.200 inet static
address .../24
```

Puis de rebooter, ou juste relancer par exemple `ifdown eth0.100 ; ifup eth0.100`

Remarque : N'oubliez pas d'activer le mode passerelle sur `immortal` pour qu'elle puisse relayer les paquets allant de `syl` vers `opeth` et inversement.

6. Configurer `syl` et `opeth` de telle sorte qu'`immortal` soit leur passerelle. Cela ne fonctionne cependant pas encore...
7. Il faut en effet encore placer le port switch sur lequel `immortal` est branchée dans les deux VLANs. Pour qu'`immortal` puisse distinguer les deux VLANs, il faut que le port soit en mode *tagged* : le traffic sortant du switch doit conserver son tag 802.1q.

   Pour ce faire, on dispose de la commande suivante dans l'interface d'administration du switch `s1`.

   `vlan/addport  Num_VLAN Num_PORT (associer Num_PORT à Num_VLAN en mode tagged)`

   Vérifier la configuration avec `vlan/print` . Si l'on s'est trompé, on peut utiliser la commande `vlan/delport` pour revenir en arrière. Si par mégarde on avait utilisé `port/setvlan` (et donc en mode tagged=0), on peut revenir en arrière en réutilisant `port/setvlan` pour mettre le port dans le vlan 0. On peut alors utiliser `vlan/addport` pour mettre le port en mode tagged=1.
8. Faites communiquer `opeth` et `syl` en vous mettant en écoute sur `immortal`. Il s'agit d'analyser le traffic passant par `eth0`, `eth0.100` et `eth0.200` avec `tcpdump`. Vous pouvez ajouter l'option `-e` pour voir le tag vlan (dans le cas d'`eth0`) et les MAC source destination, pour vérifier que cela se comporte comme prévu.

3 Configuration avancée

Maintenant qu'`immortal`, `opeth` et `syl` sont configurées correctement et arrivent à communiquer les unes avec les autres, nous allons nous intéresser à la configuration de `nile`, `grave` et du switch `s2`.

1. Configurer `nile` (resp. `grave`) en lui donnant une adresse IP qui est dans le même réseau que `syl` (resp. `opeth`). De plus, il faudra considérer `immortal` comme passerelle par défaut.
2. Configurer le switch `s2` pour que `nile` (resp. `grave`) soit dans le même VLAN que `syl` (resp. `opeth`).
3. Configurez le lien de type *trunk* reliant les switchs `s1` et `s2` pour faire en sorte qu'il transporte le trafic des VLAN 100 et 200. Il s'agit donc juste d'associer le port 21 (où est branché le lien trunk) aux différents VLANs en mode *tagged* pour qu'ils puissent faire le tri. Cette étape se fera sur les consoles d'administration de `s1` et `s2`.

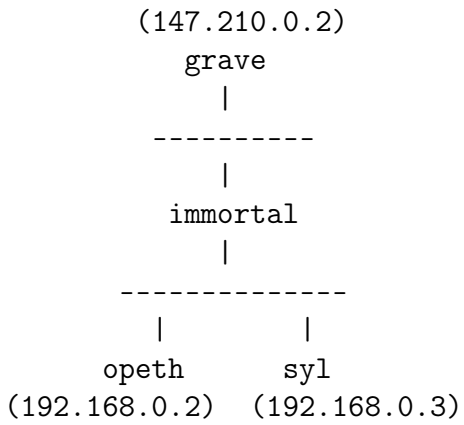
De nouveau, vous pouvez maintenant observer avec `tcpdump` que les VLANs sont bien isolés et les paquets sont donc obligés de passer par le routeur `immortal` qui pourra donc mettre en oeuvre des règles de firewall.

4 (Bonus) Manipulation de paquets avec Scapy

Scapy est un logiciel libre de manipulation de paquets réseau écrit en Python. C'est un outil (interactif) très puissant qui permet en quelques lignes de Python (contre des centaines en langage C) d'implémenter des fonctions réseau de base : *ping*, *nmap* (syn scan), *traceroute*, ... En pratique, Scapy fait deux choses simples : envoyer des paquets et recevoir les réponses.

4.1 Démarrage d'un réseau virtuel

Lancez le réseau virtuel *QemuNet* avec la topologie suivante :
`/net/ens/qemunet/qemunet.sh -x -s /net/ens/qemunet/demo/gw.tgz`



Le réseau est déjà configuré correctement (adresses IP et routage). Lancez `tcpdump -i eth0` sur la passerelle `immortal` afin d'espionner le trafic échangé entre les autres machines. Lancez `netstat -tupl` pour voir quels services (et donc quels ports) sont ouverts sur `opeth` (ou `syl`).

4.2 Prise en main de Scapy

Lancez `scapy3` sur la machine `grave` en tant que *root* (ou *sudoer*). En effet, Scapy court-circuite l'interface de programmation traditionnelle des *Sockets* pour effectuer des manipulations de bas-niveau et il nécessite à ce titre des permissions supplémentaires.

L'invite `>>>` d'un interpréteur *Python3* apparaît. Vous pouvez maintenant programmer en Scapy. Par exemple :

```
$ scapy3
>>> x=IP()
>>> x.show()
>>> exit        # pour quitter scapy
```

Il est également possible d'écrire des programmes Scapy sous forme d'un script Python, qu'il faut enregistrer avec un éditeur de texte, comme nano ou emacs.

```
#!/usr/bin/env python3
import sys
from scapy.all import *

x = IP()
x.show()
```

Vous pouvez ensuite exécuter votre script comme ceci :

```
$ python3 test.py
```

Survolez rapidement la documentation pour découvrir les nombreuses possibiltés de Scapy : https://scapy.readthedocs.io/en/latest/introduction.html

4.3 Ping

Regardez dans le fichier ping.py un exemple d'utilisation de Scapy qui envoie un ping (ICMP) puis récupère la réponse. Essayez-le en recopiant le programme ligne par ligne, ou en faisant un copier/coller.

```
ping = IP(dst='192.168.0.2')/ICMP(type='echo-request')
ping.show()
pong = sr1(ping)
pong.show()
```

4.4 ARP

Rappelez le fonctionnement du protcole ARP. Notez que le protocole ARP ne dispose que de deux opérations : la requête (*Who Has*) et la réponse. On peut alors utiliser ce protocole pour effectuer un *ping* dans le réseau local Ethernet. Il s'agit d'envoyer une requête ARP. Si la machine répond, c'est bien qu'elle est en vie !

Commencez par construire une trame Ethernet avec `Ether()` vers l'"adresse de broadcast `FF:FF:FF:FF:FF:FF` et encapsulez le datagramme `ARP()` à destination de l'adresse IP visée. Pour envoyer et recevoir une trame Ethernet, il faut utiliser la fonction `srp1()` (à la place de la fonction `sr1()` réservé aux paquets IP). On peut aussi utiliser dans cette fonction l'option `timeout=1` pour limiter le temps d'attente à 1 seconde, dans le cas où il n'y a pas de réponse.

4.5 Services UDP : Daytime et Echo

Suivez l'exemple du fichier daytime.py qui envoie un paquet UDP sur le port `daytime` (13) puis récupère et affiche la date envoyée en réponse. Essayez pas à pas.

Vous avez pu remarquer que le service `udp echo` est ouvert (port 7). Testez ce service en envoyant le message 'hello'. Quelle est la réponse ? Expliquez à quoi sert le *padding* dans la réponse ?

4.6 Syn Scan

Pour tester si un service TCP est disponible, il suffit d'essayer de s'y connecter, en envoyant un datagramme TCP à destination du port visé avec le flag SYN. Essayez avec les ports 80 et 81... Dans l'en-tête TCP, il faut mettre le champs `flags="S"` pour SYN. En déduire une méthode pour déterminer si un port donné est ouvert (ou fermé).

Écrivez une fonction qui prend en paramètre une adresse IP et qui imprime la liste des ports ouverts entre 1 et 100. Il s'agit donc de faire la même chose que ce que l'on a fait à la main ci-dessus, dans une boucle for. Pour tester si le flag S (SYN) est présent, il suffit d'utiliser `answer.payload.flags.S`. De même pour A (ACK) ou R (RST).

Pour ouvrir un port supplémentaire sur `opeth`, lancez-y `nc -l -p 42 &` par exemple.

5 Pour aller plus loin

5.1 Traceroute

A l'aide de Scapy, écrire un programme *traceroute*, qui cherche la route vers une destination IP. Le principe consiste à jouer avec le champs TTL (time to leave) de l'en-tête IP en le faisant croitre à partir de 1... pour provoquer une erreur sur les routeurs qui vont rejeter consécutivement ce paquet en renvoyant un message d'erreur ICMP (time to leave exceeded) !

5.2 Une connexion TCP

Essayez d'établir (à la main, donc) une connexion vers le service `echo` mais cette fois-ci en TCP. Il s'agit d'effectuer la poignée de main TCP en trois temps : SYN, SYN/ACK, ACK. Attention, il faudra donc bien sûr récupérer le numéro de séquence, et renvoyer les bons numéro d'acquittement, etc. Ignorez l'apparition de datagrammes RST dans le tcpdump, ils sont produits par le noyau parce que Scapy est fourbe, mais ça ne gênent pas !