TCP/IP

Sites intéressants :

http://www.laissus.fr/cours/cours.html

<http://irp.nain-t.net/doku.php>

Livre de référence : <https://www.amazon.fr/R%C3%A9seaux-2014-Guy-Pujolle/dp/2212138520>

Le modèle OSI

APPLICATION

PRESENTATION

SESSION

TRANSPORT 🡪 UDP/TCP

RESEAU 🡪 IP, ARP, ICMP

LIAISON 🡪 Ethernet

PHYSIQUE 🡪 Les câbles.

Internet = Interconnexion des réseaux.

Adressage IP :

Classe A 🡪 1 octet pour le réseau et 3 octets pour les machines soit 256 réseaux et 16 777 216 - 2 machines possibles par réseau. 10.0.0.0 /8

Classe B 🡪 2 octets pour le réseau et 2 octets pour les machines soit 65536 réseaux et 65536 machines possibles par réseau – 2. 172.16.0.0 /16

Classe A 🡪 3 octets pour le réseau et 1 octet pour les machines soit 16 millions de réseaux et 254 machines possibles par réseau. 192.168.0.0/24

Adresse IP de la machine : 10..0.0.1

Masque de sous réseaux : 192.168.1.1

On passe l’adresse IP en binaire et on applique un ET logique entre ces deux valeurs afin d’avoir l’IP du réseau qui sera 10.0.0.0.

Si deux machines ne sont pas sur le même réseau, il va falloir utiliser une gateway ( le routeur) pour qu’elles puissent communiquer entre elles.

Le masque de sous réseau va nous permettre de découper un réseau en sous réseau.

GNS3 🡪 Afin de simuler une architecture réseau.

Voir la chaîne Youtube de David bombal

La couche 4 est celle de transport avec les protocoles UDP et TCP.

On y retrouve les ports, il y en a plusieurs car il y a divers services.

UDP 🡪 mode non connecté, pas d’accusé de réception

TCP 🡪 mode connecté, accusé de réception et contrôle de flux.

Services utilisant TCP : http (80), smtp (25), ftp(21), ssh(22), etc.

Services utilisant UDP : Voip, dns(53), ntp(123), etc.

**Le routage**

Dans ce chapitre, nous allons essayer de comprendre comment les informations transitent d'un réseau à un autre.

### Un protocole, IP

Pour rappel, un protocole est un langage. Il permet aux machines qui dialoguent ensemble de se comprendre.

Nous sommes la machine A qui a pour adresse 192.168.0.1/24 et nous souhaitons envoyer un message à une machine B d'adresse 192.168.1.1/24.  
Ce qui est important pour moi, en tant que machine A, c'est de savoir si la machine B est sur mon réseau. En effet, si elle est sur mon réseau, je lui parlerai grâce à la couche 2. Si elle est sur un autre réseau, il faudra que je fasse appel à la couche 3.

Dans notre cas, ma plage d'adresses va de 192.168.0.0 à 192.168.0.255. Elle ne contient donc pas l'adresse de la machine B (192.168.1.1).  
J'en déduis donc que B n'est pas sur mon réseau et qu'il va falloir utiliser la couche 3 pour communiquer avec elle.

Donc pour l'instant, nous n'avons besoin que de l'adresse IP de l'émetteur et de celle du récepteur. Nous les appellerons **adresse IP source** et **adresse IP destination**.

Pour le protocole IP, le message s'appelle un **datagramme ou un paquet**.

##### Le datagramme

Comme pour la couche 2, le datagramme IP va être une suite de 0 et de 1 organisés.

Format du datagramme IP :

|  |  |  |
| --- | --- | --- |
| Adresse IP SRC (source) | 🡪Adresse IP DST (destination) | 🡪Données à envoyer |
| L'encapsulation Envoi dans le modèle OSI |  |  |

Comme nous le voyons, un message est envoyé depuis la couche 7 du modèle OSI, et il traverse toutes les couches jusqu'à arriver à la couche 1 pour être envoyé sur le réseau.

En fait, un en-tête va être ajouté à chaque passage par une couche. On va ainsi accumuler les en-têtes des différentes couches

Au passage par la couche 4, on ajoutera l'en-tête de couche 4, puis celui de couche 3 en passant par la couche 3, et ainsi de suite.  
Ce mécanisme s'appelle **l'encapsulation**, car on encapsule un message dans un autre.

Nous voyons clairement qu'au final, ce qui va circuler sur le réseau est **une trame de couche 2**, **qui contient le datagramme de couche 3** (qui lui-même contiendra l'élément de couche 4).

Trame Ethernet

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresse MAC DST | Adresse MAC SRC | Protocole de couche 3 | en-tête de couche 3 | en-tête de couche 4 | Données à envoyer | CRC |

La couche 2 est incapable de lire les informations de couche 3 ou de couche 4, de même qu'elle ne comprend pas les données à envoyer. Pour elle, tout cela est une suite de 0 et de 1 qu'elle est incapable de comprendre, elle ne voit ça que comme des données...

Wireshark est un **sniffer**. Un sniffer est **un programme qui écoute sur le réseau**, intercepte toutes les trames reçues par votre carte réseau, et les affiche à l'écran.

Nous cherchions à comprendre pourquoi l'adresse IP de destination n'était pas en début d'en-tête IP ?

Quand un message arrive sur une machine, il remonte les couches du modèle OSI de la couche 1 à la couche 7. Il passe donc par la couche 2 qui lit l'adresse MAC de destination :

* si c'est bien celle de la carte réseau, il lit le reste de la trame, puis transmet les données (le datagramme en fait !) à la couche 3 ;
* si ce n'est pas celle de la carte réseau, il jette la trame à la poubelle.

Donc si le message arrive à la couche 3, cela veut obligatoirement dire que **la machine sait déjà que le message lui est destiné**, puisque l'adresse MAC de destination est la sienne. **Elle n'a donc pas la nécessité de savoir immédiatement si l'adresse IP de destination est la sienne**, puisqu'elle sait déjà que le datagramme est pour elle.

### Le routage

Le routage va nous permettre d'envoyer un message en dehors de notre réseau.

#### Le routeur

Une machine qui a plusieurs interfaces (plusieurs cartes réseau), chacune reliée à un réseau. Son rôle va être d'**aiguiller les paquets reçus entre les différents réseaux**.

Un ordinateur ayant deux cartes réseau **pourra** être un routeur.

Nous allons nous mettre dans la peau d'un routeur.

Imaginons que nous sommes une machine ayant comme adresse MAC l'adresse 00:11:22:33:44:55 et comme adresse IP 192.168.0.1/24.  
Nous recevons la trame suivante.

00:11:22:33:44:55 🡪 01:2B:45:56:78:ED 🡪 IP -> SRC -> IP SRC: 10.0.0.1 -> IP DST: 136.42.0.28 -> Data à envoyer 🡪 CRC

Quelle est l'adresse IP de la machine qui a envoyé ces informations ?

Cette adresse IP est bien l'adresse IP source 10.0.0.1.

Quelle est l'adresse MAC de la machine qui a envoyé ces informations ? (Attention au piège !)

**Nous ne pouvons pas la connaître**

Une adresse MAC est propre à un réseau local. En dehors de ce réseau, nous ne la voyons pas. Justement ici, la trame arrive sur l'interface de notre machine ayant pour adresse IP 192.168.0.1/24. Son réseau ne contient donc pas l'adresse IP 10.0.0.1.  
La machine 10.0.0.1 ne fait pas partie de notre réseau et nous ne connaîtrons **jamais** son adresse MAC.  
L'adresse MAC que nous voyons ici en adresse MAC source est celle du dernier routeur qui nous a envoyé la trame. Nous allons approfondir tout cela par la suite.

Que se passe-t-il quand notre machine reçoit cette trame ?

La trame arrive à ma carte réseau qui reçoit les 0 et les 1 et les envoie à mon système d'exploitation. La couche 2 de mon système d'exploitation reçoit les 0 et les 1 et les interprète pour me donner l'adresse MAC de destination de la trame.  
C'est **mon adresse MAC** 00:11:22:33:44:55 !  
Donc je lis la suite de l'en-tête de la trame pour voir qui m'envoie cette trame et à quel protocole de couche 3 la couche 2 doit l'envoyer. Il est inscrit IP, donc j'envoie la trame en enlevant l'en-tête Ethernet, ce qui donne le datagramme IP, à la couche 3 et plus précisément au protocole IP.  
La couche 3, donc le protocole IP, lit l'ensemble des informations de l'en-tête IP, puisque nous savons maintenant que ce datagramme nous est destiné.  
Et là, badaboum, l'adresse IP de destination du datagramme n'est pas la nôtre...

Ceci étant, ce n'est pas grave, car nous avons vu auparavant qu'il est normal pour un routeur de recevoir un message qui ne lui est pas destiné.  
Son rôle va maintenant être d'aiguiller le datagramme vers sa destination.

Il possède en fait **une table** dans laquelle est indiqué le prochain routeur auquel il doit envoyer le datagramme pour que celui-ci arrive à sa destination.  
Cette table est très importante et s'appelle **la table de routage** !

##### La table de routage

La table de routage va donc lister les routeurs auxquels je peux envoyer mon datagramme pour joindre une destination donnée.  
La destination donnée ne va pas être une machine, mais un réseau.

Le principe est d'avoir d'un côté la liste des réseaux que l'on veut joindre, et de l'autre la liste des routeurs à qui nous devons envoyer le datagramme pour joindre ces réseaux. On appelle aussi ces routeurs des **passerelles**, car ils servent de "passerelle" entre deux réseaux.

Voici un exemple de **table de routage** :

|  |
| --- |
| Table de routage |
| Réseau à joindre | passerelle |
| 192.168.1.0/24 | 10.0.0.253 |
| 192.168.122.0/24 | 10.0.0.45 |
| 192.168.8.0/24 | 10.0.0.254 |

On récapitule :

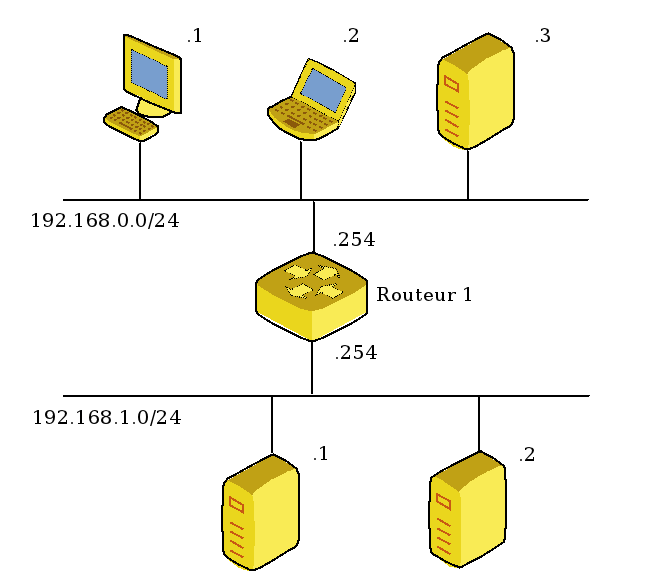
* un routeur est une machine possédant **plusieurs interfaces** ;
* chaque interface d'un routeur est connectée à un réseau, **le routeur relie ainsi plusieurs réseaux entre eux** ;
* toute machine ayant plusieurs interfaces peut jouer le rôle de routeur, même le vieux PC de mamie ;
* un routeur se différencie d'une simple machine, car **il accepte de relayer des paquets qui ne lui sont pas destinés** ;
* un routeur aiguille les paquets grâce à sa **table de routage** ;
* la table de routage indique **quelle passerelle utiliser pour joindre un réseau**.

Il est important de bien **comprendre et retenir ce qui précède**, car le routage est la base du fonctionnement d'Internet !

##### La route par défaut

La première info dans une table de routage n'est pas un réseau qui est indiqué, mais le mot défaut.  
Cela indique que **si une adresse que je veux joindre n'appartient à aucun des réseaux indiqués dans ma table, il faudra emprunter la passerelle indiquée dans la route par défaut**.

Voici en figure suivante un schéma réseau qui contient plusieurs réseaux. Nous allons essayer d'écrire les tables de routage du routeur.



Sur ce schéma, nous voyons deux réseaux (192.168.0.0/24 et 192.168.1.0/24) reliés entre eux grâce au routeur 1 qui possède une interface réseau dans chacun de ces réseaux.

Par exemple pour la machine en haut à gauche d'adresse .1 qui est dans le réseau 192.168.0.0/24, on peut déduire son adresse complète qui est **192.168.0.1**.

Ecriture de la table de routage du routeur 1.

Pour cela, je vais vous donner une méthode qui s'appliquera **toujours et qui fonctionnera pour tous les cas** :

1. indiquer les **réseaux auxquels ma machine est connectée** ;
2. indiquer **la route par défaut** ;
3. indiquer **tous les autres réseaux** que je ne peux pas encore joindre avec les deux étapes précédentes.

Appliquons la méthode.  
1 - Indiquer les **réseaux auxquels ma machine est connectée**.  
Mon routeur 1 est connecté à deux réseaux, 192.168.0.0/24 et 192.168.1.0/24.

Passons à la seconde étape.  
2- Indiquer **la route par défaut**.  
Le cas est un peu particulier, car notre routeur est déjà connecté à tous les réseaux du schéma. Il n'a donc pas besoin d'une route par défaut pour aller vers d'autres réseaux, il les connaît déjà tous !

3- Indiquer **tous les autres réseaux** que je ne peux pas encore joindre avec les deux étapes précédentes.  
Même chose que la réponse précédente, il n'y a pas de réseau supplémentaire à indiquer.

Il nous reste à y indiquer les passerelles. Pour cela, nous allons appliquer une règle simple : la passerelle pour joindre un de **mes** réseaux est **mon** adresse.

La table de routage sera donc :

|  |
| --- |
| Table de routage du routeur 1 |
| Réseau à joindre | passerelle |
| 192.168.0.0/24 | 192.168.0.254 |
| 192.168.1.0/24 | 192.168.1.254 |

Maintenant, est-ce que cela suffit pour faire dialoguer nos deux réseaux entre eux ?

Non, car il faut aussi paramétrer une table de routage sur les machines. **Toute machine connectée à un réseau possède une table de routage**, même une imprimante, un téléphone, ou le vieux PC de mamie...

C'est grâce à cette table de routage qu'une machine peut savoir à quelle passerelle envoyer un paquet quand elle veut joindre un autre réseau que le sien. On peut donc reprendre le schéma précédent et, par exemple, faire la table de routage de la machine 192.168.0.1.

|  |
| --- |
| Table de routage de 192.168.0.1 |
| Réseau à joindre | passerelle |
| 192.168.0.0/24 | 192.168.0.1 |
| défaut | 192.168.0.254 |

Réseau avec plusieurs routeurs

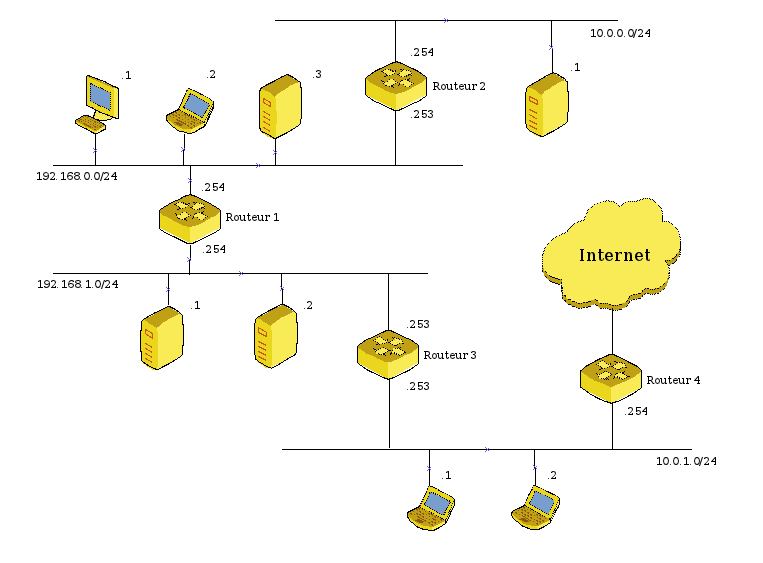


Table de routage du routeur 1

Réseau à joindre Passerelle

192.168.0.0 192.168.0.254

192.168.1.0 192.168.1.254

défaut 192.168.1.253

10.0.0.0 192.168.0.253

Inutile de mettre la route pour le réseau 10.0.1.0 car il peut y accéder via la passerelle par défaut.

Il existe une autre écriture possible pour la route par défaut qui est parfois identifiée par le réseau 0.0.0.0/0

Table de routage de 192.168.0.1

Réseau à joindre Passerelle

192.168.0.0 192.168.0.1

0.0.0.0 192.168.0.254

10.0.0.0 192.168.0.253

### Mise en pratique du routage

##### L'architecture

En réseau, on parle souvent d'architecture pour indiquer comment les machines sont branchées entre elles.

Si on a 3 machines dont 2 sur deux réseaux différents, on doit en prendre une comme routeur qui aura donc 2 IP, une reliée à chaque réseau.

Sur les autres machines il faudra préciser que la machine routeur est leur gateway.

Pour qu’une machine se comporte comme un routeur, après lui avoir mis les cartes réseaux propres aux différents réseaux il faut **activer le routage**. Pour cela, c'est très simple, car il suffit de mettre 1 à la place de 0 dans un fichier en tapant cette commande entièrement :

echo 1 > /proc/sys/net/ipv4/ip\_forward

Pour ajouter une route par défaut sous debian on tape :

route add default gw ipDuServeur ex : route add default gw 192.168.0.254

* Il faut toujours penser qu'on ne peut joindre une machine **QUE si le routage fonctionne dans les DEUX SENS**.
* Il est souvent intéressant d'écrire les tables de routage sur papier avant de mettre en place une infrastructure pour éviter que cela ne fonctionne pas, une fois mis en place.

Pour créer une route qui ne soit pas une route par défaut, donc pour un réseau spécifique, la syntaxe est :  
route add -net 192.168.10.0 netmask 255.255.255.0 gw 192.168.11.254

Et pour enlever une route :  
route del -net 192.168.10.0 netmask 255.255.255.0

### Le protocole ARP

ARP est un protocole qui permet d'associer une adresse MAC de couche 2 à une adresse IP de couche 3 et ce en utilisant une requête ARP ou broadcast ARP.

##### La table ARP

Pour éviter d'avoir à renvoyer en permanence des broadcasts ARP à chaque fois que l'on veut envoyer une information à une machine, nous allons utiliser une table qui va garder les associations adresses IP <-> Adresses MAC pendant un court moment.  
Ainsi, si j'envoie un paquet à ma passerelle, je noterai son adresse MAC dans ma table ARP et la prochaine fois que je voudrai lui parler, je n'aurai plus à envoyer de broadcast sur le réseau.

La table ARP va donc **associer adresse IP et adresse MAC correspondante**.

Une valeur va rester environ deux minutes dans la table avant d'être effacée s'il n'y a pas eu de dialogue avec cette adresse entre-temps. C'est pour cela que l'on dit que **la table ARP est dynamique**. Elle évolue au cours du temps en fonction des machines avec lesquelles je dialogue.

La commande sous Unix pour voir sa table ARP est **arp –an**

##### Déroulement de A à Z d'une requête ARP

Reprenons l'exemple précédent : nous sommes la machine 192.168.0.1 et voulons envoyer un message à la machine 192.168.1.2.  
Nous savons que nous voulons joindre d'abord le routeur 192.168.0.254, mais ne connaissons pas son adresse MAC.  
C'est là que le protocole ARP entre en jeu :

* on regarde d'abord dans la table ARP locale si on possède l'association entre l'adresse IP 192.168.0.254 et son adresse MAC ;
* si on la possède, on envoie l'information et c'est terminé ;
* sinon, on envoie un broadcast ARP sur le réseau ;
* la machine 192.168.0.254 va nous répondre avec son adresse MAC ;
* nous allons noter cette adresse MAC dans notre table ARP ;
* nous allons enfin pouvoir envoyer notre information.

Le protocole ARP est un protocole de couche... 2 ET 3 !  
Oui, il manipule des informations de couche 2, les adresses MAC, et des informations de couche 3, les adresses IP. Ainsi, on dit que ce protocole est "à cheval" entre ces deux couches.

Les adresses MAC présentes sont obligatoirement celles du réseau sur lequel la trame est en train de circuler.

### Récapitulons tout cela !

La machine 192.168.0.1 veut faire une requête web vers la machine 192.168.1.2

Le message démarre de la couche 7, puis descend à la 6, 5, 4, 3

Dans la 3 il regarde la table de routage et sait qu’il doit envoyer le message à 192.168.0.254 pour sortir de son réseau.

Il fait une requête ARP afin de connaître l’adresse MAC de 192.168.0.254

Maintenant la trame va être formée avec l’IP de destination 192.168.1.2

Le message sort de l’ordinateur et se trouve sur le câble.

Puis il arrive au switch qui va le rediriger vers le routeur

Le routeur va lire la MAC de destination, c’est la sienne, il va garder que le datagramme IP, voir que l’ip de destination n’est pas la sienne et donc renvoyer la trame en regardant dans sa table de routage à quelle gateway l’envoyer. Il verra que l’adresse de destination est sur son propre réseau

Cependant, pour envoyer la trame sur le réseau, il va avoir besoin de l'adresse MAC de 192.168.1.2. Il va donc faire une requête ARP.  
Une fois l'adresse MAC de 192.168.1.2 reçue, il va pouvoir former la trame et l'envoyer sur le réseau.

La trame sort du routeur et arrive sur un switch

Le switch va regarder l'adresse MAC de destination et aiguiller la trame vers la machine 192.168.1.2.

La machine 192.168.1.2 va recevoir la trame en couche 2 et va lire l'adresse MAC de destination.  
C'est la sienne. Elle va donc lire la suite de l'en-tête et renvoyer le datagramme contenu dans la trame à la couche 3, c'est-à-dire au protocole IP.  
La couche 3 reçoit le datagramme et lit l'en-tête.  
**L'adresse IP de destination est la sienne**, elle va donc envoyer les informations à la couche 4, qui va elle-même envoyer les informations à la couche 7 applicative.

Et le message est enfin reçu

### Le protocole ICMP

Le protocole ICMP, lui, ne va pas non plus concurrencer le protocole IP, car son objectif n'est pas de transporter de l'information.   
Son rôle est de **contrôler les erreurs de transmission**, et d'**aider au débogage réseau**.

En gros, son utilisation nous permet de **comprendre rapidement d'où peut venir un problème réseau**, et de nous donner des outils pour **investiguer un problème réseau**.

##### Fonctionnement du protocole

Il y a globalement deux rôles principaux pour le protocole ICMP :

* ICMP sert à indiquer **automatiquement** des erreurs quand elles surviennent ;
* ICMP peut **fournir des outils** pour étudier un problème réseau.

##### Les messages automatiques

Il y a deux informations qui nous intéressent dans l'en-tête ICMP, le **type** et le **code**. Le type permet de dire à quoi sert le message ICMP, le code permettant de préciser le rôle du message.

* type 5, ICMP redirect, indique qu'il y a un chemin plus court vers la destination ;
* type 11, TTL exceeded, indique que la durée de vie du paquet a expiré.

Nous avons déjà vu un TTL, c'était celui de la table CAM du switch. Il indiquait la **durée de vie** d'une information dans la table.   
Eh bien un mécanisme équivalent a été implémenté dans le protocole IP pour éviter que les paquets ne circulent indéfiniment entre différents routeurs.

Imaginons qu'un routeur A ait comme passerelle par défaut un routeur B, et que le routeur B ait comme passerelle par défaut le routeur A.  
Un paquet envoyé à l'un des routeurs à destination d'un autre réseau va circuler alternativement d'un routeur à l'autre, comme une balle de ping-pong, sans jamais s'arrêter. Après quelque temps, beaucoup de paquets feront de même, et le réseau sera saturé.

Pour éviter ce problème, on a implémenté un système de TTL dans l'en-tête IP. Quand une machine envoie un paquet IP sur le réseau, un des éléments de l'en-tête est le TTL qui est une valeur entre 0 et 255. Par exemple, tout paquet envoyé depuis un Linux a un TTL de 64, cela varie d'un système à l'autre, 64 étant la plus petite.  
À chaque passage par un routeur, celui-ci **va enlever 1 au TTL**. **Si le TTL arrive à 0, il jette le paquet à la poubelle** ET envoie un message d'erreur ICMP "TTL exceeded".

##### Messages utiles pour déboguer le réseau

Ces paquets ICMP vont en fait nous être utiles pour des commandes qui vont nous permettre de déboguer des problèmes réseau. Or, ces commandes, nous les connaissons déjà...

Il s'agit de la commande ping et de la commande traceroute.  
Le ping est en fait la combinaison de deux types de messages ICMP, un echo request, type 8, et un echo reply, type 0.   
Le principe du ping est qu'une machine envoie un echo request, auquel répond une machine destinataire avec un echo reply. C'est pour cela que quand on arrive à pinguer une autre machine, on sait que le routage est correct dans les deux sens.

Pour le traceroute, c'est un peu plus compliqué.  
On utilise en fait une petite astuce en se servant d'un message automatique ICMP, le TTL exceeded.

Imaginons que je veuille faire un traceroute vers le Site du Zéro. Comment connaître le premier routeur par lequel je passe ?  
On pourrait aller voir dans notre table de routage, mais cela ne fonctionnerait que pour le premier routeur...  
L'indice nous dit d'utiliser un paquet ICMP TTL exceeded. L'idée pourrait donc être de faire générer ce paquet par le premier routeur. Ainsi, en voyant ce message d'erreur, je pourrais voir l'adresse du routeur dans ce paquet.

Comment faire pour faire générer ce message d'erreur par le premier routeur ?

Il suffit de mettre un TTL à 1 dans le paquet envoyé.  
Le premier routeur va le recevoir, décrémenter le TTL de 1 et donc le mettre à 0. Il devra jeter le message à la poubelle et me renvoyer un message d'erreur ICMP TTL exceeded. Ainsi, je pourrai connaître son adresse IP !

Si vous avez compris le principe, pour connaître l'adresse du second routeur, il me suffira de mettre le TTL à 2 dans le paquet envoyé. Et ainsi de suite pour connaître tous les routeurs traversés !

11111111 11111111 11111111 10000000

## Rendre mes applications joignables sur le réseau

### La couche 4, ses rôles

Grâce à la couche 2, nous savons dialoguer sur un réseau local.  
Grâce à la couche 3, nous savons dialoguer entre réseaux.  
Nous sommes donc capables de dialoguer entre deux machines sur des réseaux distants, chacune à un bout du monde.

Le rôle de la couche 4 est donc de **gérer les connexions applicatives**.

### Un identifiant, le port

#### Le port

##### Définition

L'adresse MAC identifie la carte réseau en couche 2, et l'adresse IP identifie l'adresse de notre machine au sein d'un réseau, en couche 3.

Eh bien en couche 4, **l'adresse utilisée est le port**

Le port est une adresse. C'est même **l'adresse d'une application sur une machine**.

MySQL a pour port d’écoute 3306. Si je reçois une requête MySQL sur l'adresse IP 127.0.0.1 et sur le port 3306, le service MySQL va pouvoir répondre.

Le port 80 est le port utilisé pour les serveurs web. Nous pouvons donc nous douter qu'**un serveur web est en écoute sur cette machine** ! Il ne nous reste plus qu'à utiliser un client web, soit un simple navigateur, pour nous connecter à cette application.

Un client web, **fait toujours ses requêtes sur le port 80** si un port n'est pas spécifié.

#### Quelles adresses pour les ports ?

##### Beaucoup de ports !

Les ports sont codés en **décimal sur deux octets**.

Ils peuvent donc prendre 216 vale‌urs, soit 65536 valeurs !  
Vu que l'on commence l'adressage des ports à 0, nous pourrons avoir des valeurs de ports **de 0 à 65535**.  
Ça fait quand même pas mal, non ?

Donc nous pourrons faire tourner au maximum **65536 applications en réseau sur une machine**. Cela devrait aller... mais on peut quand même parfois arriver à saturation, en cas d'attaque, quand quelqu'un envoie des tonnes de paquets sur nos différents ports pour nous saturer.

##### Liste des ports

Voici une petite liste des ports réservés, et des applications associées, les plus couramment utilisés :

| **Application** | **Port réservé** |
| --- | --- |
| web | 80 |
| mail | 25 |
| ssh | 22 |
| imap | 143 |
| proxy | 8080 |
| https | 443 |
| Counterstrike | 27015 |
| ftp | 20/21 |
| dns | 53 |

Il y a donc 65535 ports réservés pour les applications ?

Non, seule une partie d'entre eux sont réservés. D'ailleurs, historiquement, ce n'était que les ports inférieurs à 1024 qui étaient réservés. Mais aujourd'hui, beaucoup d'applications qui sortent utilisent des ports au-delà de 1024.

Mais quid des applications clientes ? Ont-elles aussi une adresse avec un port ?

Eh bien les applications clientes ont des ports, elles aussi, mais ils ne sont pas réservés.  
Les ports attribués aux applications clientes **sont donnés aléatoirement, au-dessus de 1024,** par le système d'exploitation.

Ce n'est pas gênant. Pour un serveur, vu qu'il est en écoute en permanence, il est important que l'on connaisse le port auquel on doit s'adresser. Pour un client, l'application ne va être en écoute que le temps de son fonctionnement. Ainsi, il peut être choisi au hasard tant que le système d'exploitation sait quelle application se trouve derrière quel port.

Et voilà comment nous pouvons dialoguer entre applications client/serveur grâce aux ports !

### Deux protocoles, TCP et UDP

fff

v