R2.04-- Les bases des réseaux

TD1 : Les adresses IP

Adresse MAC & adresse IP

Dans un réseau isolé chaque machine est identifiée par son adresse machine (MAC). Une adresse MAC a 6 octets : 3 octets correspondant à son producteur/fournisseur et 3 octets propres à l'appareil réseau (carte réseau). Dans un réseau, des machines partageant le même médium physique (câble, fibre, etc.) peuvent communiquer les unes avec les autres sans avoir besoin de passerelle. Dans le modèle OSI, cette communication a lieu à la couche 2 (liaison). Au sein d'un seul réseau les machines peuvent connaître les adresses MACs des autres machines.

Lorsque deux machines situées dans deux réseaux différents veulent communiquer, cette communication a lieu à la couche 3 (réseau) du modèle OSI et nécessite des passerelles et des adresses IP. Alors, au delà de l'adresse MAC associée à chaque interface réseau, la machine aura aussi une ou plusieurs adresses IP (qui peuvent changer), lui permettant de communiquer à l'extérieur de son réseau.

Si la machine A veut communiquer avec la machine B, située dans un même réseau qu'elle, alors la machine A connaîtra l'adresse IP et l'adresse MAC de la machine B. La machine A peut choisir de communiquer avec la machine B à la couche 2 (liaison), sans désigner la machine B par son adresse IP.

 Deux machines dans un même réseau peuvent communiquer sans utiliser d'adresse IP.

Si les deux machines se situent dans deux réseaux différents, alors la machine A ne connaîtra que l'adresse IP de la machine B. L'envoi s'effectuera via une ou plusieurs passerelles – qui connaîtront l'adresse IP de l'expéditeur et du destinataire du message. Toutefois, les adresses MAC des machines A et B ne seront utilisées, ni connues, que par les machines dans leurs réseaux respectifs.

 Les adresses MAC ne seront connues (et utilisables) qu'au sein du réseau de la machine.

Dans la communication en réseau deux adresses MACs spéciales sont :

- 00 :00 :00 :00 :00 :00 , qui indique le fait que l'adresse MAC de la machine en question est inconnue. Cette adresse apparaîtra par exemple dans le cadre du protocole ARP, présenté plus tard dans cette ressource
- FF : FF :FF :FF :FF:FF, qui indique l'adresse MAC de broadcast. Si une machine envoie un message qui a pour destination l'adresse FF :FF :FF :FF :FF :FF, alors ce message est envoyé à toutes les machines partageant un même médium physique que la machine expéditeur

Adresse IP & domaine de collision

Dans cette ressource nous allons principalement utiliser des adresses IPv4. Une adresse IPv4 a quatre octets, ou 32 bits, chaque octet ayant des valeurs entre 0 et 255. On utilise un point « . » pour séparer les octets de l'adresse IP entre eux. Un exemple d'adresse valide est 12.34.56.78. Par contre, 12.345.6.78 n'est pas une adresse valide, car elle contient un octet supérieur à 255.

Un domaine de collision est un sous-ensemble d'adresses IP contigües, représentant un réseau informatique. La taille d'un domaine de collision est une puissance de 2 : c'est-à-dire un domaine de collision peut contenir 1, 2, 4, 8, 16 … adresses IP. Un domaine de collision peut être défini de trois façons distinctes, que nous allons détailler ci-après :

- **Notation CIDR (voir aussi ci-dessous)** : le domaine est défini par une adresse IP (typiquement la première adresse du domaine) suivie par un CIDR – un nombre entre 0 et 32 qui indique la taille du domaine de collision, les deux valeurs séparées par un / . Par exemple, une adresse d'un domaine de collision peut être 12.34.56.0/24. Plus le CIDR est petit, plus le domaine de collision est grand.
- **Masque de réseau** : le domaine est défini par une adresse IP (typiquement la première adresse du domaine) et une valeur sur 4 octets indiquant le masque de réseau (qui lui aussi indique la taille du réseau). Par exemple le domaine de collision 12.34.56.0/24 est le domaine 12.34.56.0 avec un masque de 255.255.255.0.
- **Classe de réseau** : le domaine de réseau est défini par une adresse IP (typiquement la première adresse du domaine) et une classe (A, B, C ou D), qui indique sa taille. Un réseau de classe A est très grand et correspond à un CIDR de /8, tandis qu'un réseau de classe D est petit et correspond à un CIDR de /28.

Si la première adresse incluse dans un domaine de collision est l'adresse du réseau lui-même, la dernière est utilisée pour faire du broadcast, c'est-à-dire, pour envoyer un message à toutes les machines du réseau.

Adresse IP d'un réseau : notation CIDR

L'adresse IP d'une machine se ressemble au nom d'une personne. La première partie de l'adresse – qui s'appelle la partie réseau – correspond au nom de famille, qui reste identique pour tous les membres de la même famille (c'est-à-dire toutes les machines qui font partie d'un même réseau). La dernière partie de l'adresse IP est la partie machine, qui se ressemble au prénom d'une personne dans une famille (typiquement les prénoms sont uniques dans une famille est permettent de distinguer les différents membres de la famille).

En fonction de la taille de la partie réseau d'une adresse IP, elle peut faire partie d'un réseau plus ou moins grand. La taille d'un réseau correspond au nombre de machines qui peuvent être hébergées dans le réseau – notamment le nombre de « prénoms » distincts possibles dans la partie machine.

La notation CIDR indique le nombre de bits de la partie réseau pour une adresse IP. Par exemple l'adresse 12.34.56.0/24 aura 24 bits sur la partie réseau, et donc 32-24 = 8 bits pour la partie machine. Chaque bit peut prendre 2 valeurs, 0 ou 1. En total, nous avons $2^8 = 256$ adresses possibles sur 8 bits. La première adresse possible est [00000000] = 0, tandis que la dernière correspond en binaire à [11111111] = 255.

Ceci veut dire que la première adresse IP dans le réseau 12.34.56.0/24 est 12.34.56.0, tandis que la dernière est 12.34.56.255. La première adresse est l'adresse du réseau et la dernière adresse est l'adresse de broadcast. Entre ces deux valeurs, les adresses de 12.34.56.1 à 12.34.56.254 peuvent être configurées à des machines faisant partie du réseau.

Un réseau de taille CIDR de x bits contient $2^{(32-x)}$ adresses, dont une utilisée pour l'adresse du réseau, une pour l'adresse de broadcast et $2^{(32-x)} - 2$ adresses machine

L'adresse IP du réseau est la première adresse de la plage donnée par son CIDR. Si un réseau a un CIDR de x, alors l'adresse IP de ce réseau :

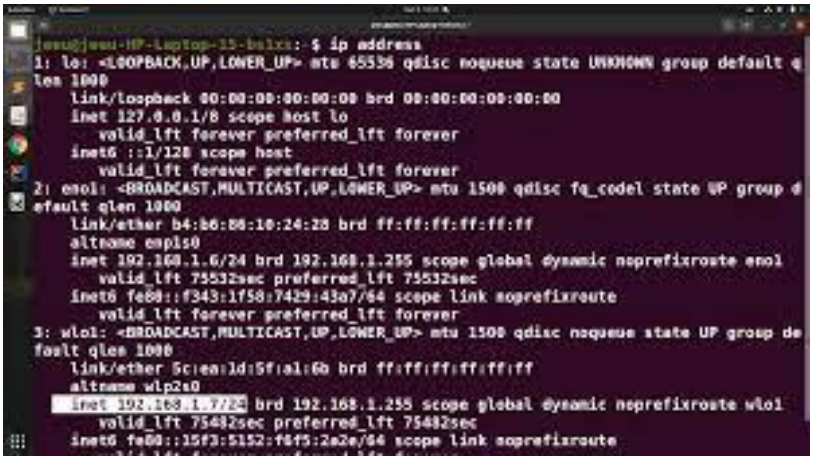
- *Peut* contenir des zéros sur ses x premiers bits ;
- *Doit obligatoirement* contenir des zéros sur toute la partie machine (les 32-x derniers bits).

Une adresse IP contenant au moins une valeur non-0 dans la partie machine n'est pas une adresse correcte de réseau.

Par exemple, l'adresse 12.34.56.0/24 est une adresse correcte de réseau, car ses 32-24 = 8 derniers bits (correspondant au dernier octet) sont égaux à 0. Au contraire, l'adresse 12.34.56.7/24 n'est pas une adresse correcte de réseau, car le dernier octet de cette adresse est [00000111], et contient des 1s sur la partie machine.

Comme les domaines de collision sont des plages contigües d'adresses IP, il est possible de découvrir à partir des informations présentes sur chaque machine de ce réseau.

En utilisant la commande `ip address` ou `ip address show` dans un terminal sur Linux, il est possible de visualiser la configuration IP de chaque machine.



Cette capture nous montre que la machine possède 3 interfaces réseau, dont une (lo) est l'interface de loopback (c'est-à-dire la machine l'utilise pour communiquer avec elle-même).

La valeur sélectionnée ci-dessus (192.168.1.7/24) est l'une des adresses IP de cette machine. La valeur CIDR de 24 indique la présence de $2^{32-24} - 2 = 2^8 = 256$ adresses, dont une (la première) est l'adresse du réseau et la dernière est l'adresse de broadcast, donc il restent 254 adresses machine.

L'adresse 192.168.1.7/24 est une adresse machine (pourquoi ?). En analysant cette valeur, on apprend les valeurs suivantes :

- Sur les 4 octets de l'adresse IP, les 24 premiers bits (les 3 premiers octets) forment la partie réseau. Pour obtenir l'adresse IP du réseau, les 32-24=8 bits restants sont mis à 0. Ceci forme l'adresse 192.168.1.0/24.
- L'adresse IP de la première machine dans ce réseau est l'adresse suivante à l'adresse du réseau : 192.168.1.1/24.
- L'adresse de broadcast est la dernière adresse de la plage : les 24 premiers bits sont la partie réseau (le préfixe commun à toutes les adresses du réseau), tandis que les 32-24 derniers bits sont mis à 1. Ceci veut dire que l'adresse du broadcast est : 192.168.1.[1111 1111] = 192.168.1.255. Elle est, d'ailleurs, indiquée dans la capture d'écran ci-dessus à côté du mot clé `brd`.
- L'adresse de la dernière machine (hors broadcast) est celle juste d'avant l'adresse de broadcast, notamment 192.168.1.254.

Prenons un deuxième exemple, pour l'adresse réseau 192.168.137.0/23. Première question : cette adresse est-elle une adresse machine ou une adresse réseau ? Le CIDR est de 23, donc la partie réseau contient 23 bits (2 octets et 7 sur les 8 bits du troisième). Nous écrivons en binaire le troisième octet : 137 = [1000 1001]. Les 7 premiers bits sont inclus dans la partie réseau de l'adresse, mais le huitième (égal à 1) est dans la partie machine. On peut conclure que l'adresse 192.168.137.0/23 est une adresse machine, car elle contient au moins un 1 dans la partie machine. Pour cette adresse :

- La partie machine se compose des 23 premiers bits : 192.168.[1000 100] suivis par 32-23 = 9 zéros : 192.168.[1000 1000].0/23 = 192.168.136.0/23
- L'adresse réseau se compose des 23 premiers bits de l'adresse, suivis par 32-23 = 9 uns, donc : 192.168.[1000 1001].[1111 1111] = 192.168.137.255
- L'adresse contient $2^{32-23} = 2^9 = 512$, dont l'adresse réseau et l'adresse broadcast, donc 510 adresses machines
- L'adresse IP de la première machine est 192.168.136.1, tandis que la dernière est 192.168.137.254.

Adresse IP avec un masque de réseau (netmask)

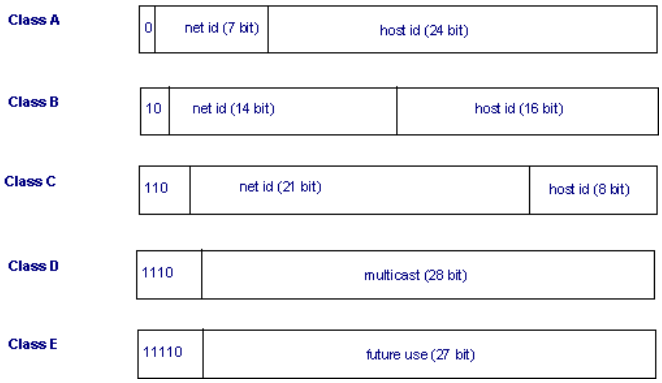
Une façon alternative d'écrire une adresse réseau est de donner l'adresse et un masque de réseau. Le masque de réseau indique le nombre de bits qui sont fixés comme partie réseau, en utilisant un numéro sur 4 octets écrit dans le même format qu'une adresse IP. Les premiers bits du masque sont tous égaux à 1 et les derniers sont tous égaux à 0. Pour un CIDR de x, les x premiers bits sont la partie réseau. Le masque

de réseau aura alors x fois 1, suivis par 32-x fois 0. Par exemple, un CIDR de 20 donne un masque de réseau de [1111 1111].[1111 1111].[1111 0000].[0000 0000] = 255.255.240.0.

- Exemple : 192.168.33.0, netmask 255.255.255.0
  On écrit 255.255.255.0 en notation binaire : 11111111. 11111111.11111111.00000000. Ceci nous donne donc 24 bits égaux à 1 et 8 égaux à 0. Notre adresse réseau est donc équivalente à 192.168.33.0/24 en notation CIDR. Ceci est une adresse réseau (aucun 1 parmi les 32-24 = 8 derniers bits). L'adresse de broadcast contient les 24 premiers bits réseau, suivi par des uns : 192.168.33.[1111 1111] = 192.168.33.255. Dans ce domaine a un total de $2^8 = 256$ adresses dont une adresse de réseau, une adresse de broadcast et 254 adresses machine. La première adresse machine est : 192.168.33.1. La dernière adresse qu'on peut donner à une machine est : 192.168.33.254.
- Exemple : 192.36.2.0 netmask 255.255.255.128
  On écrit le masque en binaire : 11111111.11111111.11111111.10000000. Ceci nous donne 25 bits fixés, soit une adresse en notation CIDR de 192.36.2.0/25. Ce réseau contient 2^7 = 128 machines, dont l'adresse réseau, une adresse broadcast et 126 adresses machine. L'adresse 192.36.2.0/25 est l'adresse réseau. L'adresse de broadcast est : 192.36.2.[0111 1111] = 192.36.2.127. La première adresse machine est 192.36.2.1 et la dernière adresse machine est : 192.36.2.126.

Classes de réseaux

Avant l'introduction de la notation CIDR, on utilisait beaucoup les classes de réseaux, une répartition qui dépendait de la taille approximative du réseau et fixait les premiers bits de l'adresse réseau. On avait plusieurs classes de réseaux, comme indiqué ci-dessous :



Source : https://www.eventhelix.com

La classe d'un réseau est le plus facilement visible sur le premier octet de l'adresse réseau :

Classe	Premier bit(s) :	Premier octet :	Equivalent CIDR :
Classe A	0	0-127	/8
Classe B	10	128-191	/16
Classe C	110	192-223	/24
Classe D	1110	224-239	/28

Ce système est encore en usage, même s'il donne beaucoup moins de flexibilité qu'une notation CIDR.

- Exemple : imaginons une machine dont l'adresse IP est 135.166.2.34
  Le premier octet nous indique déjà qu'il s'agit d'une machine dans un réseau de classe B (car le premier octet de l'adresse IP est 135). Un réseau de classe B est sur /16 bits, donc le premiers 16 bits de l'adresse sont la partie réseau et les derniers 16, de la partie machine. Notamment cette machine fait partie du réseau 135.166.0.0/16. Le nombre maximal de machines qui peuvent être incluses dans ce réseau est $2^{16} - 2 = 65534$ machines.

Adresses IP particulières

Parmi l'espace de toute adresse IP valide, il y a quelques plages d'adresses qui ne peuvent pas être utilisées pour une machine dans une connexion directe à l'Internet :

- En classe A : 10.0.0.1 -- 10.255.255.254
- En classe B : 172.16.0.1 -- 172.31.255.254
- En classe C : 192.168.0.1 -- 192.168.255.254

Puisque ces adresses ne sont pas directement liées à l'Internet, on peut les utiliser pour un grand nombre de machines en parallèle, dans des divers sous-réseaux, notamment pour les utilisateurs qui n'ont pas acheté une autre plage d'adresses.

En plus de ces adresses, il y a également quelques adresses particulières à un usage dédié :

- 127.0.0.1/8 : est une adresse de loopback (le message revient à la machine qui l'a envoyé). Ceci peut être utile lorsque la machine tente de se connecter à un serveur local
- 255.255.255.255 : broadcast IP
- 0.0.0.0 : absence d'adresse IP (utilisé en DHCP lorsqu'on demande justement d'avoir une adresse IP qui sera attribué à notre machine)
- 0.0.0.0/0 : toutes les adresses possibles.

Configurer une adresse IP

Une machine peut avoir une ou plusieurs interfaces réseau, dont certaines correspondent à des cartes réseau physiques et certaines correspondent à des ponts virtuels (par exemple dans le cas de l'utilisation des conteneurs). Chaque interface réseau a un nom, par exemple eth0, eth1, lo, wif31, etc.

L'interface `lo` est une interface spéciale : elle ses réfère à l'interface qu'une machine utilise pour communiquer avec elle-même. Ceci est le cas par exemple lorsqu'une machine héberge plusieurs services (par exemple un serveur http et un serveur mail).

Par défaut l'adresse IP de l'interface loopback d'une machine est 127.0.0.1/8

Une interface peut avoir plusieurs adresses IP – toutefois, cela a peu de sens dans le cas des adresses IPv4. Pour le cas IPv6 la situation est différente : il n'est pas inhabituel de voir une adresse IPv6 à portée globale (routable) et une adresse IPv6 à portée locale (non-routable) sur une même interface de réseau.

Pour configurer une adresse IPv4 la commande à utiliser est :

```
ip address add <adresse>/<CIDR> dev <nom interface>
```

Par exemple, la commande :

```
ip address add 192.168.1.2/24 dev eth0
```

configure l'adresse 192.168.1.2/24 sur l'interface eth0 de la machine.

La commande ip fonctionne également en formule raccourcie : tout raccourci du mot « address » peut être substitué dans la commande et elle fonctionnera également, par exemple :

```
ip a add 192.168.1.2/24 dev eth0
```

Pour enlever une adresse IP, la commande est identique, en remplaçant add par del :

```
ip address del 192.168.1.2/24 dev eth0
```

Attention ! Si, dans la commande `ip address add` l'adresse IP n'est pas suivie par le CIDR, alors par défaut l'adresse est configurée en /32. Combien d'adresses y a-t-il dans un réseau en /32 ?

Une interface peut être dans un état actif ou passif. Par défaut, toutes les interfaces d'une machine sauf l'interface `lo` sont dans un état passif. Même la configuration d'une adresse IP sur une interface ne peut pas la rendre dans un état actif. Pour activer une interface, il faut taper la commande suivante :

```
ip link set dev <nom de l'interface> up
```

ou

```
ip link set <nom de l'interface> up
```

Par exemple : `ip link set dev eth0 up` rendra l'interface eth0 active.

Même si une interface est configurée, on ne peut pas l'utiliser pour échanger des messages si elle n'est pas active. Même dans son propre réseau, la machine restera invisible. Pour visualiser les interfaces d'une machine (actives ou passives) la commande à utiliser est :

```
ip address  ou    ip address show
```

Si, en revanche, le but est de visualiser seulement les interfaces actives d'une machine, la commande est :

```
ip address show up
```

Configurer une infrastructure en réseau

Pour construire son infrastructure en réseau, une entreprise peut acheter une plage d'adresses publics (c'est-à-dire, routables). Elle peut également utiliser des plages d'adresses privées selon besoin et à volonté. Une fois en possession d'une plage d'adresses de la taille souhaitée, l'entreprise peut la diviser en plusieurs sous-plages, selon besoin.

Prenons l'exemple d'une entreprise qui se spécialise dans le développement d'applications web pour la logistique. Cette entreprise achète la plage 192.3.25.0/24. Elle veut héberger 200 machines en total. Pour assurer la sécurité de son réseau, l'entreprise décide de la diviser en trois sous-parties, chacune représentant un sous-réseau indépendant :

- Un sous-réseau de 120 machines pour les employées de l'entreprise (y compris services administratives, direction, etc.)
- Un sous-réseau de 80 machines pour les clients et les invités de l'entreprise

La première question à se poser est : la plage en /24 suffira-t-elle pour héberger 200 machines en total ? La réponse est oui, car une infrastructure en /24 héberge $2^8 - 2 = 254$ machines.

La division d'une plage en plusieurs sous-plages se fait par des divisions successives en 2. À chaque fois, une plage en /x divisée en deux donnera deux sous-plages en /x+1.

Ci-dessus nous avons besoin de 2 sous-plages, une de 120 et une de 80 machines. Une division de la plage en /24 en deux donnera deux plages en /25. Une plage en /25 peut héberger $2^7 - 2 = 126$ machines, donc chaque sous-plage en /25 suffit pour héberger soit 120 soit 80 machines.

La grande plage achetée a l'adresse IP suivante : 192.3.25.0/24. La partie réseau est 192.3.25 (correspondant aux 3 premiers octets) et la partie machine est le dernier octet, c'est-à-dire [0000 0000]. Chacune des sous-plages en /25 aura les 24 premiers bits de la grande plage : 192.3.25. Les 32-25 = 7 derniers bits seront fixés à 0. L'adresse de la première sous-plage est donc de la forme 192.3.25.[x000 0000] et l'adresse de la deuxième sous-plage est de la forme 192.3.25.[y000 0000]. Pour différencier donc entre les deux, l'une des deux valeurs (x ou y) doit être 0 et l'autre, 1.

L'entreprise peut alors choisir par exemple la sous-plage 192.3.25.[0000 0000]/25 = 192.3.25.0/25 pour les employeurs et la sous-plage 192.3.25.[1000 0000] = 192.3.25.128/25 pour les invités et les clients.

Attention : une fois qu'une plage est réservée pour une partie de l'infrastructure, on ne peut plus la rediviser pour faire place à une autre partie de l'infrastructure !

Exercice

Dans le tableau ci-dessous complétez les éléments manquants.

Réseau (CIDR)	Masque	Adresse début	Adresse fin	Broadcast	# adresses
192.168.1.0/24	255.255.255.0	192.168.1.1	192.168.1.254	192.168.1.255	254
172.16.24.32/28					
10.25.51.0/16					
10.253.45.0/30					
		10.253.44.1			510
		192.287.3.1	192.287.3.254		
			192.3. 2.254		62

TD2 : Le routage

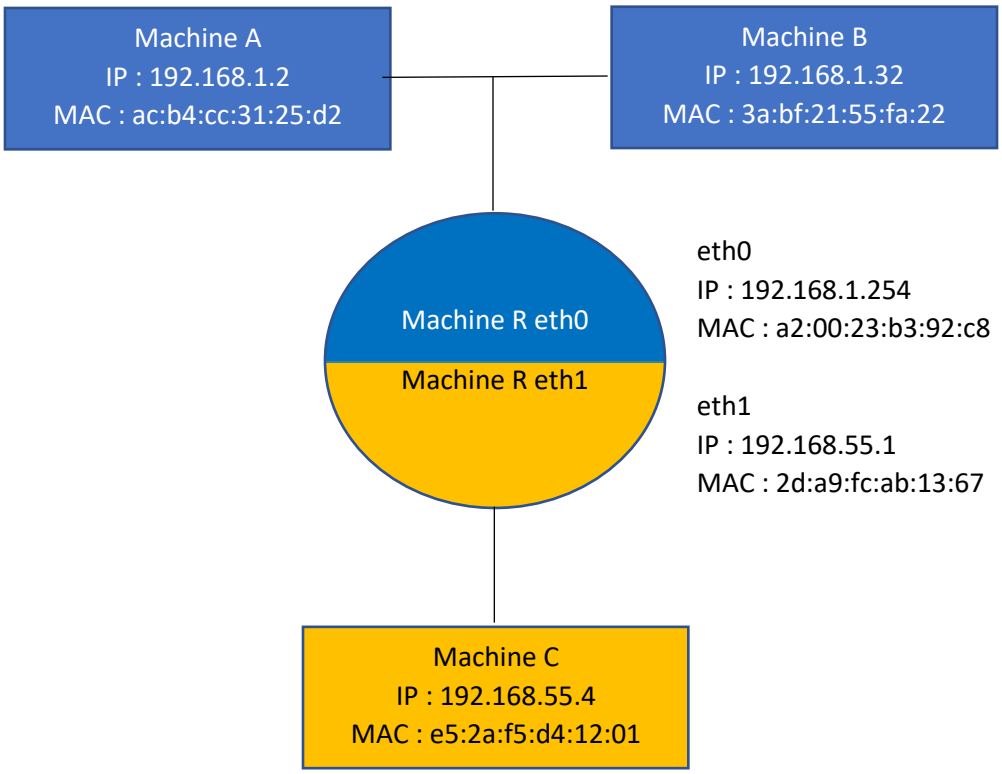
Les envois intra- et inter-réseau

Dans un seul sous-réseau, sur lequel les machines partagent un même médium physique, les machines peuvent communiquer en utilisant le protocole Ethernet (transmission à la couche 2). De plus tout message d'un protocole de couche 2 (par exemple ARP) sera directement encapsulé dans Ethernet 2.

Rappel : Le modèle OSI de réseaux consiste de 7 couches (la plus basse étant la couche 1 – Physique alors que la plus haute est la couche 7 – Application). Un message peut être envoyé de n'importe quelle couche du réseau, en l'encapsulant successivement à la couche d'envoi et aux couches inférieures. L'encapsulation d'un message consiste du rajout de préfixes et suffixes qui permettent au destinataire de pouvoir traiter le message correctement.

Pour les transmissions inter-réseau on a besoin de machines spéciales dans les réseaux, notamment des passerelles. Une passerelle fait la connexion entre deux (ou plus) sous-réseaux et appartient à chacune de ces réseaux (elle a notamment une adresse IP dans chacun des sous-réseaux qu'elle connecte).

Une passerelle qui connecte deux sous-réseaux aura au moins deux interfaces. Sur chaque interface elle aura une adresse IP dans le sous-réseau respectif (et bien-sûr, une adresse MAC). Notamment :



IP-forwarding : Passerelle vs. machine à plusieurs adresses IP

Une passerelle a plusieurs adresses IP. Toutefois, une machine avec plusieurs adresses IP n'est pas forcément une passerelle. La différence principale entre une passerelle et une machine à plusieurs adresses IP intervient dans la communication. Normalement, une machine qui reçoit un message destiné à une adresse IP différente aux siennes ne traite pas ce message. De plus, si un message est reçu sur une des interfaces d'une machine, il n'est pas obligatoirement transféré aux autres interfaces (ce qui peut être problématique si par exemple la machine héberge un serveur http qui est à l'écoute seulement sur une des interfaces de la machine.

Pour transformer une machine à plusieurs interfaces dans une passerelle, il faut activer l'ip-forwarding : notamment le processus par lequel :

- Un message reçu sur une interface peut être transféré aux autres interfaces
- Un message destiné à une adresse IP distincte aux adresses de la machine elle-même sera transféré plus loin (selon le tableau de routage de la machine)

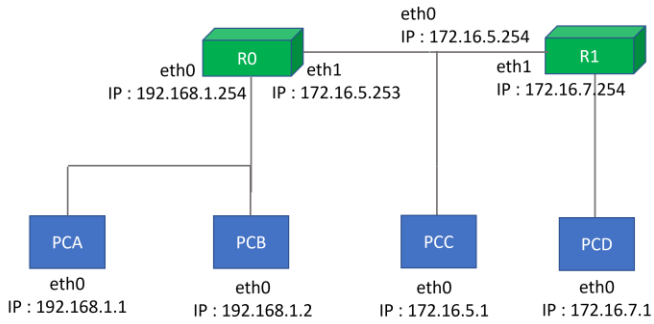
Attention : Dans l'émulateur Kathará (que nous allons utiliser parfois en TP), souvent l'IP forwarding est activé par défaut. Toutefois, ceci n'est pas le cas dans une configuration standard d'un élément de réseaux. Pour activer l'IP forwarding en IPv 4, il faut taper la commande :

```
echo 1> /proc/sys/net/ipv4/ip_forward
```

Les envois intra- et inter-réseau

Dans le cas d'une transmission de messages d'une machine située dans un réseau à une machine faisant partie d'un autre réseau comme celui décrit ci-dessous. Dans cette figure, nous supposons que les réseaux des machines PCC et PCD sont en /24, et celle de PCA et PCD ont un CIDR cohérent avec sa classe.

L'infrastructure comporte 2 passerelles : R0 et R1, et 3 réseaux : un premier réseau contenant PCA, PCB et R0 (eth0), un deuxième réseau avec PCC, R0 (eth1) et R1 (eth0) et un dernier réseau avec PCD et R1 (eth1).



Supposons que la machine PCA veut envoyer un message à PCB. Il s'agit d'un envoi intra-réseau (c'est-à-dire, la machine source et la machine destination se trouvent dans un même réseau). Dans ce cas, l'envoi

ne nécessite pas de passerelle et le message sera encapsulé dans le protocole Ethernet 2 à la couche 2 (liaison). Le message spécifie les éléments suivants :

- L'adresse MAC de PCA en source
- L'adresse MAC de PCB en destination

Potentiellement (en fonction du type de message échangé), l'en-tête du message spécifiera également :

- L'adresse IP de PCA en source
- L'adresse IP de PCB en destination.

Maintenant supposons qu'il y a une transmission entre PCA et PCC. Pour l'instant nous allons supposer que le routage a été correctement configuré (nous verrons comment réaliser le routage ci-dessous), donc la transmission passe via R0. Le message arrive au destinataire en deux temps. A chaque étape de l'envoi, le message est encapsulé par le protocole IP (couche 3), puis par le protocole Ethernet (couche 2). Les adresses IP (couche 3) indiquent les vrais expéditeurs et destinataires du message : donc, pour les deux étapes de la transmission inter-réseau, ces adresses resteront identiques. Les adresses MAC, par contre, n'indiquent que les deux traiteurs actuels du message : les entités qui vont faire suivre le message de son expéditeur initial à son destinataire final.

Pour la première étape de transmission, effectuée entre PCA et R0, les adresses évoquées dans l'en-tête du message sont :

- L'adresse IP de PCA en tant qu'adresse source
- L'adresse IP de PCC pour l'adresse IP Destinataire
- L'adresse MAC de PCA en tant qu'adresse MAC source
- L'adresse MAC de R0 (eth0) en tant qu'adresse MAC du destinataire

Lorsque R0 reçoit ce message, il trouve son adresse MAC en tant que destination et regarde les adresses IP de l'expéditeur et du destinataire. L'adresse IP du destinataire est dans le réseau de R0 (eth1) et donc la transmission entre ces deux machines peut s'effectuer directement, sans nécessiter une passerelle supplémentaire.

La deuxième étape de transmission comportera les éléments suivants :

- L'adresse IP de PCA en tant qu'adresse IP source
- L'adresse IP de PCC en tant qu'adresse IP destinataire
- L'adresse MAC de R0 (eth1) en tant qu'adresse MAC source
- L'adresse MAC de PCC en tant qu'adresse MAC destinataire

Le routage

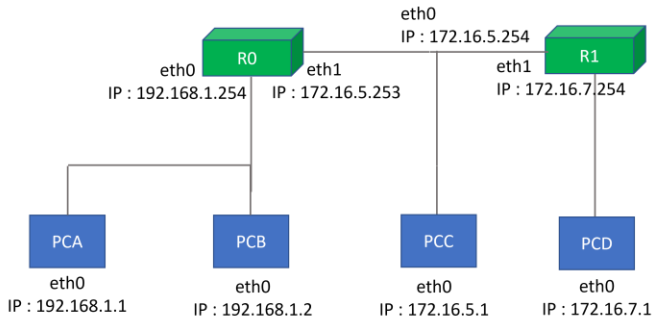
Pour configurer un réseau il faut configurer la façon dont les messages destinés à une machine hors du réseau vont être envoyés. Chaque machine aura donc une ou plusieurs passerelles responsables de faire passer des messages hors du réseau de la machine.

Il ne faut jamais configurer une passerelle pour transmettre des messages dans son propre réseau – elle le fera par défaut !

Le rôle de chaque passerelle est d'effectuer une seule étape de transmission : accepter un paquet et le transmettre une étape plus loin :

- Soit à la machine destinataire, si le destinataire du message et la passerelle partagent un réseau
- Soit à une prochaine passerelle, si une règle de routage existe dans le tableau de routage de la passerelle pour le destinataire du message
- Si aucune route n'existe sur la passerelle pour faire passer le message au destinataire, alors la passerelle renvoie le message vers la machine source (dans un message d'erreur encapsulé par le protocole ICMP).

Reprenons l'architecture présentée ci-dessus.



Si PCA veut envoyer un message à PCB, elle n'a pas besoin de passerelle.

Pour des envois hors-réseau, par exemple vers PCC, par contre, PCA a besoin d'une passerelle. En général, les machines utilisateur auront une **passerelle par défaut**. Pour configurer sur une machine une passerelle par défaut il faut utiliser la commande :

```
ip route add default via <adresse IP passerelle sans CIDR>
```

L'adresse IP de la passerelle doit être dans le même domaine de collision que la machine sur laquelle on configure cette passerelle. Sinon, on aura un message d'erreur et la configuration ne sera pas prise en compte.

Dans notre exemple, il faudrait taper sur PCA la commande :

```
ip route add default via 192.168.1.254
```

Pour une machine sur laquelle une seule passerelle est configurée – et elle est une passerelle par défaut – cette passerelle est empruntée à chaque envoi hors-réseau, peu importe l'adresse IP du destinataire.

Pour vérifier l'état de la connexion entre PCA et PCC, nous pouvons utiliser un message de type `ping`.

Pour envoyer un ping, on utilise la commande :

```
ping <@IP du destinataire>
```

Cette commande tentera d'envoyer des ping jusqu'à ce qu'on arrête l'exécution avec Ctrl+C. Pour envoyer un nombre fixe de pings, on utilise l'option -c :

```
ping -c <nombre de pings> <@IP du destinataire>
```

Un message de type ping est encapsulé dans le protocole ICMP sur IP (car il vérifie la connectivité inter-réseau plutôt qu'intra-réseau). Lorsqu'on fait un ping de PCA vers PCC, le protocole prévoit de premièrement faire passer un message (ping request) de PCA vers PCC. Lorsque PCC reçoit ce message, il envoie une réponse (ping response) vers PCA.

Il y a donc plusieurs raisons pour lesquelles un ping ne fonctionne pas entre les deux machines :

- Il y a un problème de routage sur la route de PCA vers PCC
- Il y a un problème de routage sur la route inverse (PCC vers PCA)
- Il y a un problème avec soit l'un, soit l'autre réseau

Un logiciel du type Wireshark peut nous indiquer les messages reçus et envoyés sur chaque machine. En utilisant les indices donnés par un tel logiciel, nous pouvons comprendre où se trouve le problème et le résoudre.

Dans le cas de notre figure, si on tape sur PCA :

```
ping 172.16.5.1
```

alors le ping échouera, même après avoir configuré une passerelle par défaut pour PCA. En effet, cette configuration suffit pour assurer la transmission PCA → R0 → PCC, mais pas pour assurer le retour du ping dans la direction PCC → R0 → PCA. Pour faire passer le ping, il faut configurer une passerelle par défaut sur PCC :

```
ip route add default via 172.16.5.253
```

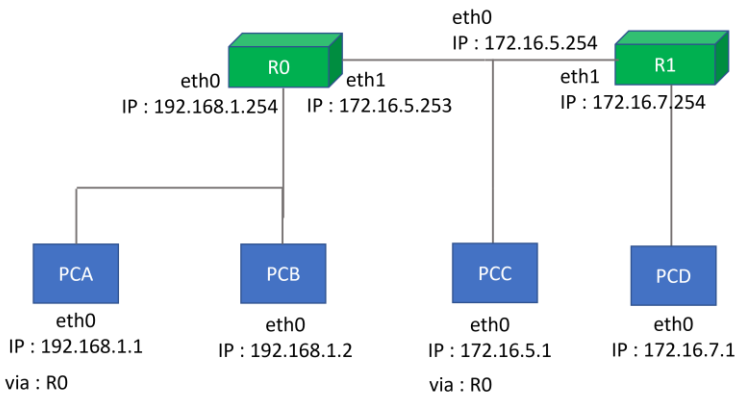
Si les machines utilisateur ont souvent une passerelle par défaut, les passerelles elles-mêmes peuvent avoir un tableau de routage plus fin, avec certaines passerelles dédiées pour certaines routes et certaines, pour d'autres routes. Pour ajouter une passerelle spécifique, l'instruction à taper est :

```
ip route add <réseau du destinataire avec CIDR> via <@ passerelle sans CIDR>
```

Attention : la passerelle dont l'adresse est incluse dans cette commande n'est que la prochaine étape de transmission. On réitère le fait que la chaque passerelle est responsable seulement de la transmission du paquet une étape plus loin !

Attention : la passerelle dont l'adresse est incluse dans cette commande n'est que la prochaine étape de transmission. On réitère le fait que la chaque passerelle est responsable seulement de la transmission du paquet une étape plus loin !

Reprenons l'architecture de réseau ci-dessus, avec les configurations de passerelles par défaut sur PCA et PCC (ceci est indiqué dans la figure ci-dessous par le texte « via : R0 »).



Sur PCA on tape la commande suivante :

```
ping 172.16.7.1
```

Le ping partira de PCA vers sa passerelle par défaut (R0). Celle-ci se rend compte que le destinataire du ping (PCD) se trouve dans un autre réseau que le sien. Il lui faut une passerelle pour envoyer le ping, mais elle n'a pas une passerelle.

Pour l'architecture ci-dessus nous avons deux choix pour configurer une passerelle sur R0 :

- Comme la seule autre passerelle dans le réseau de R0 est R1, donc R1 pourrait devenir la passerelle par défaut de R0 (option facile, mais potentiellement une mauvaise pratique) ;
- On configure, sur R0, une route spécifique vers le réseau de PCD via R1 (option potentiellement laborieuse mais plus sécurisée).

Pour configurer, sur R0, une route spécifique vers le réseau de PCD, la commande à taper est :

```
ip route add 172.16.7.0/24 via 172.16.5.254
```

Plus généralement la commande est :

```
ip route add <réseau destination avec CIDR> via <@ passerelle sans CIDR>
```

Maintenant la transmission du ping effectué sur PCA (vers PCD) est garantie de PCA à PCD car :

- Il y a une passerelle par défaut sur PCA (R0)
- R0 a une route spécifique (R1) pour les messages à destination de 172.16.7.0/24 (réseau de PCD)
- R1 partage un réseau avec PCD et peut lui livrer le ping

Malheureusement, le chemin de retour n'est pas encore configuré (pouvez-vous trouver les routes à configurer et sur quelles machines ?)

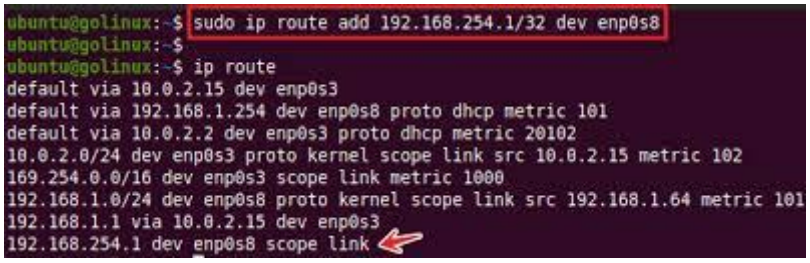
Le tableau de routage de chaque machine peut être visualisé en utilisant la commande :

```
ip route show
```

ou par le raccourci `ip route` .

Même une machine qui n'a aucune passerelle configurée aura une ligne dans son tableau de routage, indiquant son propre réseau (et implicitement, les machines avec lesquelles elle peut communiquer). Le tableau de routage est parcouru par spécificité de l'adresse de destination (par exemple les destinations /26 avant les /8 ou les /0).

Dans la capture suivante, la première commande configure une route vers le réseau 192.168.254.1/32 (c'est un réseau en /32, donc une machine isolée), tandis que la deuxième affiche les routes configurées.



Les 3 premières lignes de ce tableau de routage sont des passerelles pas défaut. Les lignes 2 et 3 indiquent des passerelles pour deux interfaces différentes, configurées via un protocole d'adressage dynamique qui s'appelle DHCP (nous regarderons ce protocole plus tard cette année), tandis que la première ligne est plus générique et dénote l'adresse de la passerelle par défaut de la machine.

Les lignes 4-6 indiquent les réseaux desquels la machine fait partie (elle n'a pas besoin de passerelle pour ces réseaux). Les lignes 4 et 6 sont spéciales, car elles correspondent à un adressage dynamique (DHCP).

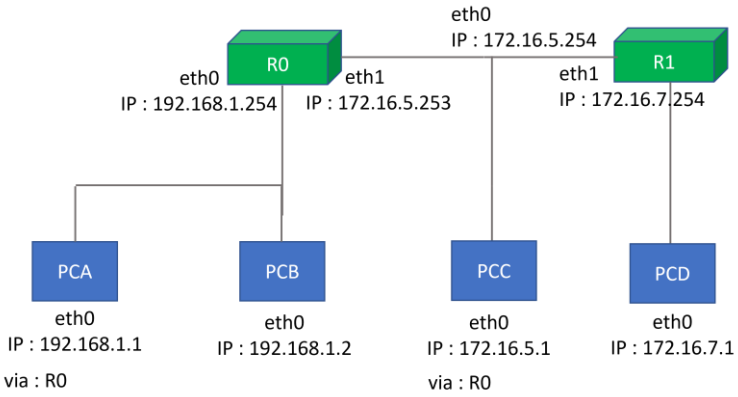
Les lignes 7 et 8 sont des routes spécifiques, la ligne 7 vers 192.168.1.1/32 et la ligne 8, vers 192.168.254.1/32.

Le protocole ARP

À chaque fois qu'un message est envoyé en réseau, il est encapsulé de la couche à laquelle fonctionne le protocole à la couche 1, avant d'être physiquement transmis. Pour la réception, la décapsulation est réalisée en ordre inverse.

Par exemple, un ping est encapsulé par le protocole ICMP, qui fonctionne à la couche 3. Les messages ICMP sont encapsulés premièrement à la couche 3 (par le protocole IP) et ensuite à la couche 2 (par le protocole Ethernet). À la couche 3, l'en-tête IP comporte un nombre important d'informations y compris l'adresse IP de la source et l'adresse IP du destinataire. À la couche 2, l'en-tête Ethernet demande une adresse MAC pour la source et une adresse MAC pour la destination.

Rappelons-nous de l'architecture de notre exemple :



Si PCA veut envoyer un message à PCB, nous rappelons que les adresses IP et MAC de la source correspondront au PCA et les adresses IP et MAC de la destination seront celles de PCB.

Comment la machine PCA peut-elle connaître l'adresse MAC de PCB ?

Le protocole ARP permet de trouver une adresse MAC à partir d'une adresse IP connue. Chaque machine sauvegarde un tableau de voisins, qui stocke des correspondances connues entre des adresses IP et des adresses MAC. Cette correspondance peut être visualisée en utilisant la commande :

```
ip neigh show
```

et vidée en utilisant

```
ip neigh flush
```

À chaque nouvel envoi d'un message, la machine vérifie si l'adresse MAC est déjà connue et, si ceci n'est pas le cas, elle envoie une requête ARP pour trouver l'adresse MAC.

L'adresse MAC d'une machine ne peut être demandée/apprise que par une autre machine dans un même réseau. Les adresses MAC ne sortent jamais de leurs réseaux.

Les messages ARP sont encapsulés sous la forme d'un trame Ethernet, et donc elles sont seulement utilisables à la couche 2.

Le protocole se compose de deux messages :

- Une requête ARP envoyée en broadcast, vers l'adresse MAC FF:FF:FF:FF:FF:FF. Le message de la requête est « Qui a l'adresse <adresse IP de la machine ciblée> ? Dites à <adresse de la machine qui cherche l'adresse MAC> »
- Une réponse envoyée en unicast (contenant l'adresse demandée) à l'expéditeur

Le protocole ARP se déroule alors entre plusieurs entités :

- L'expéditeur de la requête, notamment l'entité qui cherche l'adresse MAC correspondant à une machine ciblée (dont l'adresse IP est connue)
- L'expéditeur de la réponse, notamment une machine qui connaît l'adresse MAC de la cible

- Les autres machines dans le réseau de l'expéditeur, qui ont toutes reçues la requête ARP

L'expéditeur de la réponse et la cible peuvent représenter deux machines différentes : il suffit que l'expéditeur connaisse l'adresse MAC de la cible.

Un échange ARP a lieu à chaque étape d'une transmission.

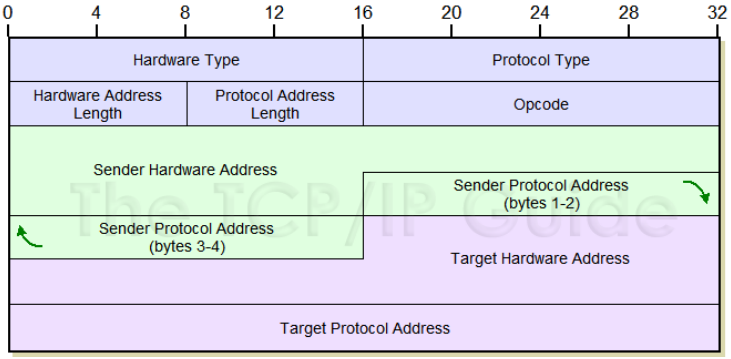
Reprenons l'exemple du ping de PCA à PCB (les deux machines dans un même réseau). Supposons que les machines viennent d'être allumées et leurs caches ARP sont vides. Un logiciel de traçage de paquets comme Wireshark interceptera les messages suivants entre les deux machines :

- Une requête ARP de PCA ciblant l'adresse IP 192.168.1.2 (PCB)
- Une réponse ARP de la part de PCB (ou de R0 eth0 si ce dernier a stocké l'adresse de PCB) répondant avec l'adresse MAC de PCB
- Le ping part de PCA à PCB

La même suite de messages sera interceptée pour le chemin de retour.

Les éléments des messages ARP

Le protocole ARP a l'en-tête suivant :



Source : http://www.tcpipguide.com

Cet en-tête détaille les éléments à renseigner à la fois dans la requête et à la fois dans la réponse ARP. Dans les deux cas, l'expéditeur doit renseigner son adresse MAC et son adresse IP, ainsi qu'une adresse MAC et une adresse IP pour la cible (attention, pas pour le destinataire.

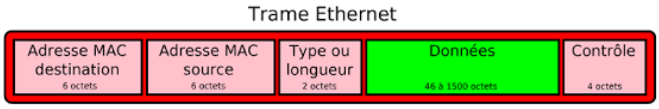
Dans le cas de la requête ARP, l'adresse MAC de la cible n'est pas connue par l'expéditeur de la requête (c'est pourquoi il fait une demande ARP !). À sa place le message contiendra une adresse MAC dédiée : 00 :00 :00 :00 :00 :00, ce qui signale aux receveurs du message que l'adresse MAC est inconnue.

L'encapsulation

La notion d'encapsulation est centrale aux échanges en réseau. Lorsqu'un message doit être transmis à une couche de réseau, il s'intègre dans un protocole de cette couche, puis il est encapsulé à l'envoi à la couche immédiatement inférieure, puis à la couche inférieure à celle-là, etc. jusqu'à la couche 1 (physique). À chaque encapsulation, on rajoute des préfixes et suffixes selon les protocoles utilisés aux couches inférieures.

Par exemple, ARP est un protocole de couche 2 et il est encapsulé dans le protocole Ethernet.

Voici pour rappel l'en-tête Ethernet.



Une message ARP est encapsulé dans une trame Ethernet, c'est-à-dire il sera inséré (avec son en-tête) dans l'espace Données ci-dessus.

Un exemple de requête ARP pourrait être la trame suivante.

```
FF FF FF FF FF FF 08 00 27 76 53 17 08 06 00 01
08 00 06 04 00 01 08 00 27 76 53 17 ac 10 02 ea
00 00 00 00 00 00 ac 10 02 e8
```

L'extérieur de la trame est l'encapsulation Ethernet. Les 6 premiers octets représentent l'adresse MAC du destinataire de cette trame : FF:FF:FF:FF:FF:FF (broadcast MAC). Les 6 octets suivants représentent l'adresse MAC de l'expéditeur : 08:00:27:76:53:17. L'identifiant 08 06 dénote le protocole ARP.

Les données qui suivent sont une encapsulation ARP. La première valeur (00 01) correspond au «Hardware Type» (Ethernet), tandis que la deuxième (08 00) est « protocol Type » : IP. La taille des adresses «Hardware » (c'est-à-dire MAC), est 06. La taille des adresses protocole (IP) est 4. Le type de message ARP est 00 01 (requête) -- ça aurait été 00 02 pour une réponse. Les champs suivants sont :

- L'adresse MAC de l'expéditeur : 08:00:27:76:53:17
- L'adresse IP de l'expéditeur : ac 10 02 ea
- L'adresse MAC de la cible : 00:00:00:00:00:00
- L'adresse IP de la cible : ac 10 02 e8

Comme il s'agit d'une requête ARP, l'adresse MAC de la cible est inconnue.

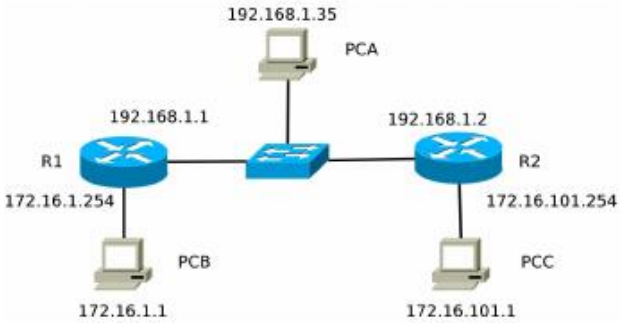
Lors de la réception de cette trame, le récepteur traite premièrement la couche extérieure d'encapsulation pour décider si le message lui est destiné (il regarde l'adresse MAC de destination). Si

c'est le cas, le récepteur vérifie le code de contrôle du message pour évaluer l'intégrité du message reçu, avant d'analyser l'adresse MAC de la source et les données en eux-mêmes.

Pour un message à une couche supérieure, par exemple la couche 3, le message sera premièrement intégré dans le protocole IP. Le paquet IP sera ensuite encapsulé dans le protocole Ethernet.

Exercice 1

Nous allons partir sur la configuration suivante de réseau, pour laquelle les réseaux de PCB et PCA sont en /24. Le réseau de PCA a un CIDR correspondant à sa classe.



1. Analyse de la topologie : écrivez (en notation CIDR) les adresses IP
    a. De toutes les passerelles dans ce réseau
    b. De tous les réseaux

2. Pour la machine PCA nous voulons mettre en place une route vers le réseau de PCB qui passe par R1, et une route vers le réseau de PCC qui passe sur R2.
    a. Ecrivez les commandes qu'il faut taper sur PCA
    b. Quelle sera la table de routage correspondante ?
    c. Pourquoi cette approche a-t-elle des inconvénients par rapport à celle d'une route par défaut sur la machine PCA ?

R2.05-- Les services en réseaux

TD1 : Le fichier interfaces, la couche 4 et l'accès vers Internet

Le fichier /etc/network/interfaces

Jusqu'à présent nous avons utilisé la commande `ip` pour configurer les adresses IP et le routage sur chaque machine. Cependant, mettre en place la configuration de cette façon résulte dans une configuration temporaire, on peut facilement le perdre. Une alternative pour faire une configuration à longue durée est d'utiliser le fichier `/etc/network/interfaces`.

Le fichier `/etc/network/interfaces` permet la configuration à la fois des interfaces réseau (avec des adresses IP statiques ou dynamiques) et à la fois des passerelles à utiliser.

Un exemple d'un fichier `/etc/network/interfaces` qui fait seulement la configuration d'une seule interface (eth0) avec une configuration statique serait :

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.56.11
netmask 255.255.255.0
gateway 192.168.56.1
```

Ce fichier fait la configuration de deux interfaces : l'interface lo de loopback et l'interface eth0. Sur l'interface eth0 nous avons configuré une adresse statique : 192.168.56.11 avec un masque de réseau 255.255.255.0 (quelle est le CIDR en question ?). De plus on indique que la machine 192.168.56.1 sera la passerelle par défaut de la machine en question.

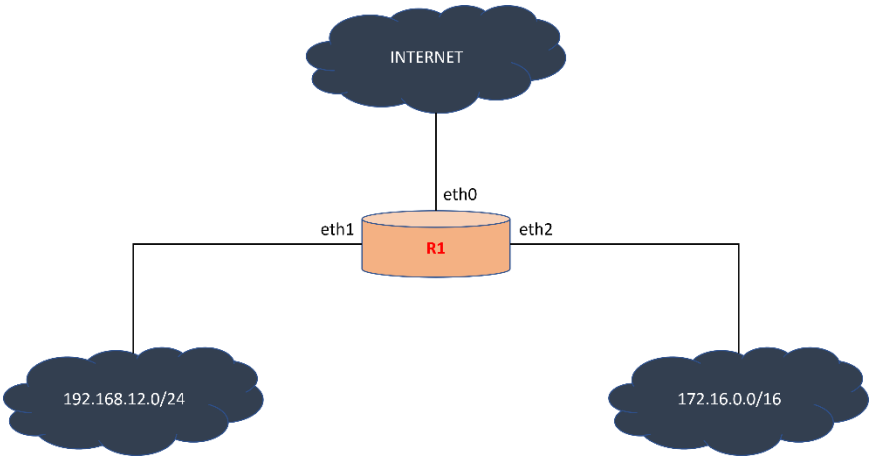
Pour mettre en place les modifications indiquées par le fichier interfaces, il faut utiliser la commande :

```
/etc/init.d/networking restart
```

La masquerade

Souvenez-vous du concept des plages d'adresses privées et publiques en IPv4. Des plages comme 10.0.0.0/8 ou 192.168.0.0/16 sont non-routables. Pourtant, la plupart des réseaux sont configurés avec une majorité des machines utilisant des adresses privées, alors que la passerelle qui achemine les packets envers et à partir du réseau aura une interface dont l'adresse sera routable.

Voici un exemple d'architecture de ce type.



Le routeur R1 a 3 interfaces. Les interfaces eth1 et eth2 auront des adresses IP dans les pages à gauche et à droit respectivement – donc des adresses privées et non-routables. Par contre, l'interface eth0 aura une adresse routable.

Lorsqu'une machine qui se trouve dans les réseaux à gauche ou à droite veut envoyer un message vers Internet, ce message passera par R1. Le packet qui arrive à R1 (pour être envoyé plus loin) sera encapsulé à la couche 3 par le protocole IP et à la couche 2 par le protocole Ethernet, et devra donc renseigner l'adresse IP de la source et du destinataire et l'adresse MAC de la source et du destinataire.

Les adresses MAC changeront à chaque envoi successif, car ces adresses n'indiquent que la source et le destinataire de l'étape actuelle de transmission. Par contre, les adresses IP sont censées indiquer la source originale et la destination finale du packet.

 La source ayant une adresse privée, la transmission n'aboutira pas.

Pour permettre le routage des messages d'un réseau privé vers Internet, le routeur R1 doit utiliser une *masquerade* : lorsque R1 doit envoyer un message de la part d'une machine du réseau privé, il remplace, dans le packet sortant vers Internet, l'adresse privée de la source par une adresse routable (la sienne sur l'interface eth0) et, lorsqu'il reçoit une réponse, il la fait suivre plus loin. Le masquage se fait donc lors de l'envoi sortant de eth0.

 La commande qui permet de faire la masquerade est :

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

La couche transport

La couche transport est la couche 4 du modèle OSI et apporte deux notions par rapport aux couches précédentes :

- La notion de port : les messages encapsulés à cette couche peuvent provenir d'une multitude d'applications (une application web, un email, etc.). Lors de leur encapsulation à la couche 4, chaque message sera envoyé par une machine sur un port spécifique et reçu sur une autre machine sur un autre port.
- La notion de fiabilité : Les deux protocoles standards utilisés à cette couche sont TCP et UDP. Si UDP est un protocole relativement simpliste, qui ne donne garantit aucune fiabilité, le protocole TCP offre une garantie de la réception correcte de messages, dans le bon ordre.

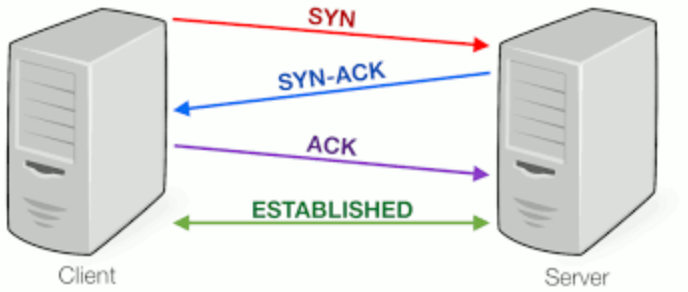
La plupart des protocoles standard de la couche application sont associés à des protocoles de couche transport dédiés : http fonctionne habituellement sur TCP, tandis que le protocole d'adressage dynamique DHCP fonctionne sur UDP.

Le protocole TCP assure la fiabilité des messages en envoyant des confirmations (acknowledgement en anglais) pour chaque message reçu.

Supposons qu'une machine PCA héberge une application web qui fonctionne sur https. Une machine PCB (potentiellement localisée dans un autre réseau) veut se connecter à l'application. Dans ce cas d'usage, PCB joue le rôle d'un client, tandis que PCA est un serveur. Pour assurer un échange de message https, premièrement PCA et PCB établissent une connexion TCP, en effectuant un poignet de main.

Deux machines ne peuvent se connecter en TCP que si le routage permet une connexion bilatérale entre elles.

Le poignet de main TCP consiste de 3 messages. Typiquement, dans un environnement client-serveur, c'est le client qui envoie le premier message : SYN. Ce message a pour but de synchroniser le client et le serveur sur un numéro de séquence (SEQ) du client. Le serveur retourne un message de SYN – qui indique au client un numéro de séquence SEQ qui sera utilisé au départ par le serveur – et un message ACK qui indique le fait que le serveur avait reçu le SYN du client. Finalement, le client envoie un message ACK, qui confirme la réception du SYN du serveur.



Source : https://www.coengoedegebure.com/

SEQ et ACK

TCP assure la fiabilité en accusant la réception de chaque message envoyé. Pour ce faire, les deux participants utilisent des numéros de séquence, qui augmentent avec chaque message, et des numéros d'accusé de réception.

Le client aura un numéro de séquence SEQ, qui comptabilise les messages qu'il envoie au serveur, et un numéro d'accusé de réception ACK, qui comptabilise les messages qu'il reçoit. Le serveur aura également un SEQ qui augmente à chaque envoi de ses messages et ACK qui augmente avec la réception des messages du client. Les deux numéros de séquence, SEQ, du client et du serveur, ne sont pas forcément liées l'un à l'autre ; par contre, le ACK du client est lié au SEQ du serveur et vice-versa.

Supposons que le client envoie au serveur un message avec SEQ = 70. L'ACK du serveur aura ACK = 71, ce qui indique au client : « J'ai bien reçu le message SEQ = 70, le prochain message que j'attends de ta part c'est le message avec un SEQ = 71 ».

Supposons maintenant que le client envoie un message avec SEQ = 70 et ACK = 49. Ceci veut dire que le client vient d'envoyer le message dont le numéro de séquence d'envoi est 70 (cela peut être le 70ème message mais aussi le 30ème si le compteur a démarré à 40) ; le client attend le message 49 du serveur. Le message logique avec lequel le serveur doit répondre serait le message avec SEQ = 49 et un ACK = 71 (il a reçu le message 70 et attend le message 71).

L'accusé de réception (ACK)

Le protocole TCP peut être utilisé pour l'encapsulation à la couche transport de n'importe quel message envoyé à une couche supérieure. Par exemple, les messages du protocole http (couche 7) sont encapsulés dans TCP ; toutefois, un message de type ping (couche 3) ne peut pas l'être.

Supposons une connexion https entre une machine client avec l'adresse IP 125.22.91.2 et un serveur web hébergé sur une machine avec adresse IP 164.81.110.29. Comme le protocole https (protocole de couche 7) est encapsulé à la couche 4 par le protocole TCP (le port standard étant 443), la connexion https aura lieu seulement après un poignet-de-main (handshake) TCP. Les messages https dans les deux directions seront toujours suivis par un ACK sur TCP.

Pourquoi sur TCP et pas sur https ?

Tout simplement car la fiabilité d'un protocole encapsulé sur TCP est assurée par TCP, indépendamment du protocole de la couche supérieure.

Les protocoles client-serveur

La plupart des protocoles de couche 7 que nous allons étudier en réseau sont des protocoles client-serveur. C'est le cas par exemple de http, https, smtp, smtps, ssh, ftp, etc.

Lorsqu'une machine héberge un serveur, celui-ci est en permanence à l'écoute (état : LISTEN). Au contraire, le client se connecte au serveur d'une façon ponctuelle.

Un protocole client-réseau est encapsulé par l'un des deux protocoles de la couche transport : TCP ou UDP. Typiquement, le serveur utilise un protocole standard de couche 4 et un port standard pour l'écoute, même si, en fonction de la configuration du service, un autre port peut, exceptionnellement, être utilisé.

Voici quelques exemples de ports standard :

- http, resp. https : TCP port 80, resp. 443
- telnet : TCP port 23
- ftp resp. sftp, resp. ftps : TCP port 21, resp. 22, resp. 990
- dns : TCP ou UDP, port 53
- dhcp : UDP port 67 pour le serveur, UDP port 68 pour le client

Un client ne peut joindre un service hébergé sur un serveur que si le serveur est à l'écoute sur le port recherché par le client. Si le protocole client-serveur est encapsulé par TCP, alors le client et serveur commencent par établir une connexion TCP et ensuite, une fois la connexion ouverte, des messages plus spécifiques aux protocoles client-serveur peuvent passer.

Nous allons apprendre comment démarrer et arrêter un serveur, puis comment vérifier l'état d'écoute d'un serveur lors du prochain TD.

La résolution de nom

En réseau, chaque machine est identifiable à partir de son adresse IP et de son adresse MAC. En revanche, un utilisateur aurait du mal à identifier son correspondent de cette manière. En conséquence, les machines qui fournissent des services utilisent des noms plus facilement identifiables, comme par exemple `mail.google.com`, `www.yahoo.fr`, etc.

Pour qu'une machine associe `mail.google.com` à une machine destinataire, il faut un mécanisme qui trouve l'association entre ce nom de domaine et une adresse IP. A l'inverse, on doit également pouvoir reconnaître le nom d'une machine à partir de son adresse IP. Une bonne parallèle à cette situation est celle d'un téléphone. On ne dit pas « Je vais appeler le 06 78 90 12 34 », sinon « Je vais appeler Jean ». Nous allons ensuite chercher dans la liste de contacts jusqu'à trouver le nom du correspondant. En revanche le téléphone, lui, stocke l'association entre le nom du contact « Jean » et son numéro de téléphone. A l'inverse, lorsque Jean nous appelle, le téléphone trouve l'association inverse : à partir de son numéro de téléphone, on identifie Jean.

Trouver une adresse IP à partir d'un nom s'appelle la résolution directe de nom. Si on veut trouver un nom à partir d'une adresse IP, il s'agit d'une résolution inverse.

Une résolution statique de noms peut être réalisée (sur des petits réseaux, en utilisant le fichier */etc/hosts/.* Ce fichier stocke des correspondances spécifiques entre certains noms de domaines et certaines adresses IP. Le besoin de rajouter et de mettre à jour chaque association machine – adresse IP rend l'utilisation du fichier */etc/hosts* inefficace et lente à grande échelle, mais elle peut devenir pratique pour des petits réseaux.

Un fichier `hosts` pourrait avoir le contenu suivant :

```
127.0.0.1       localhost
192.168.0.22    IUTINFO-VPN-04.iut.unilim.fr  IUTINFO-VPN-04
192.168.0.5     hercule hercule.iut.unilim.fr

# The following lines are desirable for IPv6 capable hosts
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters
```

Vous pouvez remarquer qu'on a mis dans ce fichier l'adresse de loopback, ainsi que deux autres associations pour des machines spécifiques.

Une façon alternative – distribuée et dynamique – d'assurer la résolution de nom sur une machine est d'utiliser le protocole DNS. Le protocole DNS est un protocole client-serveur : lorsqu'une machine (client) veut demander une résolution de nom, elle fait une requête DNS à une autre machine qui héberge un serveur DNS. Le protocole DNS fonctionne à la couche 7 du modèle OSI et peut utiliser, à la couche transport, soit le protocole TCP soit UDP, sur le port 53.

Dans ce deuxième cas notre machine jouera le rôle d'un client DNS, utilisant un `resolver` pour se connecter à un serveur DNS

L'utilisation d'un serveur DNS

Une méthode plus centralisée et dynamique de résolution est l'utilisation d'un serveur DNS. Nous verrons comment mettre en place un serveur DNS dans la deuxième année. Dans ce module nous allons seulement traiter la machine client, qui s'appelle un resolver. Le resolver est définit dans le fichier */etc/resolv.conf*. Ce fichier stocke l'information nécessaire pour faire la résolution de nom : notamment là on stocke l'adresse IP d'au moins un serveur DNS qui peut être contacté en cas de besoin.

Un contenu possible du fichier */etc/resolv.conf* serait :

```
search unilim.fr
nameserver 164.81.1.4
nameserver 164.81.1.5
```

Dans la figure ci-dessus on indique premièrement le domaine dans lequel une machine se trouve-t-elle. Les deux lignes suivantes représentent chacune l'adresse d'un serveur DNS. Elles sont parcourues dans l'ordre citée : premièrement le resolver essaie à effectuer la résolution de nom avec la machine 164.81.1.4, puis, si cela ne marche pas, elle ressaie avec la machine 164.81.1.5.

La résolution directe et la résolution inverse

Pour faciliter la connexion entre deux machines, au lieu d'utiliser son adresse IP, on utilise plutôt un nom associé. Le service DNS s'occupe de l'association dans les deux sens :

1. **La résolution directe** : à partir d'un nom, trouver une adresse IP
2. **La résolution inverse** : à partir d'une adresse IP, trouver le nom de la machine

Dans un petit réseau la résolution directe et inverse peut se réaliser en modifiant à la main le fichier ***hosts***. A grande échelle, toutefois, l'utilisation du fichier ***hosts*** n'est pas pratique. Sur les grands réseaux on utilise plutôt le système ***Domain Name System (DNS)***.

Il est essentiel que toutes les machines utilisent la même association entre les noms et les adresses IP. De plus cette association doit être mise à jour régulièrement. C'est pourquoi une organisation mondiale, l'IANA, est chargée de gérer les noms de domaine. Elle délègue une partie de ces responsabilités à quelques organisations comme par exemple NIC France, RIPE, etc.

Dans ce cours, nous n'allons pas trop approfondir ces notions ; toutefois, nous allons voir quelques notions de base concernant les noms de domaines et la configuration d'un serveur DNS en vue d'une résolution directe.

TD2 : L'acquisition dynamique des adresses, DHCP, SS

L'adressage IP dynamique

La configuration statique des adresses IP de divers périphériques peut être très pratique, particulièrement dans un petit réseau qui probablement changera très peu avec le temps -- par exemple un réseau privé qu'on met en place à la maison.

Toutefois, une configuration statique est incompatible avec la mobilité des usagers d'aujourd'hui. C'est pourquoi nous avons la possibilité d'utiliser DHCP (Dynamic Host Configuration Protocol), qui fait en sorte qu'une machine (jouant le rôle d'un client) puisse demander dynamiquement une adresse IP à une autre machine (jouant le rôle d'un serveur DHCP).

Le protocole DHCP est un protocole client-serveur de couche 7 (Application), encapsulé à la couche 4 (Transport) par le protocole UDP. Le client utilise le port standard UDP 68 et le serveur, le port UDP 67.

Le serveur DHCP est une machine configurée dans un réseau hôte. La configuration de ce serveur spécifie des détails comme par exemple :

- une plage d'adresses IP qui peuvent être utilisées pour les machines client (visitant le réseau hôte)
- une durée maximale et moyenne de location d'une adresse
- un routeur par défaut que les machines client pourront utiliser dans ce réseau
- un serveur DNS que les machines client peuvent utiliser dans le réseau

À chaque fois qu'une nouvelle machine lui demande une adresse temporaire, le serveur DHCP lui offre une adresse disponible, parmi la plage d'adresses disponibles. De plus, le serveur fournit un masque de réseau, un nom de domaine, les noms ou adresses IP des serveurs DNS et le nom ou adresse IP de la passerelle à utiliser. Ainsi, la nouvelle machine pourra joindre l'Internet dans le nouveau réseau.

L'adresse prêtée est bloquée tout au long de son utilisation -- par contre lorsqu'elle ne sera plus nécessaire l'adresse pourra être remise en circulation et donnée à une autre machine.

Bien que l'utilisation de DHCP pour l'adressage dynamique soit très pratique, le système ne fonctionne que s'il y a un serveur DHCP qui peut attribuer des adresses aux machines entrant dans le réseau.

Si un serveur DHCP est en panne, un deuxième serveur peut bien être mise en place. Cependant il va falloir faire attention à la configuration de ce deuxième serveur, car si les deux serveurs fonctionnent en même temps, il peut y avoir des conflits d'adressage, ce que va paralyser le réseau.

Attribution fixe et dynamique

Le fonctionnement du protocole DHCP a été déjà expliqué en CM. On se rappelle que lorsqu'une machine fait une demande au serveur pour qu'on lui attribue une adresse IP, le client lui envoie son adresse MAC. Le serveur DHCP peut alors faire une attribution fixe ou une attribution dynamique.

Avec une attribution fixe, à chaque fois qu'une certaine machine se connecte, on lui attribue la même adresse IP, qu'on réserve exclusivement pour cette machine. On différencie entre les machines par leurs adresses MAC.

A quoi peut servir une telle attribution ?

Une bonne pratique est de garder les mêmes adresses IP pour les machines dont on se sert souvent (passerelles, imprimantes, etc.) ou pour maintenir un contrôle d'accès plus fin.

Par défaut, l'attribution d'adresse en DHCP se fait toutefois dynamiquement : à chaque nouvelle demande d'attribution, une machine aura une nouvelle adresse IP, choisie sur une plage donnée en fonction des disponibilités actuelles.

Dans les deux cas, l'adresse donnée par le serveur et acceptée par la machine ne lui sera attribuée que pour une durée fixe, décidée par le serveur. La machine peut garder son adresse IP si au bout de la durée du bail celui est renégocié. Sur le serveur, les baux sont conservés dans un fichier actualisé en permanence, et trouvable dans le dossier */var/dhcp*.

```
lease 192.168.1.11 {
  starts 3 2012/10/24 08:53:10;
  ends 3 2012/10/24 09:03:10;
  cltt 3 2012/10/24 08:53:10;
  binding state active;
  next binding state free;
  hardware ethernet 2e:18:cc:d4:8c:e7;
}
lease 192.168.1.10 {
  starts 3 2012/10/24 08:53:53;
```

Configuration DHCP sous Linux

Lors du prochain TP nous allons configurer un serveur DHCP sous Linux. Lors du TP vous allez installer la version dans le paquet *`isc-dhcp-server`*. La configuration du serveur est retenue dans le fichier */etc/dhcp/dhcpd.conf* (mais l'emplacement de ce fichier peut être différent sur une autre version du serveur DHCP ou sur une autre distribution de Linux).

Un exemple de fichier dhcpd.conf est donné dans la suite :

```
ddns-update-style none ;

subnet 137.12.84.0 netmask 255.255.255.0
{
      range 137.12.84.15  137.12.84.254 ;
      default-lease time 21600 ;
      max-lease-time 43200 ;
      option routers 137.12.84.10 ;
      option domain-name-servers
137.12.84.11 ;
}

host pc1 {
     hardware ethernet 00:4A:5C:24:3E:FF ;
     fixed-address 137.12.84.20 ;
}
```

Ce fichier spécifie que la plage d'adresses disponibles pour DHCP est de 137.12.84.15 à 137.12.84.254, dans le réseau 137.12.84.0/24. Par défaut une seule adresse sera louée pour 21600 secondes (=360 minutes), jusqu'à une durée maximale de 43200 secondes (=720 minutes). On indique que le routeur par défaut qui sera indiquée en DHCP (donc le routeur par défaut des machines qui demanderont une adresse dynamiquement) à l'adresse IP 137.12.84.10, tandis que le serveur de noms alloué à l'adresse 137.12.84.11.

Il n'est pas obligatoire d'indiquer un routeur par défaut, ni un serveur DNS, aux machines entrant dans le réseau ; cependant, sans routeur, les machines seront isolées dans le réseau.

Nous pouvons également indiquer pour la machine du client qu'elle doit demander son adresse IP sur DHCP. Ceci peut être indiqué dans le fichier de configuration IP d'une machine, notamment */etc/network/interfaces*. Disons qu'on veut indiquer, pour une certaine machine, que sur son interface eth1 elle doit louer des adresses IP sur DHCP. Alors dans le fichier de configuration, on configure cette interface comme ci-dessous :

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.56.11
netmask 255.255.255.0
gateway 192.168.56.1

auto eth1
iface eth1 inet dhcp
```

Maintenant, l'interface eth1 prendra son adresse IP dynamiquement, en utilisant le protocole DHCP.

L'installation côté serveur

Dans ce TP vous allez télécharger un fichier qui va vous permettre d'installer un service de DHCP sur votre machine. Une fois le paquet téléchargé, il faut l'installer en utilisant la commande :

```
apt install isc-dhcp-server
```

Une fois le serveur bien installé et la configuration réalisée, cette dernière peut être vérifiée en utilisant la commande `dhcpd -t`.

Une fois la configuration correcte, pour démarrer et arrêter ce service on peut utiliser un nombre de commandes :

- Démarrage (3 alternatives) :
  - `systemctl start isc-dhcp-server`
  - `service isc-dhcp-server start`
  - `/etc/init.d/dhcp start`
- Arrêt (3 alternatives) :
  - `systemctl stop isc-dhcp-server`
  - `service isc-dhcp-server stop`
  - `/etc/init.d/dhcp stop`
- Redémarrage (3 alternatives) :
  - `systemctl restart isc-dhcp-server`
  - `service isc-dhcp-server restart`
  - `/etc/init.d/dhcp restart`

Une fois le service DHCP vérifié et démarré, un bon réflexe est de vérifier que le serveur est à l'écoute. Ceci se fait en utilisant la commande ss.

L'installation côté client

Pour s'assurer qu'un client puisse demander une adresse IP, il faut premièrement mettre en place une configuration IP lui permettant de demander des adresses IP dynamiquement. Ceci se fait dans le fichier etc/network/interfaces -- voire le TD.

Une fois la configuration mise en place, on peut ensuite demander une nouvelle adresse IP en utilisant la commande `dhclient <interface sur laquelle on fait du DHCP>`. Également utiles sont les commandes :

- Renoncer à une adresse : `dhclient -r <interface sur laquelle on fait du DHCP>`
- Refaire une demande DHCP: `dhclient -v <interface sur laquelle on fait du DHCP>`

Les commandes ss et ps

Lorsqu'une machine utilise un service réseau (HTTP, FTP, etc.), une connexion est établie entre cette machine (jouant le rôle d'un client) et un serveur. Pour vérifier l'état actuel de la connexion, ainsi que les coordonnées du serveur, nous pouvons utiliser la commande `ss`, qui vérifie l'état socket shell Celle-ci est d'autant de plus importante dans le contexte d'un serveur, pour lequel on doit souvent vérifier s'il est toujours à l'écoute.

La commande `ss`, sert à afficher les connections entre la machine sur laquelle on tape la commande et toute autre machine. Dans le cas d'une machine active, comme par exemple un serveur, la commande ss peut retourner un nombre important de connections, qui sont peut-être moins intéressantes. C'est pourquoi c'est mieux d'utiliser la commande ss avec les options appropriées.

La syntaxe d'utilisation sera alors :

```
ss -<options sans séparation>
```

Par exemple la commande netstat -nt retourne les connections sur TCP (option -t) en mettant les adresses et numéros de ports en des valeurs décimales (option -n).

Voici une liste possible d'options intéressantes à utiliser.

`-a` : retourne toutes les connexions et ports d'écoute

`-n` : retourne les ports en format numérique

`-l` : retourne seulement les ports sur lesquels la machine écoute

`-t` et `-u` : donne seulement les connexions sur TCP (option -t) ou UDP (option -u)

Si on veut peaufiner encore plus les résultats, par exemple pour trier les résultats par un mot clé connu, on peut utiliser la commande ss avec la commande grep, en utilisant la syntaxe suivante :

```
ss -<options sans séparation> | grep <mot clé>
```

Attention : si on utilise la syntaxe ci-dessus, alors la recherche donnée par grep s'effectuera strictement sur le contenu affiché par la commande ss avec ses options. Par exemple, le port 443 sur TCP correspond à une connexion HTTPS. Si on utilise `ss -nta | grep 443` , cela nous retournera toute connexion HTTPS sur la machine donnée. En revanche, utiliser la commande `ss -nta | grep` HTTPS ne nous retournera aucun résultat, car les noms des protocoles utilisés ne sont pas affichés par `ss -nta`.

La commande ss est essentielle pour la mise en place des services en réseaux. Vous pouvez également trouver plus d'infos ici : https://www.linux.com/topic/networking/introduction-ss-command/

Si on veut trier les résultats par protocole utilisé, on peut utiliser la commande `ps`.

La commande `ps` retourne de l'information sur une sélection de procès actives. Cette commande existe dans plusieurs systèmes d'opération, d'où plusieurs styles pour l'écriture des options possibles. Cependant, les options le plus souvent utilisées sont :

a : retourne les procès de tout usager

u : retourne une colonne utilisateur/propriétaire (owner)

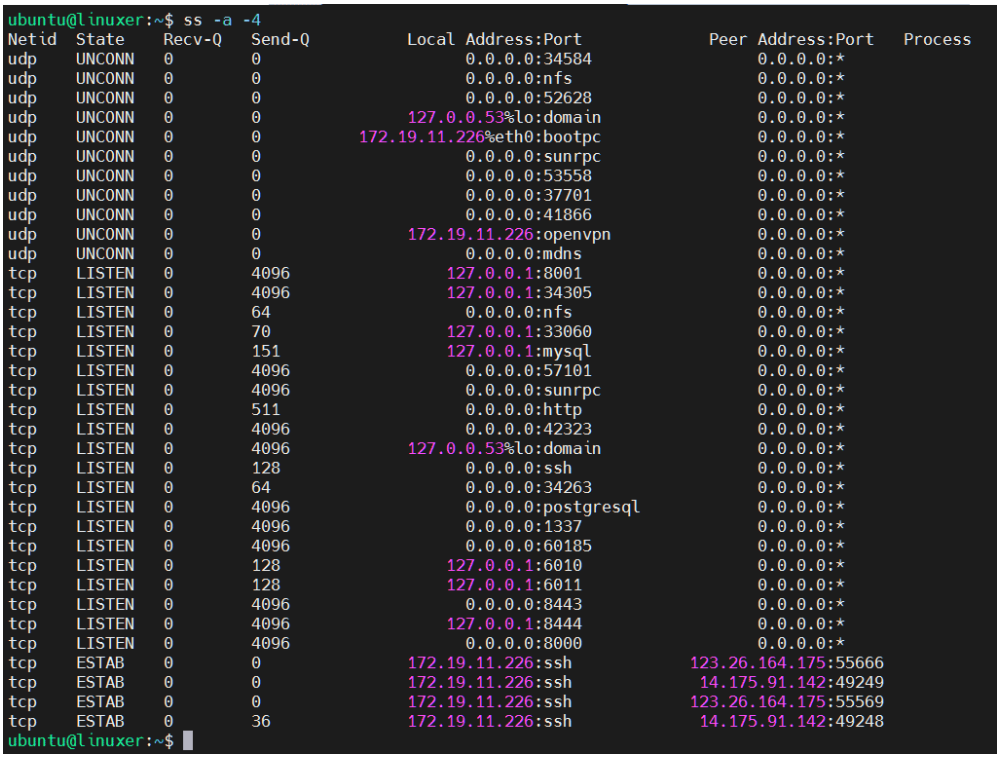
x : retourne tous les procès qui ne sont pas exécutés dans le terminal

Ensemble, ces trois options donnent le raccourci `ps aux`.

Nous pouvons encore filtrer les résultats en utilisant la commande grep, de la même façon que pour la commande ss.

Connexions et états des sessions

Voici un exemple de retour pour la commande ss sur une machine :



```
ubuntu@linuxer:~$ ss -a -4
Netid  State    Recv-Q   Send-Q          Local Address:Port             Peer Address:Port   Process
udp    UNCONN   0        0                     0.0.0.0:34584                 0.0.0.0:*
udp    UNCONN   0        0                     0.0.0.0:nfs                   0.0.0.0:*
udp    UNCONN   0        0                     0.0.0.0:52628                 0.0.0.0:*
udp    UNCONN   0        0                127.0.0.53%lo:domain               0.0.0.0:*
udp    UNCONN   0        0          172.19.11.226%eth0:bootpc                0.0.0.0:*
udp    UNCONN   0        0                     0.0.0.0:sunrpc                0.0.0.0:*
udp    UNCONN   0        0                     0.0.0.0:53558                 0.0.0.0:*
udp    UNCONN   0        0                     0.0.0.0:37701                 0.0.0.0:*
udp    UNCONN   0        0                     0.0.0.0:41866                 0.0.0.0:*
udp    UNCONN   0        0                172.19.11.226:openvpn              0.0.0.0:*
udp    UNCONN   0        0                     0.0.0.0:mdns                  0.0.0.0:*
tcp    LISTEN   0        4096              127.0.0.1:8001                   0.0.0.0:*
tcp    LISTEN   0        4096              127.0.0.1:34305                  0.0.0.0:*
tcp    LISTEN   0        64                    0.0.0.0:nfs                   0.0.0.0:*
tcp    LISTEN   0        70                127.0.0.1:33060                  0.0.0.0:*
tcp    LISTEN   0        151               127.0.0.1:mysql                  0.0.0.0:*
tcp    LISTEN   0        4096                  0.0.0.0:57101                 0.0.0.0:*
tcp    LISTEN   0        4096                  0.0.0.0:sunrpc                0.0.0.0:*
tcp    LISTEN   0        511                   0.0.0.0:http                  0.0.0.0:*
tcp    LISTEN   0        4096                  0.0.0.0:42323                 0.0.0.0:*
tcp    LISTEN   0        4096             127.0.0.53%lo:domain               0.0.0.0:*
tcp    LISTEN   0        128                   0.0.0.0:ssh                   0.0.0.0:*
tcp    LISTEN   0        64                    0.0.0.0:34263                 0.0.0.0:*
tcp    LISTEN   0        4096                  0.0.0.0:postgresql            0.0.0.0:*
tcp    LISTEN   0        4096                  0.0.0.0:1337                  0.0.0.0:*
tcp    LISTEN   0        4096                  0.0.0.0:60185                 0.0.0.0:*
tcp    LISTEN   0        128               127.0.0.1:6010                   0.0.0.0:*
tcp    LISTEN   0        128               127.0.0.1:6011                   0.0.0.0:*
tcp    LISTEN   0        4096                  0.0.0.0:8443                  0.0.0.0:*
tcp    LISTEN   0        4096              127.0.0.1:8444                   0.0.0.0:*
tcp    LISTEN   0        4096                  0.0.0.0:8000                  0.0.0.0:*
tcp    ESTAB    0        0                172.19.11.226:ssh          123.26.164.175:55666
tcp    ESTAB    0        0                172.19.11.226:ssh           14.175.91.142:49249
tcp    ESTAB    0        0                172.19.11.226:ssh          123.26.164.175:55569
tcp    ESTAB    0        36               172.19.11.226:ssh           14.175.91.142:49248
ubuntu@linuxer:~$
```

La commande a été accompagnée par deux paramètres : -a -4 . Le premier se réfère au paramètre « all », alors que le deuxième limite la recherche ss aux adresses IPv4 (si on veut faire la même chose en IPv6, il faut changer le paramètre à -6).

L'affichage contient des connexions sur tcp et udp (première colonne) dans différents états :

- UNCONN : unconnected (aucune connexion sur ce port)
- LISTEN : écoute (le serveur est à l'écoute sur ce port)

- ESTAB : établie (la connexion est établie).

D'autres états existent, comme par exemple TIME-WAIT (la connexion est perdue et en train d'être clôturée), LAST-ACK (on attend un dernier ACK avant de partir), etc.

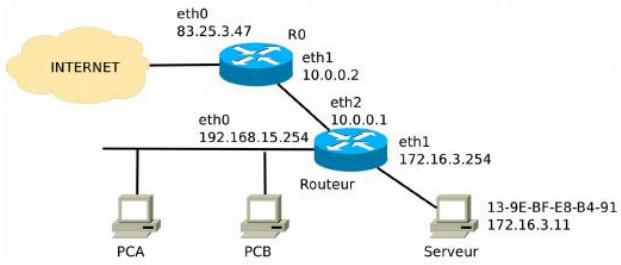
Les colonnes Recv-Q et Send-Q se réfèrent au nombre de packets reçus et envoyés sur la connexion en question.

Les deux colonnes suivantes indiquent les deux bouts de la communication : Local Address correspond à l'adresse de la machine sur laquelle on a tapé la commande ss, alors que Peer Addresse correspond à l'adresse de son partenaire. Chaque adresse est suivie par : <port>. Comme la commande ss a été appelée avec en paramètre -a, toutefois, les ports sont déjà affichés par leurs services correspondants : domain = DNS, port 53 ; ssh = port 22, etc.

La figure affiche premièrement un nombre de possibilités de connexion en UDP sur la machine, puis un nombre de serveurs qui sont à l'écoute sur TCP, et finalement 4 connexions établies. On note que chacune des machines Peer 123.26.164.175 et 14.175.91.142 a deux connexions ouvertes vers le serveur hébergé sur la machine locale, dont l'adresse IP est 172.19.11.226.

Exercice 1

Cet exercice concerne le réseau dont la topologie est dans la figure ci-dessous :



1. Indiquez les réseaux présents dans cette figure, avec les machines qui en font partie.
2. Donnez le contenu du fichier interfaces qui permet la machine appelée Routeur d'avoir la configuration dans la figure ci-dessus.
3. La machine appelée Routeur joue aussi le rôle d'un serveur DHCP pour le sous-réseau des machines PC A et PC B et pour celui de la machine Serveur. Pour les deux, la plage disponible est 10-200.
    a. Quel fichier faut-il modifier pour mettre cela en place ?
    b. Quel sera le contenu de ce fichier ?

4. Nous voulons que le serveur DHCP accorde toujours au Serveur la même adresse IP, notamment 172.16.3.11. Comment faut-il modifier le fichier de l'exercice précédent ?

TD3 : Cas d'usage : développement et réseaux

Les protocoles HTTP et HTTPS

A sa base, l'Internet se compose de deux composants :

1. **Le protocole HTTP** : pour le transfert d'informations en formes de données
2. **DNS** : une arborescence de noms de domaine, qui permet l'accès par nom plutôt que par IP

Le fonctionnement actuel des sites web a évolué avec le temps, les technologies qui permettent de mettre en place le transfert de données ont changé également ; malgré cela, le fonctionnement de l'Internet n'a pas vraiment changé d'un point de vue des réseaux.

Le protocole http fonctionne dans une architecture **client-serveur** à la couche 7 OSI, et est encapsulé à la couche 4 par TCP. Le serveur http utilise le port standard 80 tandis que https utilise le port 443.

Le client HTTP est le navigateur qu'on utilise. Le processus serveur se trouve sur un serveur Apache ou IIS (Internet Information Services).

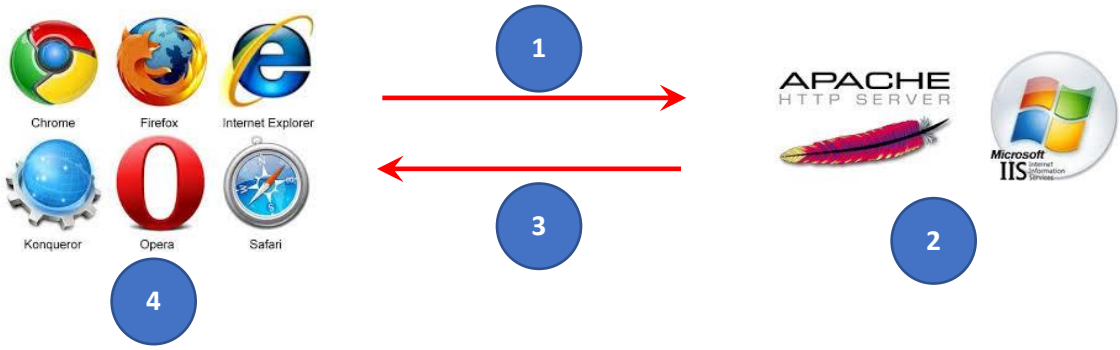


Figure 1 : le fonctionnement de HTTP

Le protocole HTTP se déroule en quatre étapes, montrées aussi dans la Figure 1 :

1. Le client initie une connexion TCP/IP. Puis il demande un document au serveur en langage HTTP.
2. Le serveur cherche le document demandé ou peut le générer en utilisant un script PHP
3. Le serveur envoie le document sur la connexion TCP/IP
4. Le client reçoit le document. En interprétant le langage HTML/CCS, le client peut la rendre dans un document visuel affiché dans le navigateur.

1. La syntaxe URL

Les informations cherchées par le client peuvent avoir plusieurs formes et être accédées de différentes façons. Elles peuvent être accueillies sur des machines différents.

C'est pourquoi toute ressource disponible sur le Web est référencée par le biais d'une ***Uniform Ressource Locator*** (URL). Une URL est un type spécial de URI (***Uniform Ressource Identifier***).

Une ressource peut être un document HTML, mais également une image, une Applet Java, un fichier qui doit être transféré. En tant que référence unique à une ressource, un URL inclut :

- ***Un préfix*** constant : "URL`:`" ;
- Un mode d'accès à la ressource, appelé également un ***schéma URI*** -- `http, ftp, mailto, irc`, etc. On peut enregistrer un schéma auprès de la IANA -- Internet Assigned Numbers Authority. Le schéma est suivi par un ":" ;
- La partie ***protocoles Internet***, qui commence par "//" et finit par "/". Les éléments de cette partie incluent :
    - (optionnellement) Un ***nom d'utilisateur***, si besoin. Cette option est utile lorsque la ressource se trouve par exemple sur un serveur FTP. Nous pouvons rajouter un mot de passe, en le séparant par un ";" du nom d'utilisateur. A la fin de cette section il faut ajouter un "@" ;
    - Un ***nom de machine***. De préférence ceci est le nom du domaine de la machine dans le format donné par le standard RFC1037. Parfois on peut avoir une adresse IP plutôt qu'un nom de domaine, mas ceci est moins encouragé ;
    - Un ***numéro de port***. Cette valeur-ci est obligatoire si on veut que l'utilisateur joigne la machine par un port différent à celui qui est standard pour le schéma donné ;
    - Un ***chemin***. Le chemin indique où la ressource se trouve-t-elle dans le domaine indiqué ;
    - (optionnellement) Une ***requête de type chaine de caractères*** (query string). Cette partie suit le chemin. Le… commence par un "?" et consiste en des tuples nom et valeur, séparés par des "&". Par exemple "?term=bluebird&source=browser-search" ;

2. Les protocoles HTTP et HTTPS

Le transfert de données peut s'effectue d'une façon non-sécurisée (en http) ou d'une façon sécurisée (en https). Nous allons premièrement regarder la variante non-sécurisée, puis expliquer le fonctionnement du protocole sécurisé.

2.1 Le protocole HTTP

Le protocole HTTP se déroule sur TCP à la couche transport. Ce protocole représente un dialogue entre un client et un serveur Web. Les échanges entre ces deux parties se concrétisent dans des requêtes et réponses.

Plusieurs versions du protocole HTTP existent.

- **HTTP V0.9** : la première version de HTTP documentée, publiée en 1991. Cette version est actuellement obsolète.
- **HTTP V1.0** : décrit par le RFC 1945.
- **HTTP/1.1** : défini actuellement en RFC 7230.
- **HTTP/2** : défini en 2015 dans le RFC 7540. Cette version modifie la façon dont les données sont transportées entre un client et un serveur. De plus, elle permet également le fonctionnement avec TLS (version 1.2 ou ultérieure) pour les URI en HTTPS. Actuellement, autour de 41% des sites Web fonctionnent en HTTP/2 (Source : w3techs.com).
- **HTTP/3** : HTTP over QUIC. Cette version utilise UDP plutôt que TCP. En septembre 2019 Cloudflare et Chrome ont rajouté du support pour cette version.

Si nous regardons la communication entre le client et le serveur, les premiers messages sont ceux qui établissent la connexion TCP. Ensuite il y aura une négociation de version du protocole. Puis, le client va proposer une version de HTTP qu'il souhaite utiliser et le serveur va confirmer si cette version est acceptable ou non. Le client et le serveur vont échanger des requêtes et des réponses qui seront envoyés en clair. Finalement, la connexion TCP sera fermée.

Quelques commandes intéressantes HTTP sont :

- GET : permet de récupérer un document sur HTTP
- HEAD : permet de ne récupérer que les informations concernant une ressource
- OPTIONS : récupère des informations concernant le serveur HTTP
- PUT : permet d'ajouter ou de remplacer une ressource
- DELETE : supprime une ressource

2.2 Le protocole HTTPS

Lorsqu'on utilise le protocole HTTP, le contenu du dialogue entre le client et le serveur sont publiquement visibles.

Si on veut que ces contenus soient confidentiels, il faut soit utiliser un filtrage en réseau pour limiter l'accès, soit utiliser un protocole qui permet le chiffrement des messages, comme par exemple http sur TLS (HTTPS).

Le client et le serveur commencent en établissant une connexion TCP. Puis, les deux parties vont négocier une session du protocole Transport Layer Security (TLS). Les messages de TLS sont échangés en clair sur TCP. Une fois la connexion TLS négociée, les messages HTTP que le client et le serveur échangent seront chiffrés et authentifiés en utilisant des outils cryptographiques.

L'échange de clé authentifié -- son rôle et son fonctionnement

Le protocole TLS est un exemple de protocole d'échange de clé authentifié. D'autres exemples très utilisés aujourd'hui sont SSH (utilisé pour sécuriser l'accès à distance à une machine) et IPSec (utilisé principalement pour sécuriser la connexion par VPN).

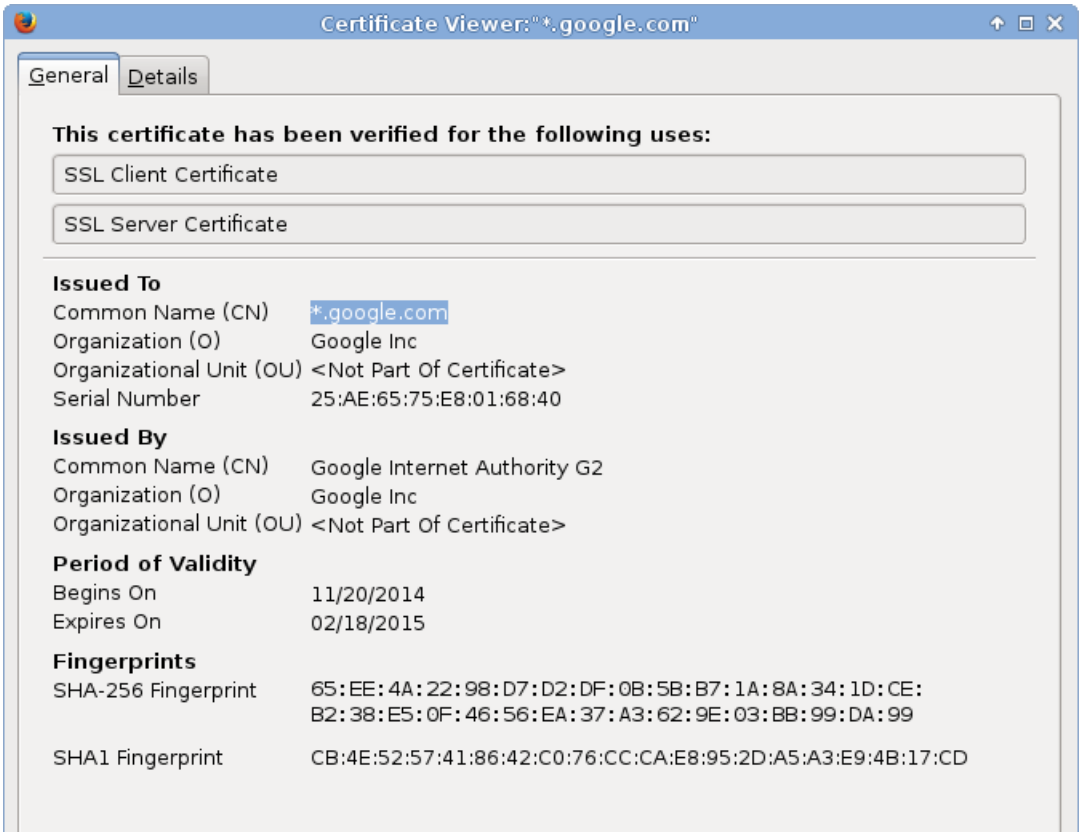
Ces protocoles fonctionnent dans un modèle client-serveur. Pour TLS, le client HTTP est souvent également le client TLS.

Le protocole marche en deux temps : premièrement le client et le serveur échangent des messages en clair pour établir une clé secrète, et puis les deux parties utilisent cette clé secrète pour chiffrer et authentifier leurs communications.

Le fonctionnement du protocole TLS est difficile à expliquer -- on le verra en plus de détail en BUT 2. Toutefois, nous allons explorer quelques éléments de ce protocole ici.

Dans la première partie du protocole TLS, ainsi appelée un poignet de main (handshake) le client et les serveurs s'envoient des messages qui peuvent être interceptés par une entité écoutant sur la ligne (qu'on appelle Man-in-the-Middle -- MitM -- ou Person-in-the-Middle --PitM). La « magie » du protocole consiste en permettant le client et le serveur de calculer une clé inconnue par le MitM malgré le fait que chaque composante de l'échange de clé soit mise à la disposition de l'attaquant.

Un aspect subtil du protocole est l'authentification. Pour que le protocole marche de façon sécurisée, il est essentiel qu'au moins le serveur authentifie les messages qu'il envoie. A ce but il aura besoin d'une paire de clés : une clé privée et une clé publique, par exemple une clé de signatures. En signant une partie de ses messages avec sa clé secrète, il convaincra le client qu'il est bien un serveur légitime. Dans notre cas, le client doit vérifier la signature du serveur en utilisant sa clé publique.



Pour s'assurer que la clé publique est associée au serveur cherché, ce dernier présentera un certificat établi par une autorité qui garantit cette association. Le navigateur vérifie les certificats à chaque connexion. Il faut surtout ne jamais faire confiance à un serveur sans un certificat valide.

Le chiffrement authentifié

A la fin du poignet de main (handshake) le client et le serveur calculent un nombre de clés de session. Ces clés sont connues seulement par les deux bouts de la communication. Dans la deuxième partie du protocole, le client et le serveur utiliseront ces clés pour sécuriser leur communication en utilisant un schéma de chiffrement authentifié. On appelle ce type de schéma « à clé symétrique » : notamment, on chiffre et on déchiffre avec la même clé.

Si la clé générée est « bonne » (d'une bonne taille et indistinguable d'une clé aléatoire de la même taille), alors les propriétés garanties par ces schémas sont :

- La confidentialité : Le contenu des messages échangés reste caché par rapport à un attaquant
- L'authenticité : Le receveur d'un message peut être sûr de l'identité de son envoyeur
- L'intégrité : Le receveur d'un message est assuré que le message n'a pas été modifié

3. La configuration d'un serveur Apache

Les fichiers de configuration

Apache est le serveur HTTP le plus utilisé sur le Web aujourd'hui. C'est un logiciel libre, distribué sous une licence particulière, dont la version actuelle est la version 2.5. Étant très utilisé, Apache est amplement documenté sur Internet, même en langue française.

Le serveur Apache peut se démarrer/arrêter sous Linux comme n'importe quel serveur en utilisant les commandes :

```
/etc/init.d/apache2 <start|stop|status|restart>
service apache2 <start|stop|status|restart>
```

Les fichiers de configuration se trouvent en général dans le dossier ***/etc/apache2*** sous Linux. Selon les versions, on va trouver un seul ou plusieurs fichiers de configuration principaux, qui peuvent faire appel à un certain nombre de modules optionnels. Du chargement ou non de ces modules vont dépendre les fonctionnalités proposées par le serveur. Sous Debian, ce répertoire se présente de la manière suivante :

```
root@interventions:/etc/apache2# ls --color
apache2.conf  envvars     magic           mods-enabled  sites-available
conf.d        httpd.conf  mods-available  ports.conf    sites-enabled
```

En ce qui concerne les directives principales, définissant les fonctionnalités de base du serveur HTTPD, on trouve notamment les directives suivantes qui seront réparties dans les différents répertoires/fichiers :

- `Listen` : précise le port sur lequel le serveur va écouter
- `ServerName` : adresse IP ou nom DNS du serveur
- `DocumentRoot` : répertoire des fichiers proposés par le serveur
- `AllowOverride` : permet de modifier les droits pour un dossier en particulier
- `DirectoryIndex` : nom du fichier par défaut
- `Alias` : définit les alias utilisables dans l'URL
- `ScriptAlias` : définit les alias utilisables pour les scripts CGI

Réglementer l'accès à un site : *.htaccess*

Dans la plupart des cas aujourd'hui, si un site demande l'authetification d'un utilisateur, ceci sera géré par le site (en https). Cependant, dans des certains cas, nous pouvons vouloir gérer l'authentification par le serveur. Pouvez-vous vous imaginer pourquoi, par exemple ?
L'utilisation de fichier .***htaccess*** permet de limiter très simplement l'accès à tout ou partie d'un site web.

Authentification pour l'accès à un répertoire

La mise en place de l'authentification dans ces cas se fait en 2 étapes. D'un côté le fichier .***htaccess*** qui définit les modalités d'accès, et de l'autre le fichier .***htpasswd*** qui contient les utilisateurs autorisés.

Exemple de fichier .***htaccess***

```
AuthUserFile /etc/httpd/conf/.htpasswd
AuthName Mapage
AuthType basic
<Limit GET>
require valid-user
</Limit>
```

Remarques :

- le fichier caché *.htpasswd*, contenant des mots de passe cachés par une fonction de hachage ou un algorithme de chiffrement assez simpliste, peut porter un nom quelconque et doit se trouver hors de l'arborescence du site Web pour des raisons de sécurité (les méthodes de chiffrement/fonctions de hachage ne sont pas vraiment sécurisées).
- S'il est nécessaire de protéger plusieurs répertoires, il est recommandé de placer tous les fichiers des mots de passe dans le même répertoire.
- La protection d'un répertoire est propagée à tous les sous-répertoires de celui-ci.

Le fichier des mots de passe

Le fichier de mots de passe chiffrés pour chacun des utilisateurs autorisés est créé avec les droits de root. La commande pour créer le fichier *.htpasswd* est `htpasswd` : .

- Si le fichier n'existe pas encore, il faut préciser qu'on désire le créer avec l'option `-c` :

```
htpasswd -c /etc/apache2/.htpasswd Nom_Utilisateur
```

- Pour ajouter un utilisateur au fichier existant, **surtout ne pas utiliser** l'option –c :

```
htpasswd -c /etc/apache2/.htpasswd Nom_Utilisateur
```

Filtrage par adresse IP d'origine

Au lieu de filtrer les usagers en leur demandant un nom d'utilisateur et un mot de passe, le filtrage par adresse IP d'origine (et/ou nom de domaine d'origine) peut se faire de manière encore plus simple en plaçant simplement le ***.htaccess*** approprié dans le répertoire à protéger.

Exemple de fichier ***.htaccess***

```
order allow,deny
allow from all
deny from domaine.tld
deny from 172.16.35.45
```

Le principe des vhosts

Le terme de « Virtual Host » (serveurs virtuels) fait référence à un principe qui permet d'héberger plusieurs sites web (www.domaine1.org, www.domaine2.org ...) sur une même machine. Ceci se fait souvent lorsqu'on demande à un tiers (un fournisseur de services) d'héberger notre site Web. Le principe des vhosts peut s'appuyer sur la technique des alias d'adresses ip « **ip-based** », ou sur la technique de multiples noms adressables sur la même adresse : « **name-based** ».

Technique "IP-based"

La technique IP-based est la plus simple à mettre en œuvre, mais elle est beaucoup plus contraignante au niveau des adresses, puisqu'elle consiste simplement à utiliser une adresse IP différente pour chacun des sites hébergés. Le gain se situe donc au niveau du nombre de machines physique et au niveau du nombre de processus nécessaires.

Technique "named-based"

Il s'agit de faire héberger plusieurs sites sur une même adresse. Le serveur associe le nom d'hôte aux entêtes des requêtes *http* passées par le client. En utilisant cette technique plusieurs « serveurs web» peuvent se partager la même adresse IP. C'est donc celle qui est privilégiée.

Les 2 méthodes nécessitent bien entendu de déclarer correctement les multiples noms de machines au niveau du serveur DNS, et de les associer aux bonnes adresses IP (adresse unique dans le cas de la technique « name-based »).

Mise en oeuvre

Pour la mise en place de cette technique, il suffit de configurer le serveur Apache afin qu'il reconnaisse les différents noms de site, et de déclarer la correspondance entre ces noms et les répertoires contenant les pages des sites (DocumentRoot).

Extrait de *ports.conf*

```
NameVirtualHost *:80
```

Extrait de *001-domain*

```
<VirtualHost *:80>
        ServerName www.domain.tld
        DocumentRoot /www/domain
</VirtualHost>
```

Extrait de *002-otherdomain*

```
<VirtualHost *:80>
        ServerName www.otherdomain.tld
        DocumentRoot /www/otherdomain
</VirtualHost>
```

Les scripts CGI

CGI (Common Gateway Interface) est une méthode standard d'extension des fonctionnalités d'un serveur Web par l'exécution de scripts sur ce serveur, en réponse aux requêtes d'un navigateur Web. Un script CGI peut être un script Shell ou autres (PERL, PhP…), ou bien un programme exécutable classique.

Un script CGI va être lancé par le daemon HTTPD en réponse à une demande provenant du client, soit directement, soit par l'intermédiaire d'une page *html*. Dans les deux cas, le script doit avoir été déclaré comme tel via la directive ScriptAlias dans la configuration d'apache.

Quand un script CGI (c'est-à-dire un exécutable) est démarré par le serveur HTTP, il se situe dans un contexte particulier :

- il a reçu du serveur HTTP un certain nombre de variables d'environnement :
  - l'@IP du client,
  - une méthode de passage de paramètres,
  - les valeurs envoyées par le client,
  - ...
- sa sortie standard est reliée au processus serveur HTTP qui se chargera de transmettre les résultats au navigateur,
- le script CGI s'exécute avec comme UserName Apache,

Voici un exemple de script CGI simple, écrit en shell :

```
#!/bin/bash
echo "Content-type:text/plain"
echo ""
echo -n "Nous sommes le : "
date
echo "Votre adresse IP est $REMOTE_ADDR"
echo "Vous utilisez le navigateur $HTTP_USER_AGENT"
```

Le script CGI ci-dessus doit constituer un document résultat. Il suffit de faire des affichages sur sa sortie standard (commande echo) car elle est indirectement reliée au navigateur. Comme premier affichage, le script doit indiquer la nature du document résultat, en précisant l'entête MIME :

- Content-type: text/plain si le document résultat est un document texte,
- Content-type: text/html si le document résultat est un document html.

Dans ce dernier cas, le script devra constituer un document html valide en mettant des balises, c'est-à-dire en faisant afficher les balises HTML.

Ce script a vocation à être lancé directement par le navigateur, il va simplement utiliser la fonction système date ainsi que des **variables d'environnement** pour créer un affichage simple mais personnalisé pour chaque client web.

L'utilisation d'un script avec un formulaire

Il est plus intéressant encore d'utiliser un script en l'appelant depuis une page HTML de manière à pouvoir lui passer plus de paramètres via une requete de type ***GET*** ou ***POST :***

- méthode **GET** : les paramètres sont dans la variable QUERY_STRING,
- méthode **POST** : les paramètres sont sur l'entrée standard (stdin).

Dans les deux cas, les paramètres sont passés sous la forme d'une *string* de la forme :

```
param1=value1&param2=value2&param3=value3
```

Le protocole FTP

FTP est un protocole utilisé à la couche application. Il sert à mettre en place le transfert de fichiers entre le répertoire courant du client et le répertoire courant du serveur (le transfert est bidirectionnel).

FTP est un protocole client-serveur qui est encapsulé à la couche 4 par TCP. Le serveur utilise le port 21 pour FTP et 22 pour SFTP.

Dans un premier temps en FTP, le client s'authentifie avec un login et un mot de passe. En revanche le transfert de données n'est pas sécurisé. Si on veut avoir la confidentialité du transfert, nous pouvons utiliser le protocole SSH d'abord, puis de continuer en FTP (ce qui donne sftp)

Utiliser FTP

Sous Linux (et Windows) nous pouvons utiliser la commande `ftp` pour accéder en ligne de commande à un client FTP :

```
ftp <adresse IP du serveur cherché>
```

Vous pouvez utiliser la command man `ftp` pour mieux connaître les commandes possibles en ftp. Parmi les commandes le plus utilisées sont :

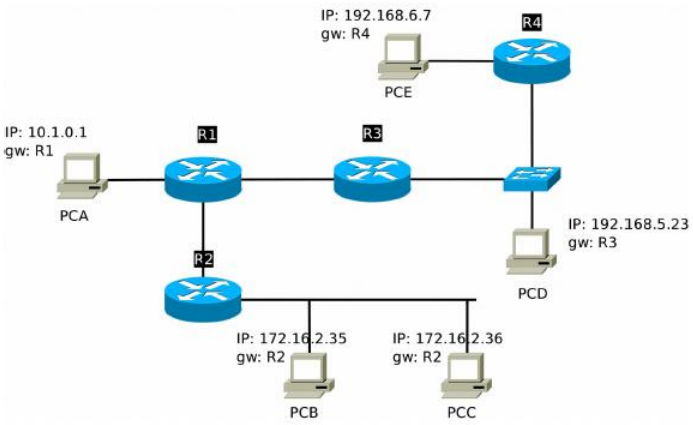
- `open` : ouvrir une connexion à une machine donnée
- `close` : fermeture de la session FTP et retour en commande cmd
- `bye` : fin de la session et exit de FTP
- `ls` : liste le contenu du répertoire actuel du côté du serveur
- `dir` : même chose que ls, seulement côté client
- `pwd` : vient de print working directory : affiche le répertoire actuel côté serveur
- `cd` : changement du répertoire actuel du côté du serveur
- `put` : effectue en transfert du client (répertoire actuel) vers le serveur (répertoire actuel)
- `get` : effectue le transfert inverse du serveur au client
- `! <commande>` : fait exécuter une commande définie pour le serveur pour le client

A chaque fois qu'on tape une commande, cela est représenté par un mot (par exemple USER ou XPWD) ; cette commande sera une question adressée au serveur ou au client FTP. La question aura une réponse (et en fonction du résultat il y aura aussi un résultat). La réponse est associée à un code numérique, comme par exemple 220, 331, etc.

Même si on ne connaît pas les codes, une capture des échanges pourraient nous donner une idée de quel code correspond à quelle requête/réponse.

Exercice 1

Cet exercice concerne le réseau dont la topologie est dans la figure ci-dessous :



Dans cette figure on indique les routeurs par un nom, de R1 à R4. Les machines sont indiquées par nom et par adresse IP. Pour chaque machine on indique sa passerelle par défaut, précédée par l'annotation gw.

1. Indiquez les réseaux présents dans cette figure, avec les machines qui en font partie.
2. La commande suivante est tapée sur une des machines ci-dessus.

   ```
   ss -ant
   ```

   Elle retourne le résultat suivant :

   ```
       Local Address       Foreign Address         state
   tcp 10.1.0.1:2568       172.16.2.36:23          ESTABLISHED
   ```

   Répondez aux questions suivantes :

   - Quel est le rôle de la commande `ss` ?
   - Pouvez-vous indiquer la machine sur laquelle on a tapé cette commande ?
   - Soulignez les ports utilisés du côté serveur et du côté client.
   - Qui est le client et qui est le serveur dans cet échange ? Justifiez votre réponse.
   - Quel protocole est utilisé à la couche application ?
   - Pourquoi a-t-on le mot « `tcp` » à gauche de l'écran de résultats ?
   - Quel est le sens du mot « `ESTABLISHED` » à gauche de l'écran ?

3. Vous pouvez supposer que chaque machine a été configurée avec la passerelle par défaut mentionnée dans la figure. Est-ce que cela suffit pour permettre la connexion affichée par la commande `ss` ? Sinon, quelles routes peut-on encore déduire ?