Réseaux Logiques

Raphaël-David Lasseri

Cachan Réseau à Normale Sup'

Mardi 26 Novembre 2013

Sommaire

Pourquoi ne pas se satisfaire d'IP ?

En envoyant un paquet IP, on l'adresse à une machine donnée sans pour autant préciser le programme auquel il est déstiné.
Il a donc fallu introduire des protocoles de couche supérieure !

Ce protocole permet la transmission simple de données entres deux programmes ou services définies par le couple (IP,port)

Structure

Source Port Number [16 bits]	Destination Port Number [16 bits
Length(UDP Header + Data) [16 bits]	UDP Checksum [16 bits]
Application Data (Message)	

Application : Envoi rapide de données par petites quantités

- Streaming
- Jeux en lignes
- VoIP
- ...

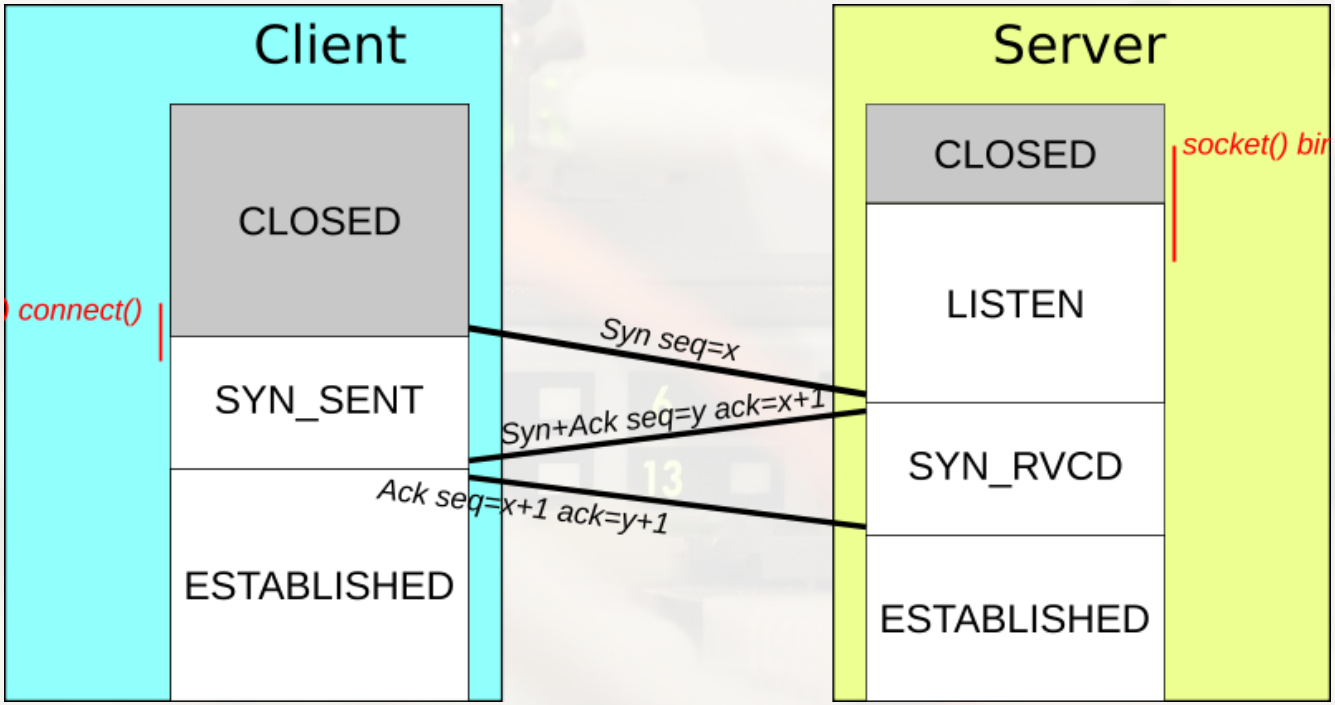
Incovénient majeur : Réception de paquets pouvant arriver dans le désordre et/ou être perdus.

Protocole plus complexe qui permet un échange ordonné des données

Structure

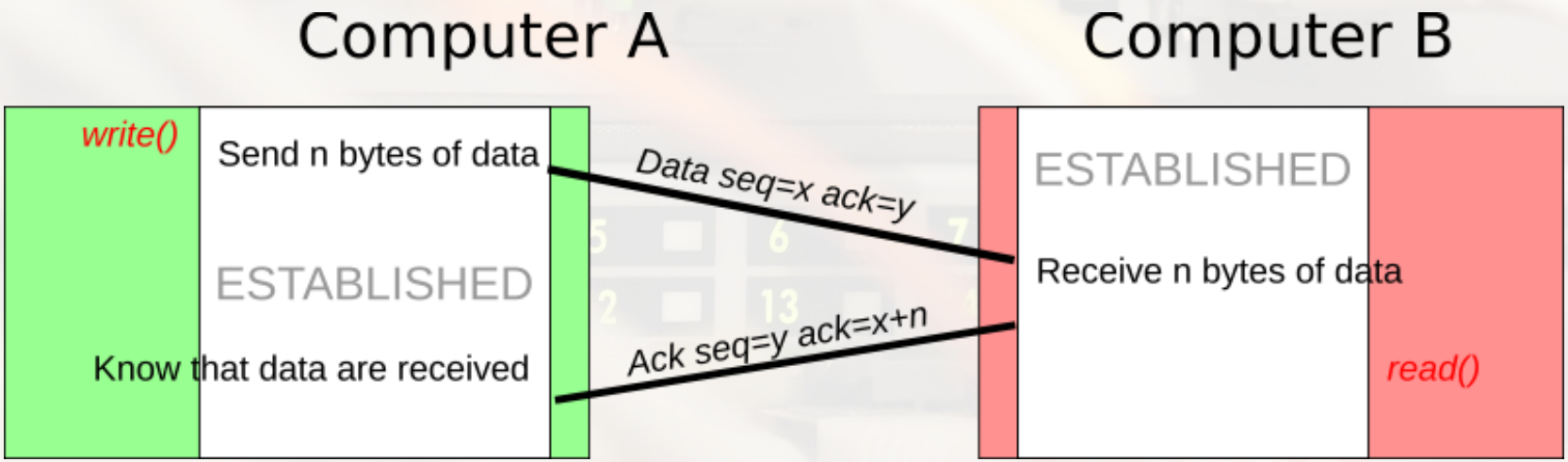
16-bit								32-bit
Source Port								Destination Port
Sequence Number								
Acknowledgement Number (ACK)								
Offset Reserved	U	A	P	R	S	F		Window
Checksum								Urgent Pointer
Options and Padding								

Initiation de la connexion

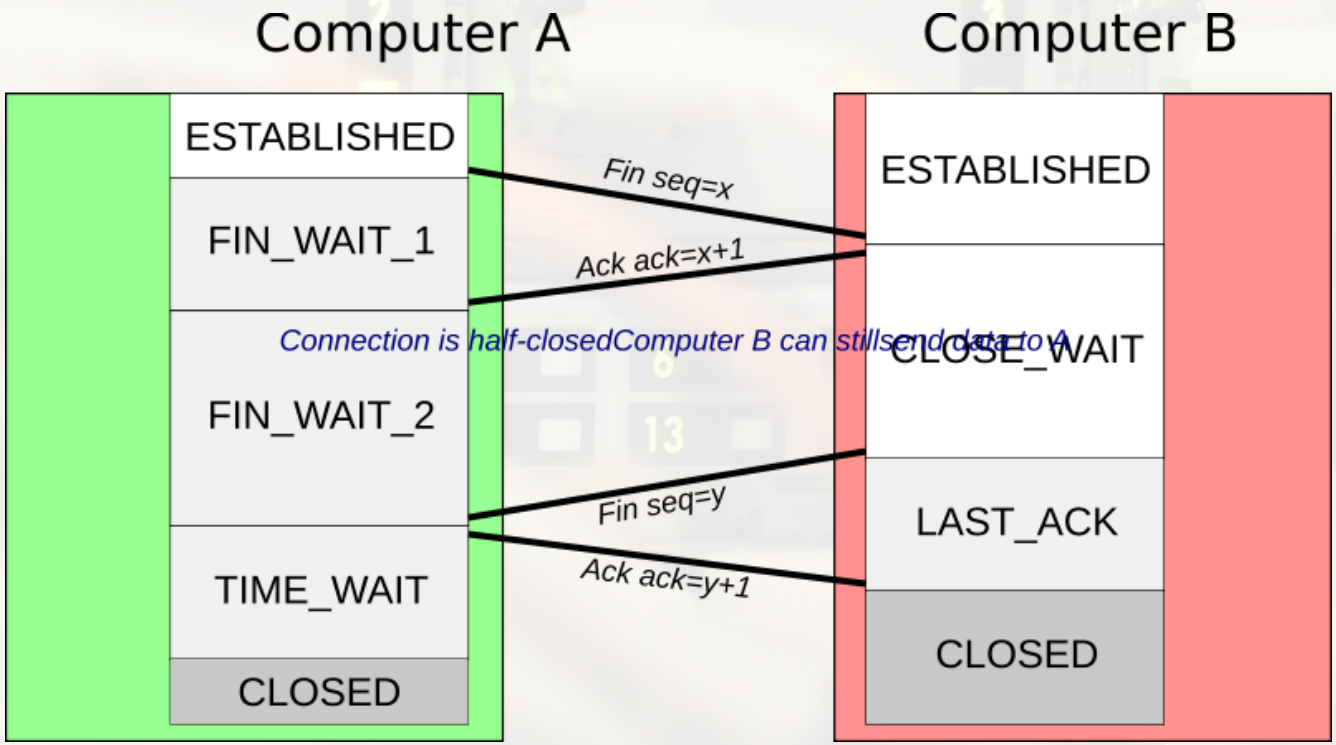


Les "ack" pour acknowledgment permettent de garantir la bonne réception du paquet.

Protocole d'échange avec le serveur



Terminaison de connexion



Quelques Outils

- `tcpdump -i [interface]` Il faut etre sudoer pour pouvoir executer ces commandes
- `Wireshark` Un sniffer de paquets trés utile

Applications

Utilisation de `Scapy` pour forger "à la main" des paquets.

Scapy est une bibliothéque python permettant de génerer ses propres paquets pour mieux en comprendre l'architecture.
Un bref exemple pour génerer un **ping** classique il faut :

```
»Mon_Ping=IP()/ICMP()
```

Le / traduit l'encapsulation

```
»Mon_Ping.dst='ip.de.destination'
»Mon_Ping.src='ip.de.la.source'
```

Pour regarder ce que contient le paquet il suffit d'executer :

```
»Mon_Ping.show()
```

Et enfin pour l'envoyer il suffit de faire :

```
»send(Mon_Ping)
```

Pour pouvoir recevoir une réponse on peut utiliser

```
»rep,non_rep = srp1(Mon_Ping)
```

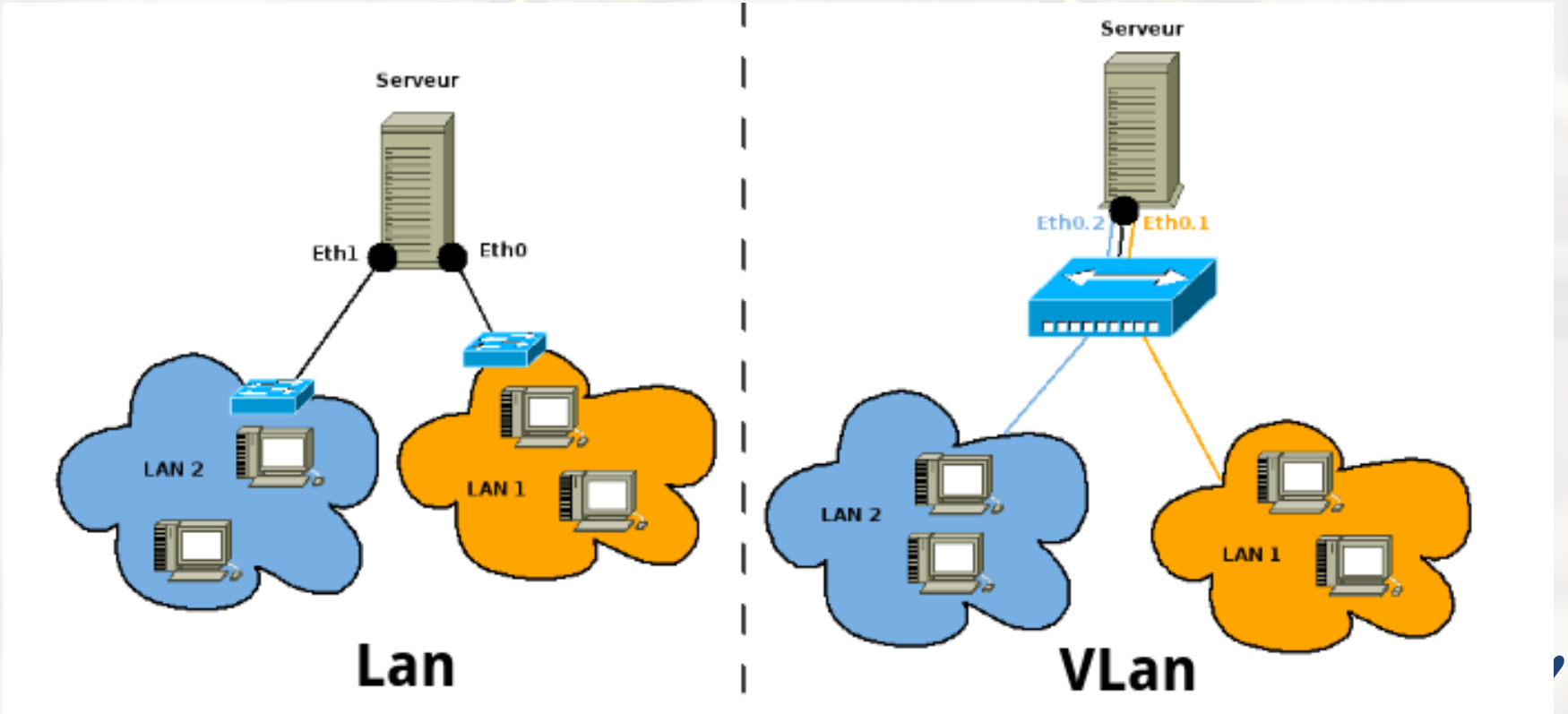
Sommaire

Virtual Local Area Network

C'est l'analogue d'un réseau local, mais sans support physique !
Intéret et exemple : On dispose d'un seul "switch" sur lequel plusieurs machines sont branchés mais on veut que ces machines ne puissent pas communiquer entre elles...
Au Cr@ns on a par exemple besoin de distinguer les

- Vlan Adhérents
- Vlan adm (Administrateur)

Virtual Local Area Network



Les types de VLan's

On peut distinguer :

- VLAN de niveau 1 On definit les plages des ports (physique) distinguant les réseaux
- VLAN de niveau 2 On spécifie les adresses MAC des machines appartenant à chaque VLAN
- VLAN de niveau 3 On spécifie cette fois les IP's des machines

Configuration
Coté Switch

Exemple de configuration des vlan d'un switch du Cr@ns :

```
vlan 1
    name "Adherent"
    untagged 1-4,17,24
    no ip address
exit
vlan 2
    name "Adm"
    tagged 1-4,24
    ip address 10.231.136.112 255.255.255.0
exit
```

Configuration
Coté serveur

Avant toute chose vous aurez besoin du paquet `vlan` et d'activer le protocole 802.1q `modprobe 8021q`
Utilisation de `vconfig`
Pour ajouter une sous interface pour le VLAN 1 on utilise :

```
sudo vconfig add eth0 1
```

L'interface créée s'appelera par convention `eth0.1`
Puis il faut configurer l'interface comme une interface classique avec `ifconfig` ou `ip`
Pour supprimer cette interface il suffit d'executer :

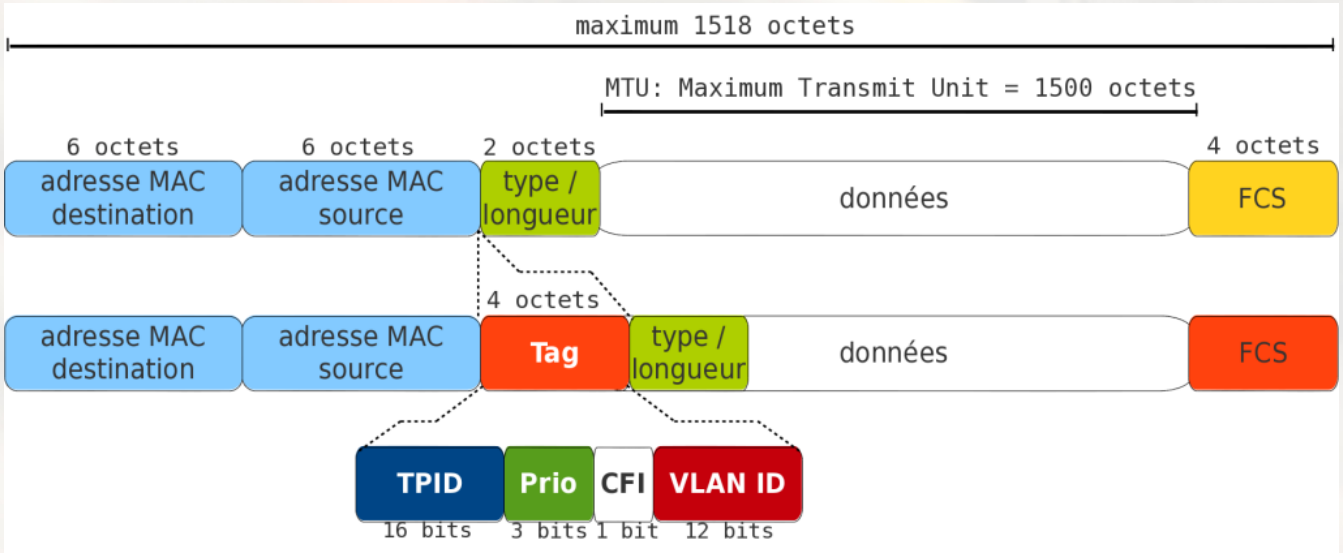
```
sudo vconfig rem eth0.1
```

Le tagging

Théoriquement il faudrait un port physique par VLAN.
Toutefois pour s'affranchir de ce problème on peut encapsuler (norme 802.1q) dans chaque trame diffusée le numéro du VLAN correspondant.
C'est ce que l'on appelle le "tagging" de VLAN.
Ce protocole permet donc de diffuser sur un même cable plusieurs VLAN's.
On peut analyser le tagging à l'aide de `tcpdump -v -i eth0 vlan`

Le tagging

La trame ethernet s'en trouve modifiée :

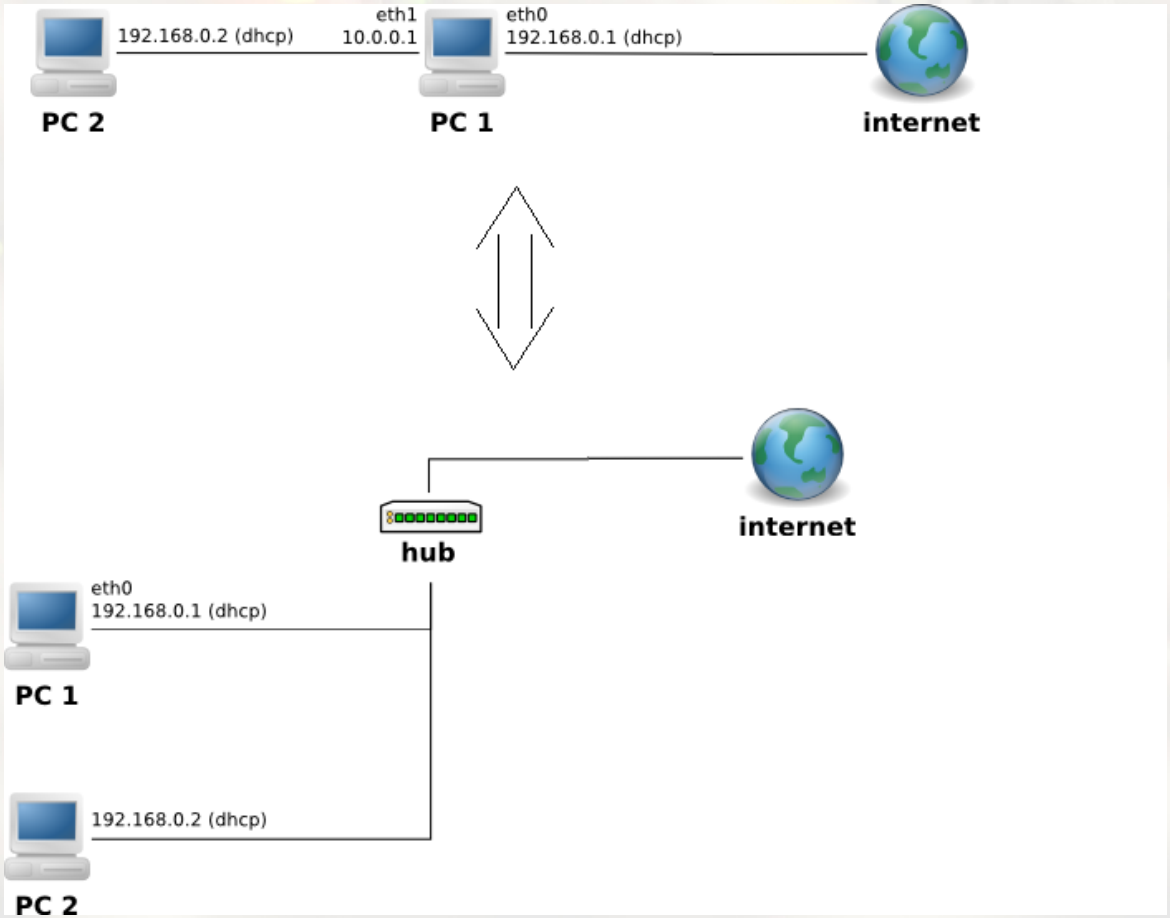


On remarque notamment la réduction de la MTU de 4 octets.

Sommaire

Quésako ?

Un bridge c'est un pont ethernet...
Si vous disposez d'une machine avec deux cartes ethernet le bridge va vous permettre de l'utiliser comme un switch :



Configuration

Il existe plusieurs méthodes, vous aurez besoin du paquet `bridge-utils` Pour créer le bridge

```
brctl addbr br0
```

Le bridge `br0` est créée il suffit maitenant de lier l'interface d'entrée avec la sortie

```
brctl addif br0 eth0 eth1
```

Pensez à verifier la bonne configuration des interfaces créer !

Et c'est tout ;)

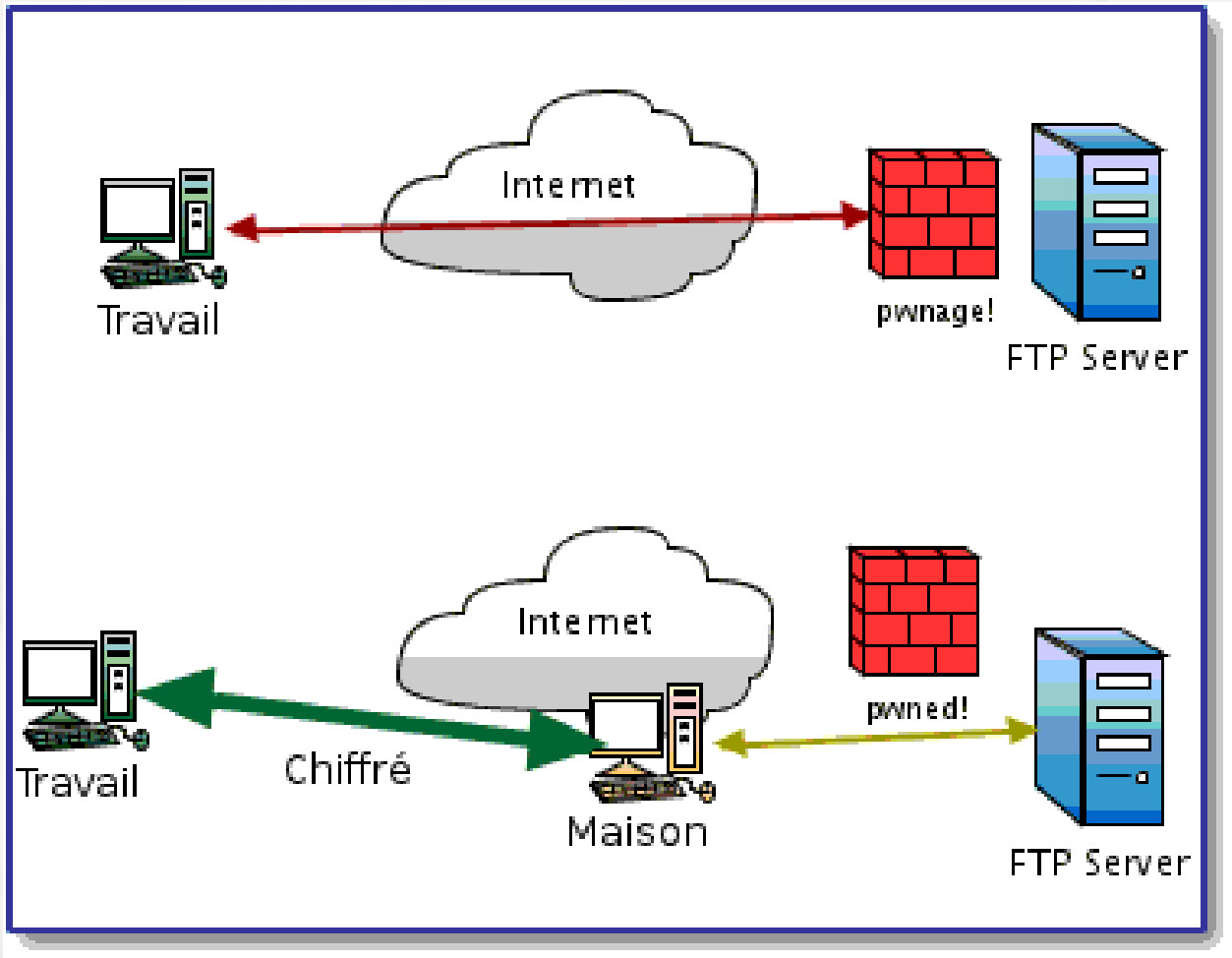
Toutefois il faudrait réiterer avec cette méthode la manipulation à chaque reboot. Pour bridger de façon plus correcte il faut modifier `/etc/network/interfaces`

Sommaire

Les tunnels

Un tunnel c'est l'encapsulation de données d'un protocole réseau dans un autre de même couche, ou de couche supérieure.
Un des avantages c'est notamment le fait que l'on peut chiffrer les données transitant par le tunnel, ce qui sécurisera la communication.

L'un des intérets des tunnels



Tunnel de port via SSH

On s'intéresse à l'encapsulation de paquets TCP/UDP dans le protocole SSH.
Ouvrir le tunnel va simplement correspondre à une redirection des ports vers le port SSH

```
ssh -L port-local:HOSTNAME:port-distant nom@serveur
```

Dans notre cas redirigeons donc le port 80 (http) vers le 2080

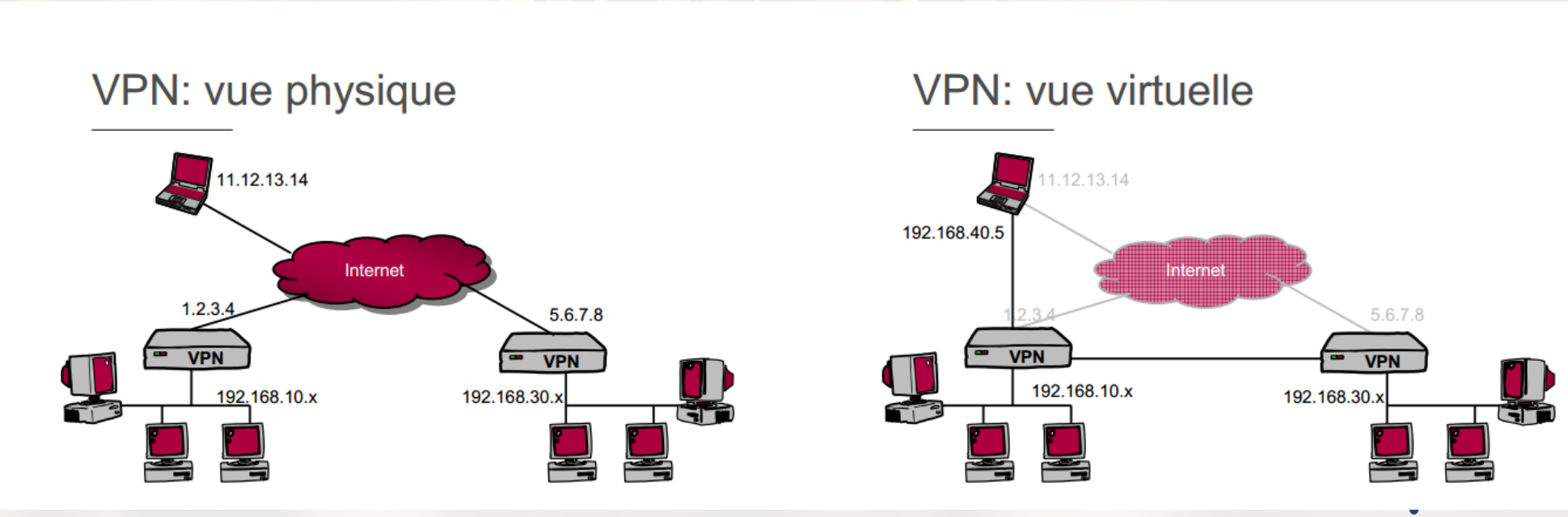
```
ssh -L 2080:localhost:80 nom@serveur
```

Pour l'utiliser il suffit de dire au navigateur d'emprunter le tunnel `http://localhost:2080` qui sort sur le port 80 du serveur distant !

Sommaire

Virtual Private Network

Un VPN est un réseau virtuel s'appuyant sur un autre réseau (Internet dans notre cas), il permet de communiquer entres les différents membre de ce VPN de manière sécurisé.
On peut considérer qu'une connexion VPN revient à une connexion LAN via Internet.



Fonctionnement

Basiquement le principe du VPN repose sur les protocoles de tunnelisation, on encapsule donc les paquets à transmettre dans un protocole de même niveau OSI)
L'opération se déroule en plusieurs étapes :

- Les paquets à transmettre sont encapsulés et chiffrés par un client VPN
- Les paquets chiffrés transitent par un autre réseau (Internet par exemple)
- Les paquets sont reçus, déchiffrés et transmis par le serveur VPN

Intérets

- Accéder à un réseau local à distance (Travail à domicile, jeux en LAN etc...)
- Sécuriser la transmission de données, en permettant la connexion à distance à un réseau de confiance.

Utilisation d'OpenVPN

Il existe de nombreux clients/serveurs VPN se basant sur divers protocoles d'encapsulation.
Dans notre cas nous allons considérer OpenVPN.
Sa configuration s'effectue en deux étapes

- Définition d'un réseau de sureté – Certificats de sécurité
- Configuration du client et du serveur

Géneration du certificat maître

Aprés installation d'OpenVPN :

```
cp /usr/share/doc/openvpn/examples/easy-rsa  /openvpn/ -R
cd  /openvpn/2.0/
```

Puis en étant sudoer transformez le fichier **vars** à votre convenance

Vous pouvez alors créer la configuration de votre certificat :

```
. ./vars
```

Supprimez les certificats déja existants

```
./clean-all
```

Et enfin génerez le certificat maître

```
./build-ca
```

Certificats et clefs pour les serveurs et clients

A présent on génere et on signe les certificats pour le client et pour le serveur

- Pour le serveur : `./build-key-server serveur`
- Pour le client : `./build-key client1`

Il reste à génerez les paramètres Diffie-Hellman pour permettre un échange sécurisé des clefs.
`./build-dh`
Vous obtenez ainsi des clefs signées pour votre serveur et votre client !

Il faut déplacer les fichiers respectivement crées dans le dossier `/etc/openvpn` de votre serveur et de votre client

Résumé

Vous devez a présent avoir
Coté Serveur :

- `ca.crt`
- `server.crt`
- `server.key`
- `dh1024.pem`

Coté Client :

- `ca.crt`
- `client.crt`
- `client.key`

Configuration du serveur

On travaillera sur le serveur `apprentis`
Vous pouvez trouver la configuration dans `/etc/openvpn/serveur.conf`
La configuration ce scinde en 3 parties importantes :

- `proto UDP/TCP` , précise si le serveur écoute sur un port TCP ou UDP
- `dev TUN`(Encapsulation IP) ou `dev TAP`(Encapsulation Ethernet) précise le type d'interface virtuelles
- Enfin configurer l'IP virtuelle du serveur et son masque de sous-réseau

Configuration du client

Au niveau du client la configuration est plus simple
Il faut éditer `/etc/openvpn/client.conf`

- Même configuration que le serveur concernant le port TCP ou UDP
- Adresse publique du serveur, la résolution DNS étant activée
  `remote apprentis.crans.org 1194`
- Verifiez le nom des clefs disponibles.

Connexion

Pour lancer le serveur

```
sudo openvpn server.conf
```

Et pour le client

```
sudo openvpn client.conf
```

C'est fait ! ! Vous êtes connectés au VPN !

Questions ?