R2.04-- Les bases des réseaux

TD1 : Les adresses IP et le protocole ARP

Adresse IP & adresse réseau

Dans un réseau isolé on identifie une machine par son adresse machine (MAC). Mais, lorsque deux réseaux différents veulent communiquer, ceci est réalisable grâce aux adresses IP. A ce but, chaque machine de chaque réseau est associée à une adresse MAC (non-modifiable) et une adresse IP (qui peut changer). Les adresses MAC ne seront connues (et utilisables) qu'au sein du réseau de la machine. De plus, chaque réseau (domaine de collision) est associé à une adresse IP. Dans ce module nous allons principalement utiliser les adresses IPv4.

Une adresse IPv4 a quatre octets, ou 32 bits. On utilise un point « . » pour séparer les octets de l'adresse IP.

Adresse IP d'un réseau : notation CIDR

L'adresse IP d'une machine indique implicitement l'adresse IP de son réseau.

L'adresse IP d'un réseau doit indiquer : le nombre maximal de machines incluses dans ce réseau, l'adresse IP de la première et dernière machine qui peut être incluse dans ce réseau et une adresse de broadcast. C'est très important, lorsqu'on indique une adresse réseau de préciser tous ces éléments. Une même adresse réseaux peut correspondre à une plage à 128 machines ou une à 2 machines.

On indique la taille d'un réseau de deux façons : en utilisant une notation CIDR ou en utilisant une masque de réseau (netmask).

La notation CIDR nous indique combien de bits, sur le total de 32 bits de l'adresse, sont inclus dans l'adresse réseau. Les autres bits, jusqu'à un total de 32, sont dédiés aux machines dans l'adresse. La première valeur possible sur 32 bits est : 0.0.0.0. La dernière est 255.255.255.255.

- Exemple : 192.168.13.0/24
  Prenons l'exemple où une université utilise une plage 192.168.13.0/24. Le nombre 24 dénote combien de bits sont inclus dans la partie réseau. Ceci veut dire que 32-24 bits sont dédiés à la partie machine -- ce qui correspond au dernier octet de l'adresse. Notamment, la première adresse dans ce réseau sera : 192.168.13.1 et la dernière adresse sera 192.168.13.255. La dernière

adresse dans chaque sous-réseau est automatiquement l'adresse de broadcast. Dans ce cas-ci, l'adresse de broadcast est 192.168.13. 255. La première adresse possible pour une machine est 192.168.13.1 et la dernière est 192.168.13.254. En général on calcule le nombre de machines qu'on peut avoir dans un réseau dont l'adresse est /x en notation CIDR selon la formule suivante : $n_{machines} = 2^{(32-x)} - 2$. Les deux adresses qui ne sont pas incluses sont l'adresse du réseau lui-même et l'adresse de broadcast. Sur cet exemple on a $2^8 - 2 = 256 - 2 = 254$ machines.

- Exemple: 192.168.136.0/23.
  Le nombre total de machines est : $2^{32-23} - 2 = 2^9 - 2 = 512 - 2 = 510$. Pour comprendre quelle est la première adresse dans ce réseau, il faut regarder la notation CIDR. Si le réseau est sur 23 bits (comme indiqué par la notation CIDR), le 23 premiers bits de l'adresse réseau sont fixés. 23 bits veut dire les 2 premiers octets et les 7 premiers bits (sur 8) du troisième. On écrit le troisième bit en binaire : 136 = 10001000. On fixe les 7 premiers bits : 1000100**0**.Cette partie ne changera pas : elle représente le préfixe obligatoire de toute adresse comprise dans notre réseau. En revanche, deux machines différentes dans ce domaine de collision auront le reste de l'adresse (notamment le dernier octet + le dernier bit du troisième octet) différents. Ceci nous donne comme première adresse : 192.168.136.1 et la dernière adresse (adresse de broadcast) 192.168.137.255. La dernière adresse qui peut être donnée à une machine sera donc 192.168/137.254.

L'adresse d'un réseau doit, impérativement, n'avoir que de 0s pour les bits qui sont non-fixés. Par exemple une adresse réseau /25 bits aura les derniers 32-25 = 7 bits égaux à 0 : 192.168.3.128/25 est une adresse réseau valide, tandis que 192.168.3.32/28 ne l'est pas.

L'inverse n'est pas vrai. Un réseau peut avoir de 0s également parmi les bits fixés. Par exemple 192.0.3.0/24 est une adresse réseau valide.

Adresse IP avec un masque de réseau (netmask)

Une façon alternative d'écrire un numéro de réseau est de donner le numéro et un masque de réseau. Le masque de réseau indique le nombre de bits qui sont fixés comme partie du réseau (les autres bits peuvent changer avec l'adresse IP de chaque machine). Un masque est un numéro sur 4 octets, dont les quelques premiers bits sont tous égaux à 1 et les derniers sont tous 0. Le nombre total de bits égaux à 1 est le numéro de bits fixés.

- 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. On a un total de $2^8 - 2 = 254$ machines. La première adresse possible est : 192.168.33.1. La dernière adresse qu'on peut donner à une machine est : 192.168.33.254. L'adresse de broadcast est : 192.168.33.255
- 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, ou une adresse en notation CIDR de 192.36.2.0/25. Ce réseau à la place pour $2^7 - 2$ = 126

machines. La première adresse est 192.36.2.1 et la dernière adresse qu'on peut donner à une machine est : 192.36.2.126. L'adresse de broadcast est au 192.36.2.127.

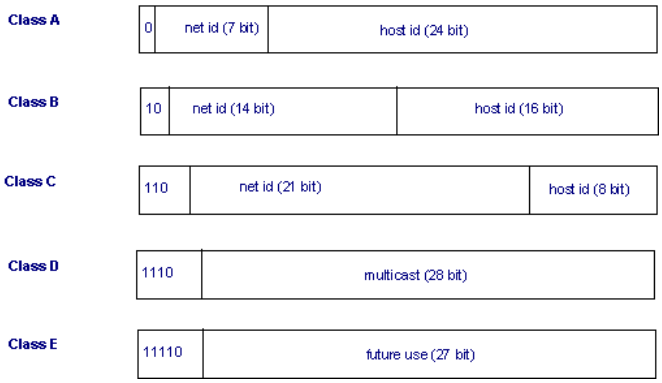
Adresse IP vs adresse réseau

Si on parle de l'adresse IP d'une machine, celle-ci aura toujours le format x.y.z.t, où x,y,z,t prennent des valeurs entre 0 et 255. Cependant, lorsqu'on veut indiquer une adresse réseau il est impératif d'indiquer la taille du réseau, soit en notation CIDR, soit avec un masque.

- Exemple : 192.168.32.128 est une adresse IP
- Exemple : 192.168.32.128/25 est une adresse réseau

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.

Parmi les adresses IP, il y a quelques plages d'adresses qui ne peuvent pas être utilisées pour une machine qui est directement connectée 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.

Si, sur une machine, on configure une adresse IP sans lui spécifier un masque de réseau en utilisant la commande ip address add, la machine sera configurée par défaut en /32. Avant, l'utilisation de la commande ifconfig mettait la machine par défaut dans un réseau correspondant à sa classe (/8 pour une adresse de classe A, /16 pour une adresse de classe B, etc.).

Le protocole ARP

Le protocole ARP est utilisé lorsqu'on essaie d'associer une adresse MAC à une adresse IP. Dans un certain sens on peut penser à l'adresse IP comme à un passeport (un document utilisé à l'international) et à l'adresse MAC comme à un numéro de sécurité sociale (utilisé seulement en France). Chaque machine garde une table ARP, 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, on cherche la nouvelle correspondance par une requête ARP.

Le protocole ARP vise les adresses MAC dans un seul sous-réseau. Les messages de ce protocole sont encapsulés sous la forme d'un trame Ethernet, et donc elles sont seulement utilisables à la couche II et sur un medium physique partagé. Ce protocole consiste en deux messages :

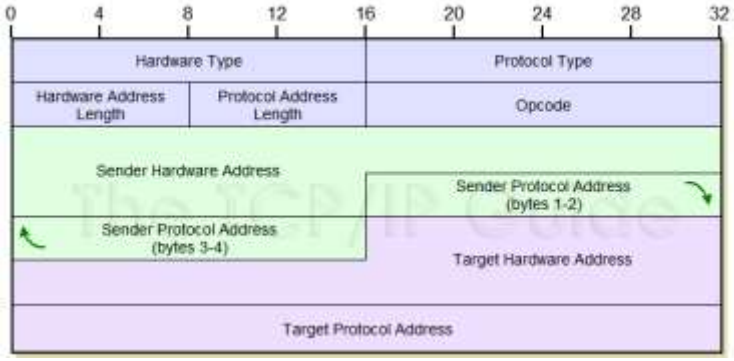
- Une requête ARP envoyée en broadcast (avec le message « Qui a l'adresse <adresse IP> ? Dites à <adresse de l'expéditeur> »)
- Une réponse envoyée en unicast (contenant l'adresse demandée) à l'expéditeur

Le protocole ARP est toujours utilisé pour les envois intra-réseau. Supposons qu'une machine A ait besoin de l'adresse MAC de sa passerelle par défaut, une machine PASS dont A connaît l'adresse IP. Pour pouvoir envoyer son message, A doit apprendre l'adresse MAC de la machine PASS -- et pour cela, elle utilise le protocole ARP. La machine A enverra donc une requête ARP, et une autre machine, soit PASS soit une autre machine, répondra dans une réponse ARP avec l'adresse MAC recherchée. Attention, cela ne marchera que si PASS fait partie du réseau de A.

Nous distinguons trois types d'entités impliquées dans chaque message ARP (requête/réponse) : l'envoyeur, le receveur et la cible. Pour une requête ARP, l'envoyeur est une machine cherchant l'adresse MAC d'une autre machine dans son réseau -- dans notre exemple, l'envoyeur est A. Le receveur est tout le monde dans le réseau local (l'adresse MAC de broadcast est utilisée en tant que destinataire). La cible est la machine pour laquelle on cherche l'adresse MAC -- dans notre exemple la machine PASS.

Pour la réponse, l'envoyeur peut être n'importe quelle machine dans le réseau, qui connait l'adresse MAC de la machine cible. Le destinataire de la réponse ARP est la machine ayant envoyé la requête ARP. La cible est toujours la machine pour laquelle on a tenté de connaître l'adresse MAC.

Les messages ARP ont le format suivant :



http://www.tcpipguide.com

Dans la requête ARP, le champ « Target hardware address » -- qui représente l'adresse MAC qu'on cherche, est mis à la valeur «inconnue », c'est-à-dire : 00 :00 :00 :00 :00 : 00. La valeur Opcode se réfère au code de l'opération cherchée, notamment 1 pour la requête et 2 pour la réponse. Ce code peut aussi prendre d'autres valeurs, actuellement entre 0 et 25 : https://www.iana.org/assignments/arp-parameters/arp-parameters.xhtml . C'est pourquoi cette valeur est représentée sur 5 bits.

Déchiffrer une trame ARP -- Ethernet

Les messages ARP sont encapsulés dans des trames Ethernet. L'encapsulation extérieure appartient au protocole Ethernet, tandis que le formatage ARP est à l'intérieur de la trame.

Voici pour rappel l'en-tête Ethernet.

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.

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

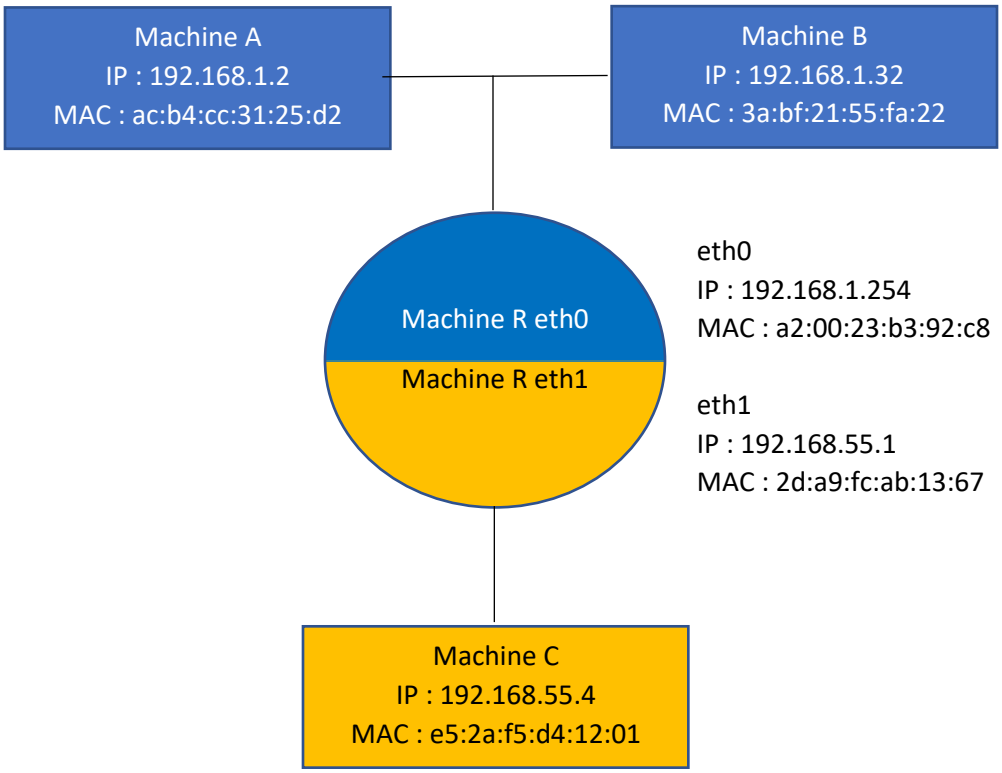
TD2 : Le routage

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 II.

Pour les transmissions intra-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 MAC et une adresse IP dans le sous-réseau respectif. Notamment :



Lorsqu'une machine dans un certain sous-réseau A veut envoyer un message à une machine dans un sous-réseau B, auquel il est lié par une passerelle R, alors 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 notre transmission inter-réseau, un premier envoi s'effectue donc entre l'expéditeur (dans le réseau A) et la passerelle en utilisant :

- L'adresse IP de l'expéditeur pour l'adresse IP Source
- L'adresse IP du destinataire final pour l'adresse IP Destinataire
- L'adresse MAC de l'expéditeur pour l'adresse MAC source
- L'adresse MAC de la passerelle (sur l'interface liée au réseau A) pour le destinataire Ethernet

Notamment cela indique à la passerelle qu'elle est censée faire suivre ce message à la machine destinataire dont l'adresse IP est mise en tant que IP destinataire.

Donc dans un deuxième temps (après une éventuelle requête ARP), la passerelle envoie un deuxième message avec les adresses suivantes :

- L'adresse IP de l'expéditeur original et l'adresse IP du destinataire original
- L'adresse MAC de destination sera l'adresse MAC du destinataire
- L'adresse MAC de la passerelle (sur l'interface liée au réseau B) pour l'expéditeur Ethernet

Exemple : prenons l'exemple du réseau ci-dessus dans lequel la machine A veut envoyer un message à la machine C. Dans ce cas-ci, l'envoi sera partagé en deux étapes :

1. A → passerelle : un message avec l'adresse MAC de l'expéditeur mise à ac:b4:cc:31:25:d2, l'adresse MAC du destinataire sera : a2:00:23:b3:92:c8, l'adresse IP de l'expéditeur sera : 192.168.1.2 et l'adresse IP du destinataire sera : 192.168.55.4
2. passerelle → C : un message avec l'adresse MAC de l'expéditeur mise à 2d:a9:fc:ab:13:67, l'adresse MAC du destinataire sera : e5:2a:f5:d4:12:01, l'adresse IP de l'expéditeur sera : 192.168.1.2 et l'adresse IP du destinataire sera : 192.168.55.4

Dans un envoi inter-réseau, la transmission des messages entre les machines A et C ci-dessus marche dans la séquence suivante :

- Première étape : la machine A cherche l'adresse MAC de la passerelle. Ceci se fait en utilisant le protocole ARP.
- Deuxième étape : le premier message ci-dessus est envoyé
- Troisième étape : la passerelle utilise le protocole ARP pour savoir l'adresse MAC de la machine C
- Etape finale : C reçoit le message

Il faut savoir que l'envoi entre la machine A et la passerelle nécessite une configuration de routage ; par contre une fois le message arrivé à une passerelle pour laquelle le IP forwarding a été activé, la transmission entre cette dernière et C se fait automatiquement (sans que la machine C ait un routage configuré).

IP Forwarding, routage

Comme on a vu en CM, le rôle d'une passerelle est de connecter deux réseaux distincts. Pour assurer la connectivité, une telle passerelle doit pouvoir jouer le rôle d'un courrier, en faisant passer des messages d'un réseau à l'autre.

On se rappelle que normalement, à la couche physique, tout bit envoyé est vu par toutes les machines qui sont connectés sur le même dispositif physique. Mais, selon les adresses MAC et IP de l'expéditeur/destinataire, une machine décidera très vite si elle doit ou non traiter un certain message reçu. Notamment, elle traitera (un sous-ensemble) des messages qui lui sont destinés (destinataire IP).

Dans le cas d'une passerelle, nous voulons qu'elle traite (c'est-à-dire qu'elle reprenne et fait passer) des messages qui ne lui sont pas destinés, et qu'elle doit faire passer plus loin. La passerelle devra aussi modifier certains champs des en-têtes du message. Pour ce faire, on doit lui activer l'IP-forwarding, ce qui se fait en IPv4 avec la commande :

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

Une passerelle pour laquelle on n'a pas activé l'IP forwarding ne pourra pas fonctionner en tant que passerelle.

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 un ou des passerelles qui seront responsables à faire passer des messages hors du réseau de la machine. Attention : on n'aura jamais besoin d'une passerelle lorsqu'on envoie un message dans notre propre réseau !

Les machines personnelles ont souvent une **passerelle par défaut**. On configure une passerelle par défaut en utilisant la commande :

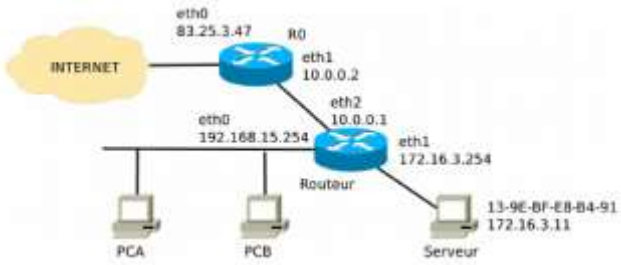
```
ip route add default via <adresse passerelle>
```

La passerelle doit toujours être dans un même domaine de collision que la machine sur laquelle elle est configurée en tant que passerelle !

Une façon alternative de configurer une interface est de la spécifier dans le fichier /etc/network/interfaces (voir la section suivante). Si une telle passerelle est configurée, alors tout message partant de la machine vers une destination hors de son réseau arrivera par défaut à la passerelle, qui l'enverra plus loin. Dans le tableau de routage, accessible via la commande `ip route show` on rencontrera deux lignes : la première spécifiera que, dans son propre réseau, la machine n'aura pas besoin d'une passerelle. La deuxième indiquera la passerelle par défaut pour tout message partant pour 0.0.0.0/0 (c'est-à-dire pour toute destination possible). Comme un tableau de routage est parcouru des destinations les plus spécifiques au moins spécifiques (par exemple les destinations /26 avant les /8 ou les /0), la première de ces deux lignes sera prise en compte en premier.

Si on veut qu'une machine soit connectée à toute destination possible, une passerelle doit correspondre à toute adresse IP de destination possible. Si, en parcourant son tableau de routage, la machine ne trouve aucune passerelle pour la destination envisagée, alors l'envoi ne peut pas s'effectuer.

Attention : un tableau de routage n'a pas comme rôle d'établir une carte complète de l'Internet sur une machine : il suffit d'indiquer, à chaque fois, quelle machine doit prendre en charge et envoyer le message plus loin vers sa destination. Par exemple, dans la figure ci-dessus, supposons que la machine Serveur veuille envoyer un message vers 8.8.8.8. Le message devra passer via la machine Routeur, ensuite la machine R0, et ensuite d'autres passerelles, jusqu'à arriver à 8.8.8.8. Mais, sur la machine Serveur il suffit d'indiquer que la machine Routeur prend tout message envoyé par Serveur hors de son réseau. Ensuite, sur la machine Routeur, il suffit d'indiquer que le prochain relais est la machine R0.



Il n'est pas obligatoire de toujours utiliser une passerelle par défaut. En effet, prenons la figure ci-dessus. Prenons le cas de la machine Serveur. Nous voulons que cette dernière soit liée à l'Internet. Pour ce faire, le serveur aura la machine appelée Routeur en tant que passerelle par défaut. Lorsque la machine Serveur envoie par exemple un ping vers 8.8.8.8, alors ce message sera envoyé en utilisant la passerelle Routeur.

Mais, une fois le message reçu par Routeur, que doit-il faire pour assurer la transmission vers l'Internet ?

On regarde la topologie donnée. On voit que le réseau de la machine Serveur (à droite) et le réseau des machines PCA, PCB (à gauche) sont des « culs de sac » -- c'est-à-dire, la seule sortie de ce réseau est donnée par la machine Routeur. Donc, pour la configuration de la machine Routeur, il semble logique que sa passerelle par défaut soit R0. Maintenant, si la machine Serveur envoie un message à 8.8.8.8, le message passe par Routeur, et Routeur l'envoie plus loin, à R0.

Et pour R0 ?

C'est clair que, pour pouvoir envoyer les messages plus loin, R0 devra avoir une passerelle par défaut dans son réseau à gauche, vers le nuage « Internet ». Mais, que se passe-t-il si R0 reçoit un message pour la machine serveur ? Ce message devrait logiquement passer, via Routeur, vers le Serveur. Il faut donc ajouter une route spécifique sur R0, notamment en utilisant la commande :

```
ip route add <reseau destination finale>/<netmask ou CIDR> via <adresse routeur>
```

Dans le cas de R0, il faut rajouter la ligne :

```
ip route add 172.16.0.0/16 via 10.0.0.1
```

Nous pouvons faire afficher les routes mises en place sur une machine par la commande `ip route show`. Celle-ci nous affiche un tableau similaire à celui indiqué ci-dessous.

```
Itinéraires actifs :
Destination  réseau     Masque réseau          Adr. passerelle
164.81.20.0            255.255.255.0          0.0.0.0
192.168.12.0           255.255.255.0          0.0.0.0
10.25.0.0              255.255.0.0            0.0.0.0
127.0.0.0              255.0.0.0              0.0.0.0
192.168.11.0           255.255.255.0          192.168.12.254
172.29.0.0             255.255.0.0            192.168.12.254
10.23.34.21            255.255.255.255        10.25.0.254
0.0.0.0                0.0.0.0                164.81.20.254
```

Contrôle de connexion : les ping

Une fois le routage mis en place sur un réseau, nous pouvons utiliser un message de type `ping` pour vérifier l'état de la connexion. Disons qu'on a le cas de la figure ci-dessus (voir la première page) et qu'on veut vérifier que, après d'avoir tapé de diverses commandes sur les machines dans la figure, les machines PCA et PCC peuvent communiquer.

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.

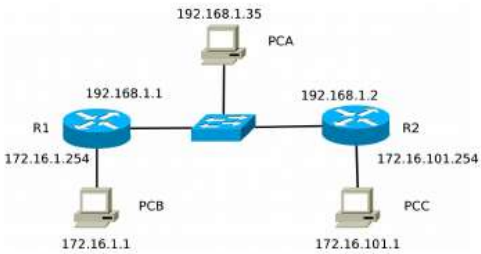
Il y a donc plusieurs raisons pour lesquelles un ping ne fonctionne pas entre les deux machines :

- 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.

Exercice 1

Nous allons partir sur la configuration suivante de réseau :



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 ?

3. Sur la machine PCB nous voulons mettre en place une route par défaut sur R1
    a. Donnez la commande pour faire cela
    b. Peut-on faire la même chose en utilisant la commande ip `route add <reseau>/<masque ou CIDR> via <adresse IP>` (et comment) ?
    c. Quelles commandes faut-il taper sur la machine R1 pour réaliser le bon acheminement de messages vers le sous-réseau de PCA et PCC respectivement ?
    d. Est-ce que ces commandes suffisent pour assurer qu'un ping fait de PCA vers PCC marche ? Justifiez votre réponse

R2.05-- Les services en réseaux

TD1 : Le fichier interfaces et l'adressage dynamique

Le fichier /etc/network/interfaces

Jusqu'au 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`. Ce fichier nous permet une 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 indique la présence 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 l'adresse en notation CIDR ?). 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
```

L’adressage IP dynamiquement

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 serveur DHCP est une machine configurée dans un réseau hôte. La configuration de ce serveur spécifie une plage d’adresses IP qui peuvent être utilisées pour les machines visiteuse, ainsi que d’autres spécifications comme par exemple la durée maximale de location pour une adresse. 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.

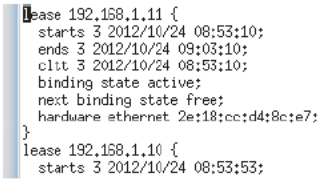
Attribution fixe et dynamique

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*.



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.

Le protocole DHCP fonctionne en général sur UDP, en utilisant les ports 67 pour le serveur et 68 pour le client.

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, on utilise les commandes suivantes pour manipuler le serveur :

- Démarrage : `systemctl start isc-dhcp-server`
- Arrêt : `systemctl stop isc-dhcp-server stop`
- Redémarrage : `systemctl restart isc-dhcp-server`

La commande `dhcpd -t` permet de vérifier la syntaxe du fichier de configuration sans démarrer le serveur.

L'installation côté client

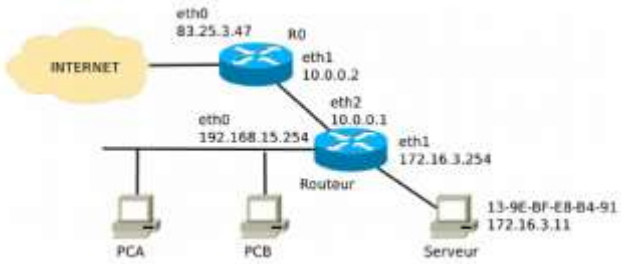
Pour s'assurer qu'un client puisse demander une adresse IP, il faut premièrement mettre en place une configuration IP lui permettant de demander des adresses IP dynamiquement. Ceci se fait dans le fichier etc/network/interfaces -- voire le TD.

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>`

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 ?

TD2 : Les protocoles client-serveur, FTP, HTTP

L'utilisation de protocoles en mode client-serveur

Beaucoup de protocoles utilisés dans les réseaux informatiques fonctionnent selon un modèle client-serveur. Sur le serveur un processus qu'on appelle un daemon reste à l'écoute des communications. Lorsqu'une requête arrive d'un processus client, alors une connexion TCP ou UDP (en fonction du protocole utilisé) est mise en place. Les échanges qui suivent sont encapsulés dans ce protocole, qui peut gérer également la fiabilité des communications.

Pour différentier entre les divers types de service qu'un serveur peut mettre à la disponibilité des clients, nous avons la notion de port. Du côté du serveur nous allons toujours utiliser un port dédié. Par exemple :

- HTTP : port 80/TCP
- FTP : port 21/TCP
- SSH + transfert sécurisé : port 22/TCP
- Telnet : port 23/TCP
- HTTPS : port 443/TCP

Un client qui cherche à se connecter à un serveur utilisera le premier port disponible qui n'est pas assigné, donc un port > 1024.

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`. 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`, également présentée lors des CM, 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 `netstat -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.

Les connexions de chaque machine

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.

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 nom

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.

Le protocole HTTP fonctionne dans une architecture **client-serveur**. Le client HTTP est le navigateur qu'on utilise. Le processus serveur se trouve sur un serveur Apache ou IIS (Internet Information Services).

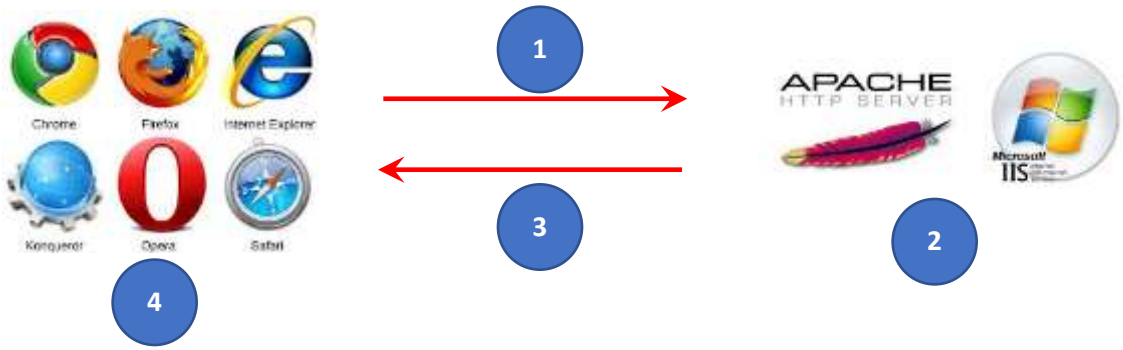


Figure 1 : le fonctionnement de HTTP

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 privés, on peut utiliser le protocole 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

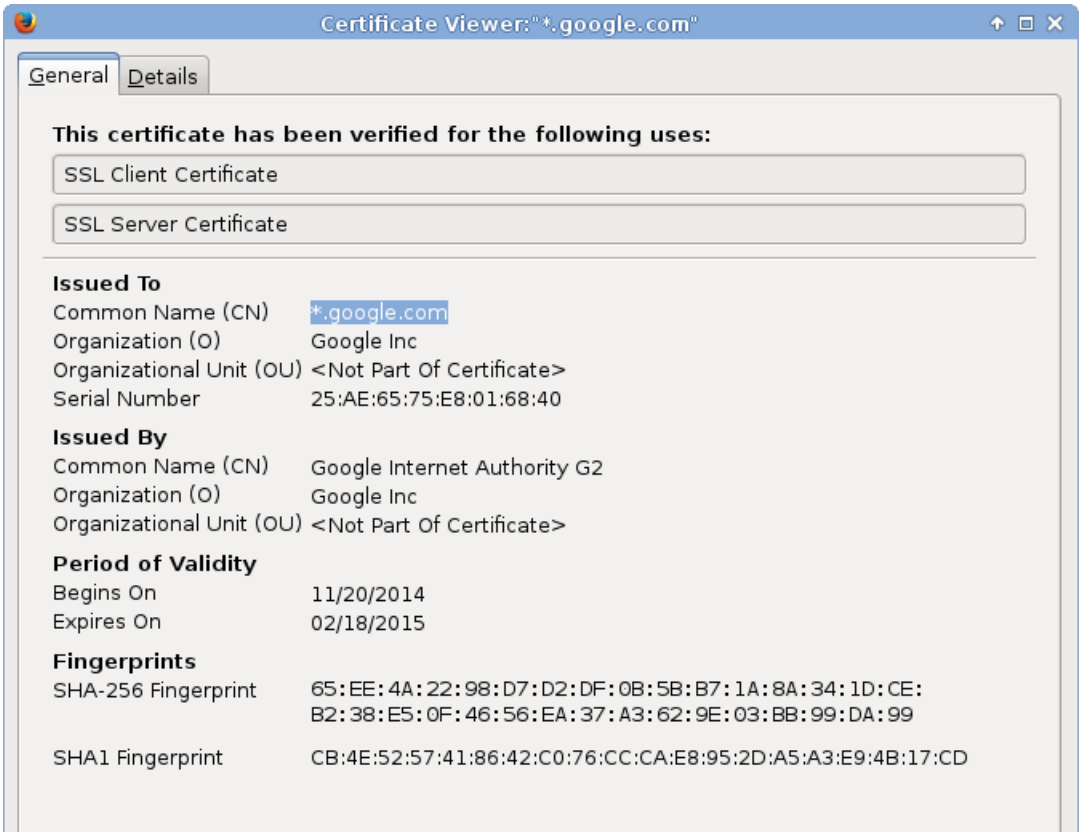
Le protocole TLS est un exemple de protocole d'échange de clé authentifié. D'autres exemples très utilisés aujourd'hui sont SSH (utilisé pour sécuriser l'accès à distance à une machine) et IPSec (utilisé principalement pour sécuriser la connexion par VPN).

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 au cours du module OS01. 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, le client aura besoin d'un certificat, établi par une autorité qui garantit l'association entre la clé publique et un nom de domaine.



Le navigateur vérifiera les certificats à chaque fois qu'on initialise une connexion sur TLS. 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 init.d :

```
$ /etc/init.d/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). Le port utilisé pour le protocole FTP est TCP/21. Dans un premier temps, 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.

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 :

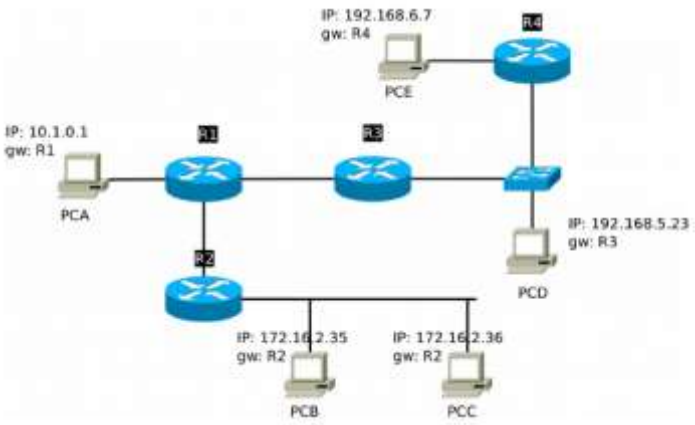
- `open` : ouvrir une connexion à une machine donnée
- `close` : fermeture de la session FTP et retour en commande cmd
- `bye` : fin de la session et exit de FTP
- `ls` : liste le contenu du répertoire actuel du côté du serveur
- `dir` : même chose que ls, seulement côté client
- `pwd` : vient de print 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 ?

TD3 : Le protocole DNS

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 que notre machine associe mail.google.com à une machine destinataire, il nous faut un mécanisme qui trouve l'association entre mail.google.com 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. Si on veut trouver un nom à partir d'une adresse IP, il s'agit d'une résolution inverse. On a deux possibilités : soit on peut trouver l'adresse qu'on cherche localement (sur un fichier *`hosts`*), soit on ne le trouve pas, et alors on la cherche en utilisant le protocole DNS pour demander la réponse à un serveur DNS. Dans la deuxième cas notre machine jouera le rôle d'un client DNS, qui s'appelle un `resolver`.

Méthode 1 : Le fichier /etc/hosts

Le fichier */etc/hosts* est une base de données où les associations entre les adresses IP et les noms de machines sont stockées. Le fichier */etc/hosts* existe en Windows de même qu'en Linux.

Même si à l'époque la résolution de nom était faite seulement en utilisant les fichiers */etc/hosts*, cette méthode présente des inconvénients très importants aujourd'hui. Un premier inconvénient est le fait que chaque machine ne peut utiliser que son propre fichier */etc/hosts* : ceci veut dire qu'on doit stocker chaque nouvelle association nom ↔ adresse IP dans chaque machine d'un réseau. C'est pourquoi on dit que l'utilisation des fichiers */etc/hosts* est une méthode locale de résolution de nom. Pour les réseaux qui ont beaucoup de machines, ceci peut être encombrant.

Un autre inconvénient est le fait que, si une machine change de nom ou d'adresse IP, les modifications devront être faites sur chaque machine individuellement. Ceci est l'aspect statique de cette méthode de résolution de nom.

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.

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 :

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

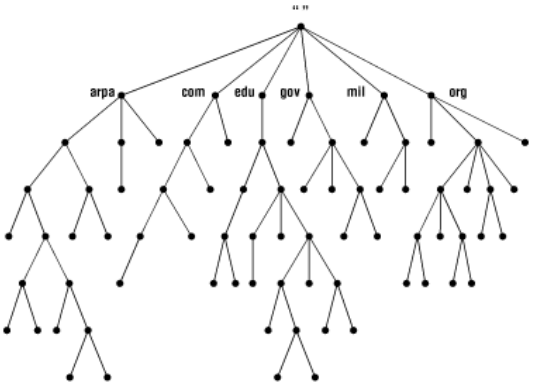
Dans un petit réseau la résolution directe et inverse peut se réaliser en modifiant à la main le fichier ***hosts***. A grande échelle, toutefois, l'utilisation du fichier ***hosts*** n'est pas pratique. Sur les grands réseaux on utilise plutôt le système ***Domain Name System (DNS)***.

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.

Les noms de domaine et leur hiérarchie

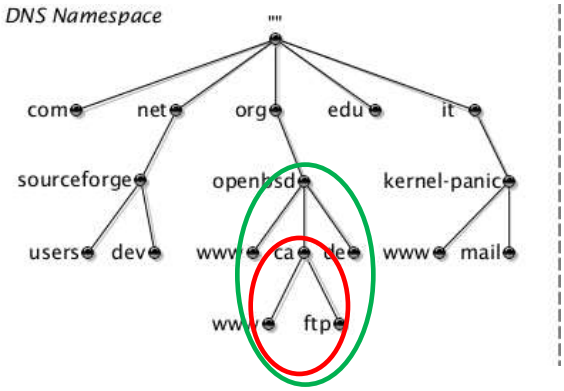
Le système DNS utilise une base de données distribuée, qui est structurée en tant qu'un arbre inverse qui s'appelle un «espace de nommage » (*namespace* en anglais). Cet espace de nommage DNS est visible ci-joint (***Source : docstore.mik.ua***).



La structure d'arbre indique une hiérarchie. Chaque nœud de l'arbre est identifié par un nom d'au max 63 caractères. Le nœud racine n'a pas un nom, sinon il est identifié seulement par un point (« . »).

La hiérarchie de nommage inclut au plus 127 niveaux (un nom a en total 253 caractères ASCII).

Chaque domaine est structuré à plusieurs niveaux. Le nom d'un domaine est la collection des noms de plusieurs nœuds, séparés par des points (par exemple ***mail.unilim.fr***). L'ordre de noms est inversée par rapport à la hiérarchie de l'arbre : premièrement on met les noms aux niveaux inférieurs et seulement après les noms aux niveaux supérieurs. Un domaine forme un sous-arbre, dont le nom commence au nom du nœud au plus haut dans la hiérarchie.



Dans l'exemple ci-dessus (***Source : kernel-panic.it***) on se concentre sur un sous-arbre sur 6 niveaux (y compris la racine). On peut distinguer par exemple un sous-domaine ***ca.openbsd.org.*** et un sous-domaine plus grand au nom de ***openbsd.org.***

Disons qu'on se trouve sur une machine ***ftp.ca.openbsd.org*** et on tape alors *ping www*. La machine cherchera dans le sous-domaine indiqué et trouvera ***www.ca.openbsd.org***.

Pour avoir un espace de nommage utilisable, il faut que chaque domaine ait un nom unique. C'est pourquoi deux nœuds d'un même sous-arbre ne peuvent pas avoir le même nom. Ceci est toutefois possible dans deux domaines différents.

L'organisation d'un nom de domaine

Les noms de domaines se structurent en plusieurs niveaux. Juste en dessous du niveau de la racine sont les niveaux les plus hauts (Top Level Domains -- TLD). En plus de son TLD, un nom de domaine aura également un nom de deuxième niveau et pourra avoir des noms d'un niveau plus bas encore (troisième, quatrième…).

Les TLDs peuvent être de plusieurs types :

- Le domaine spécial ***.arpa*** utilisé pour résoudre des problèmes d'infrastructure;
- Les domaines représentant des pays (niveau national) -- country-code TLD ou ccTLD -- comme par exemple ***.fr***, ***.de***, ***.ca***, ***.be***, etc.
- Des domaines génériques de premier niveau -- generic TLD ou gTLD -- comme par exemple ***.com***, ***.edu*** ou ***.mil***.
- Des domaines génériques restreints dont les sous-domaines font l'objet de quelques règles, notamment ***.biz***, ***.name*** et .***pro***.
- Des domaines sponsorisés de premier niveau, sTLD, dont les noms sont sponsorisés par de certaines organisations : par exemple ***.post*** est un domaine sponsorisé par Universal Postal Union.
- Le domaine de premier niveau de test, tTLD, ***.test***. Ni le niveau de test ni ses sous-domaines ne peuvent être enregistrés, donc ils peuvent être utilisés librement dans chaque autre domaine.

En général un domaine a plus qu'un seul niveau. Le niveau inférieur au TLD est le deuxième niveau (sous-domaine). Plus bas on trouve un domaine de troisième niveau (machine), et on peut continuer ainsi.

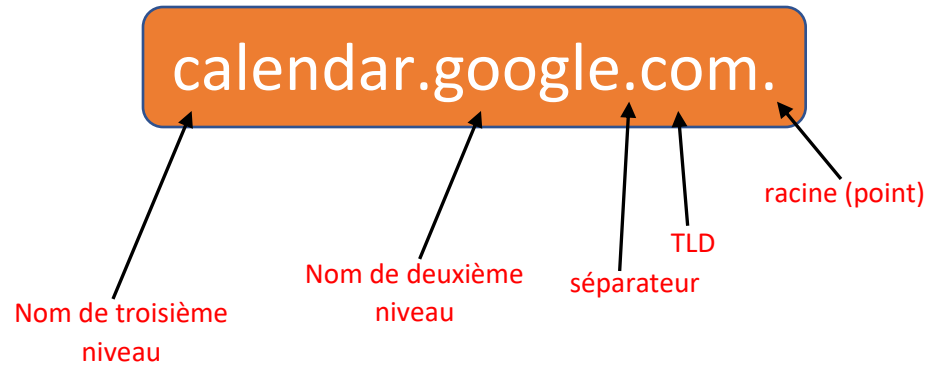
Les machines d'un domaine ne doivent pas forcement être réparties dans une même zone géographique : deux machines qui font partie de deux réseaux différents et sont localisés dans deux pays différents peuvent, néanmoins, faire partie d'un même domaine.

Nom absolu, nom relatif

Le nom absolu d'un domaine est un nom complet (Fully Qualified Domain Name, FQDN), qui finit toujours par un point. Même si à l'écriture on omet parfois le point final, ceci n'est pas possible lorsqu'on fait la configuration des serveurs DNS.

Une adresse IP s'écrit de gauche à droite. La partie la plus significative est celle de gauche, qui indique y compris la classe de réseau de laquelle elle fait partie.

Au contraire, le nom d'un domaine s'écrit de droite à gauche, avec la racine tout à gauche. Prenons l'exemple du domaine calendar.google.com. :



L'application DNS

Les machines qui gèrent les informations pertinentes aux espaces de noms s'appellent serveurs de nom (***name server***). Un serveur de nom contient des informations propres à une partie de l'espace complet de noms -- on appelle ce sous-espace la ***zone d'administration*** dont le serveur de nom est responsable. Un serveur de nom qui est responsable d'une certaine zone d'administration a l'autorité concernant toute requête et réponse sur des domaines inclus dans sa zone d'administration (***Authoritative answers***). Lorsque le propriétaire d'un domaine fait enregistrer son domaine, il reçoit de l'administrateur de la zone contenant son domaine une liste de serveurs de nom autoritaires sur la zone d'administration en question.

Pour être fiable et efficace, le système DNS doit être décentralisé, qui doit utiliser la délégation pour s'assurer que la responsabilité administrative peut le diviser en sous-domaines petits, qui peuvent être dirigés par des organisations indépendantes.

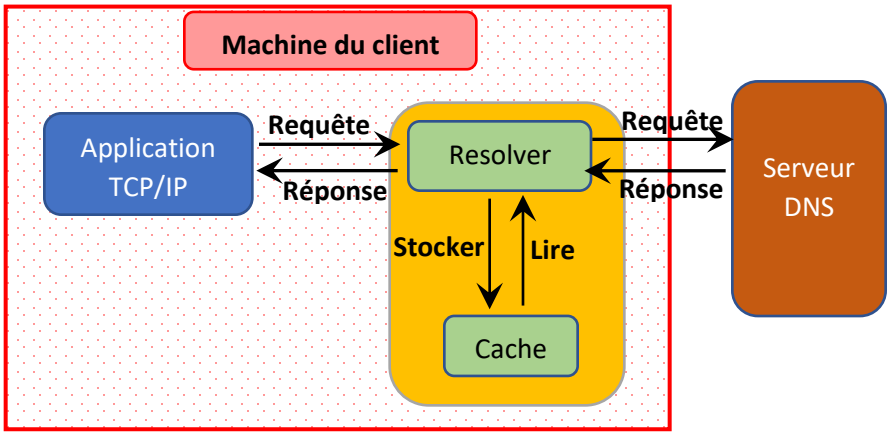
Les zones d'administration de l'espace de noms ne sont pas définies en fonction des domaines existants. C'est pourquoi un domaine divisé en plusieurs sous-domaines peut être administré par plusieurs serveurs de nom avec des zones d'administration différentes.

Si un serveur DNS reçoit une requête pour un domaine hors de sa zone d'autorité, alors il peut demander la résolution à un autre serveur DNS plus haut dans la hiérarchie de délégation.

3.1 Un service client-serveur

Le service DNS fonctionne en mode client-serveur. Une machine client fait une demande DNS (soit pour une résolution directe, soit pour une résolution inverse de nom), et une autre machine, jouant le rôle d'un serveur, donne la réponse à sa requête. La machine jouant le rôle du serveur est toujours un serveur DNS, tandis que la machine client peut être soit une machine d'un utilisateur, soit un serveur DNS demandant une résolution plus haut dans sa hiérarchie.

Lorsqu'une application TCP/IP sur une machine client a besoin d'une résolution DNS, elle contacte une application existante sur la même machine. Cette application s'appelle le ***resolver*** DNS. Celui-ci se trouve dans un fichier ***/etc/resolv.conf***.

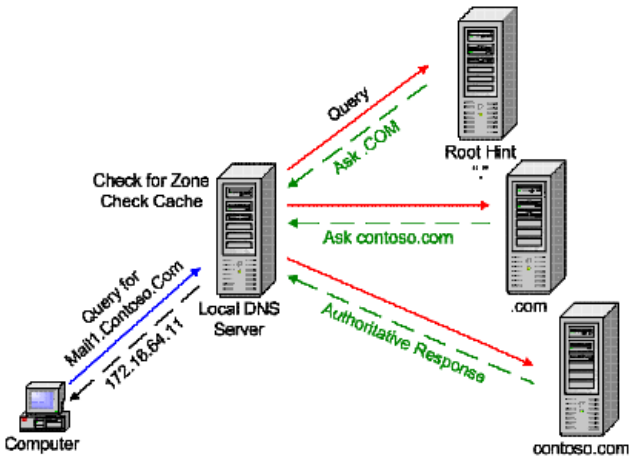


Dans l'illustration ci-dessus on voit l'interaction entre la machine du client et celle du serveur. Sur la machine du client, l'application TCP/IP fait premièrement sa requête vers le resolver. Le resolver cherche premièrement dans sa cache pour savoir si une valeur existe déjà pour l'adresse ou le domaine cherché. Si cette valeur n'existe pas, le resolver contacte le serveur DNS.

Dans le cas ci-dessus on suppose que le serveur, étant autoritaire sur le requête du resolver, peut bien répondre à cette requête. Dans ce cas-ci, la réponse du serveur sera stockée dans la cache du resolver, puis relayée à l'application TCP/IP.

Si le serveur DNS n'est pas autoritaire sur la requête et ne connait pas la réponse, alors il doit demander plus loin. S'il connait un serveur qui est autoritaire sur le sujet de la requête, alors il fait sa demande directement. Sinon, il va devoir premièrement trouver un tel serveur.

Dans ce dernier cas, le serveur va demander premièrement pour un serveur autoritaire pour le premier niveau du domaine dont on cherche l'adresse. Puis on demande plus loin, niveau par niveau.



Comme on peut voir dans cette figure, le rôle du serveur racine est essentiel pour trouver les informations que le client cherche. C'est pourquoi il est nécessaire d'avoir plusieurs serveurs de ce type, avec des redondances qui les aident à gérer les requêtes reçues.

4. Configuration d'un serveur DNS

Nous pouvons installer un serveur DNS à partir d'une implémentation qu'on peut trouver. L'implémentation la plus utilisée a été créée par l'Université de Berkeley et s'appelle Berkeley Internet Name Domain (BIND). Le daemon correspondant s'appelle ***bind*** ou ***named*** selon les versions (sur les machines netkit que vous allez utiliser il s'appelle ***bind9***).

Une fois l'installation réalisée on peut installer un ***name server*** primaire. Pour ce faire il faut utiliser le fichier ***/etc/bind/named.conf***. Ce fichier indique où se trouvent un nombre de fichiers, notamment :

- Le fichier utilisé pour la résolution directe : db.DOMAIN, qui donne la correspondance nom -> adresse IP.
- Le fichier utilisé pour la résolution inverse : db. ADDR, qui donne la correspondance adresse IP -> nom.

Voici un exemple du fichier named.conf :

```
zone "exemple.fr" {
      type master;
      file "/etc/bind/db.exemple.fr";
};
```

Le type indiqué est « master » . Ceci indique une référence vers le serveur DNS primaire, plutôt que par exemple un serveur secondaire.

La base de données DNS

Un serveur de DNS possède une base de données qui stocke des informations sur plusieurs domaines. Plusieurs informations peuvent être associées à chaque domaine, au-delà de juste son adresse IP. Si un resolver fait une requête concernant un certain domaine, alors le serveur DNS lui donnera toutes les informations (ou enregistrements -- « records » en anglais) qu'il possède concernant ce domaine.

On appelle les informations associées à un domaine les enregistrements d'une ressource (« resource records » en anglais). Un ***resource record*** consiste en cinq éléments : le nom du domaine, la durée de vie, la classe, le type et finalement une valeur. Nous détaillons ces valeurs ci-dessous.

Il faut savoir qu'en général un seul domaine correspondra à plusieurs entrées dans la base de données du serveur DNS. Chaque resource record apportera un peu plus d'information par rapport aux précédents.

- Le nom du domaine indexe les ressources stockées dans la base de données du serveur DNS. C'est à partir de ce nom de domaine que le serveur cherchera des données à retourner aux requêtes du client. Un serveur DNS peut contenir plusieurs entrées avec un même nom de domaine.
- La durée de vie indique la stabilité des informations associées à chaque domaine. Si l'information est très stable, alors sa durée de vie est mise à une valeur très large (comme par exemple 86400. Si l'information est susceptible à être beaucoup modifiée dans le temps, alors la valeur de la durée de vie est bien plus basse, comme par exemple 60 (le nombre de secondes dans une minute).
- La classe d'un ***resource record*** consiste en un code qui indique la classe d'un domaine. En pratique on utilise presque toujours le code IN, pour Internet. Parfois, les DNS BIND utilisent le code CH pour indiquer par exemple la version d'un serveur.
- Le type d'un ***resource record*** peut prendre une valeur parmi les suivantes :
    - SOA (Start of Authority) : déclare le serveur de noms ayant l'autorité sur la zone, ainsi que les données (adresse mail) de l'administrateur de la zone et de plus quelques informations concernant la potentielle expiration et les mises à jour de la base de données.
    - A (address) : une adresse IP associée au domaine en question. Si un domaine est associé en réalité avec plusieurs adresses, alors plusieurs entrées existeront dans la base de données, ayant le même nom de domaine. Le serveur DNS enverra alors l'adresse IP dans le premier RR pour une première requête, puis à la deuxième requête il enverra l'adresse du deuxième RR, etc.
    - CNAME (canonical name) : Dans cette entrée on peut définir des alias, par exemple pour s'assurer qu'une personne qui tente de joindre une adresse presque pareille à la nôtre, peut en fait nous joindre. Par exemple le sous-domaine informatique.iut.limousin.fr pourrait se faire un alias info.iut.limousin.fr pour que les gens tapant info.iut.limousin.fr puissent le joindre.
    - PTR (pointer) : Comme le CNAME, le PTR sert à associer un nom de domaine à une autre entrée. En pratique le PTR est utilisé pour la résolution inverse dans le domaine in-addr.arpa. Une feuille du domaine in-addr.arpa est un pointeur sur le nom de domaine correspondant.
    - NS (name server) : un serveur de noms pour le domaine en question.
    - MX (Mail exchanger) : ce champ indique quels serveurs de mail sont capables de recevoir de l'email pour un domaine donné. Si on configure mal ce champ, alors le mail ne pourra pas être livré.
    - HINFO (host info) et TXT : deux champs qui permettent d'ajouter des informations concernant la machine qui hôte le domaine. Le premier de ces champs indique des paramètres de la machine comme le CPU et le système d'exploitation. Le deuxième indique des informations arbitraires.
- La valeur est associée à la valeur du type indiquée ci-dessus. Son format diffère en fonction du type : il peut s'agir d'un entier à 32 bits (pour une adresse IP), mais aussi d'un nombre de paramètres (pour le champ SOA). Les valeurs correspondant à chaque type de RR sont listées dans le tableau ci-dessous.

Type	Signifiance	Valeur
SOA	Start of Authority	Des paramètres pour la zone d'autorité en question
A	Address	Entier à 32 bits
MX	Mail Exchanger	Un domaine qui acceptera des mails

NS	Name server	Le nom d'un serveur de noms pour ce domaine
CNAME	Canonical Name	Alias pour le nom du domaine
PTR	Pointer	Pointer vers une adresse IP
HINFO	Host info	Des informations en hardware : CPU, OS
TXT	Text	D'autres informations sur la machine hôte

Exemple

Voici un exemple de base de données d'un serveur DNS. Le domaine ciblé est le domaine cs.vu.nl. Dans un premier bloc nous avons des informations sur le domaine et le serveur d'autorité sur le domaine. Puis, on donne des informations sur une machine particulière, qui utilise un serveur avec un système d'exploitation Sun Unix, avec deux adresses IP. Finalement nous avons des informations sur une imprimante dont le nom est printer.

```
cs.vu.nl      86400  IN  SOA    Boss(9500,7200,7200,241920,86400)
cs.vu.nl      86400  IN  TXT    "Wiskunde en Informatica"
cs.vu.nl      86400  IN  TXT    "V.U. Amsterdam"
cs.vu.nl      86400  IN  MX     mailbox.cs.vu.nl.

mch.cs.vu.nl  86400  IN  HINFO  Sun Unix
mch.cs.vu.nl  86400  IN  A      130.37.16.20
mch.cs.vu.nl  86400  IN  A      192.31.231.157
mch.cs.vu.nl  86400  IN  MX     mailbox.cs.vu.nl

printer              IN  A      192.31.231.200
```

On peut également donner le contenu du fichier ***db.exemple.fr*** pour un domaine exemple.fr.

```
exemple.fr. IN SOA dns.exemple.fr.  root.dns.exemple.fr. (
      2019060201 # numéro de serie AAAAMMJJNN
      10800 # rafraîchissement
      3600 # nouvel essai
      604800 # obsolète au bout d'une semaine
      86400 # TTL 1 jour
)

exemple.fr.   IN NS dns.exemple.fr.

dns.exemple.fr.   IN   A       83.1.2.3
www.exemple.fr.   IN   A       83.1.2.4
ftp               IN   A       83.1.2.5

web.exemple.fr.   IN   CNAME   www.exemple.fr.
```

Exercice I

1. Donnez quelques avantages pour l'utilisation d'un fichier *hosts* pour faire la résolution directe et inverse. Puis, donnez quelques avantages de l'utilisation d'un serveur DNS.
2. Dans quel cas peut-il être utile de faire une résolution inverse de nom ?
3. Quel est le FQDN de la machine ***mail*** du domaine ***unilim.fr*** ?
4. Quel est le FQDN de la machine 164.81.20.22 lorsqu'on fait la résolution inverse ?
5. Quelle est la cible de la commande ***ping www*** sur la machine ***pc1.mondomaine.fr*** ?

Qu'est-ce qui se passe si on essaie de trouver la machine **www.** (avec le point final) ?