**TCP/IP**

Sites intéressants :

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

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

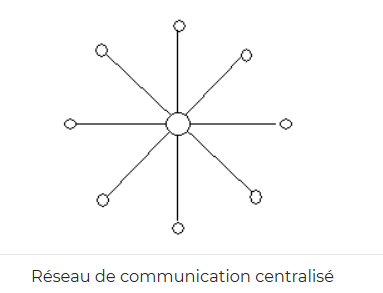
Livre de référence : <https://www.amazon.fr/R%C3%A9seaux-2014-Guy-Pujolle/dp/2212138520>

# L’histoire d’Internet

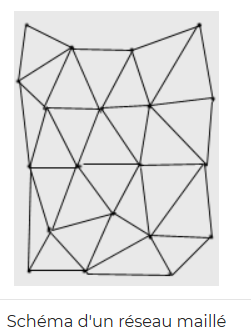
## Une histoire de mailles

Internet a été créé au départ pour une raison bien particulière.  
À l'époque, dans les années 1950, les communications étaient « point à point », c'est-à-dire qu'on ne pouvait communiquer qu'avec une seule machine à la fois. Les chercheurs qui devaient communiquer avec plusieurs autres chercheurs lors de réunions, se sont rendu compte qu'il serait intéressant de pouvoir le faire en temps réel plutôt que de passer d'un interlocuteur à l'autre successivement.

Ils ont donc cherché à créer un nouveau moyen de communication qui ne serait alors plus centralisé, **mais maillé**.



Cela veut dire que toute information pourrait passer par différents points, et que si certains points disparaissaient, cela n'empêcherait pas l'information de circuler. Observez donc la figure suivante : vous voyez qu'avec un réseau de communication maillé, si un point de communication n'est plus en état de fonctionner, l'information peut passer par un chemin différent.



Mais maintenant que l'idée est posée, il reste à la mettre en œuvre !

Les chercheurs vont travailler et notamment mettre en place un réseau pour l'armée. C'est seulement au **début des années 1960** que l'on voit apparaître des textes décrivant les prémices de ce que sera Internet.

À la **fin des années 1960**, l'Arpanet, l'ancêtre d'Internet, ne comportait que quatre machines ! Les protocoles utilisés alors ne permettaient pas d'atteindre les buts fixés, à savoir de faire dialoguer des machines provenant de différents réseaux en utilisant différentes technologies de communication.

C'est alors que les chercheurs se sont orientés vers la création d'autres protocoles de communication, et notamment TCP/IP. Internet a continué de croître au fil des années, mais c'est **en 1990** qu'une révolution va permettre sa croissance réelle : le langage HTML et le protocole d'échange HTTP qui permettent la création de pages web.

Tout va s'accélérer alors avec la création des premiers navigateurs capables d'afficher des images, et la libération de l'utilisation des noms de domaine. Nous pouvons voir sur la figure suivante la progression phénoménale d'Internet dans les **années 1990-2000.**

# Internet aujourd’hui

Des millions de serveurs et des milliards d’utilisateurs

# La création d’internet, le modèle OSI

## Comment communiquer ?

Internet se propose donc de relever le défi de pouvoir communiquer tous ensemble, en même temps, et ce, quand nous le souhaitons.

Faisons un petit inventaire des moyens de communication:

* la parole ;
* le téléphone ;
* le courrier ;
* etc.

Essayons maintenant de comprendre, parmi ces moyens de communication, ce dont nous avons besoin pour communiquer.  
Pour la parole, nous avons besoin:

* d'un émetteur ;
* d'un récepteur ;
* d'un support de transmission (l'air).

Pour le téléphone, c'est un peu pareil sauf que nous avons besoin d'un élément complémentaire qui est l'intermédiaire entre la parole et l'électronique. En effet, on transforme la parole en signaux électriques, ils arrivent côté récepteur, puis ils sont de nouveau transformés en paroles. Nous voyons qu'il y a une *encapsulation* de l'information.

Nous retrouvons ce système d'encapsulation dans le courrier, pour lequel nous avons besoin:

* d'un émetteur ;
* d'un récepteur ;
* d'un support de transmission (la lettre) ;
* d'un contenant (l'enveloppe) ;
* d'un intermédiaire (la poste).

Ainsi, nous commençons à comprendre ce qu'il nous faut pour communiquer.

Maintenant, est-ce que cela va pouvoir s'appliquer aux ordinateurs ? Comment va-t-on faire pour parler tous en même temps ? Pourra-t-on communiquer avec l'autre bout du monde instantanément ?

Nous allons voir par la suite comment les chercheurs ont fait pour passer des principes de communication humains à des principes de communication pour ordinateurs.

Ils ont ainsi regroupé l'ensemble de leurs recherches et de leurs résultats dans une norme que devront respecter les personnes se connectant à Internet.  
Il s'agit du **modèle OSI** !

## Le modèle OSI

Le modèle OSI est né en 1984. Les plus connaisseurs d'entre vous auront remarqué que celui-ci est né après la naissance d'Internet !  
La raison est simple : le modèle OSI est né quand nous avons commencé à avoir une certaine expérience des communications entre ordinateurs. Il tient donc compte des communications existantes, mais aussi des communications futures et de leurs évolutions potentielles.

Son objectif est de normaliser les communications pour garantir un maximum d'évolutivité et d'interopérabilité entre les ordinateurs.

Le modèle OSI est une norme qui préconise comment les ordinateurs devraient communiquer entre eux, c’est un modèle en couche, il y en a 7 :

APPLICATION

PRESENTATION

SESSION

TRANSPORT 🡪 UDP/TCP

RESEAU 🡪 IP, ARP, ICMP

LIAISON 🡪 Ethernet

PHYSIQUE 🡪 Les câbles.

**Cartes d'identité des couches du modèle OSI**

La couche 1 ou **couche physique** :

* Nom : physique.
* Rôle : offrir un support de transmission pour la communication.
* Rôle secondaire : RAS.
* Matériel associé : le *hub*, ou concentrateur en français.

La couche 2 ou **couche liaison** :

* Nom : liaison de données.
* Rôle : connecter les machines entre elles sur un *réseau local*.
* Rôle secondaire : détecter les erreurs de transmission.
* Matériel associé : le *switch*, ou commutateur.

La couche 3 ou **couche réseau** :

* Nom : réseau.
* Rôle : interconnecter les réseaux entre eux.
* Rôle secondaire : fragmenter les paquets.
* Matériel associé : le routeur.

La couche 4 ou **couche transport** :

* Nom : transport.
* Rôle : gérer les connexions applicatives.
* Rôle secondaire : garantir la connexion.
* Matériel associé : RAS.

La couche 5 ou couche session : on s'en fiche !

Oui, vous m'avez bien lu, au-delà de la couche 4, on s'en fiche ! Bon, j'exagère un poil, mais pas tant que ça.  
La raison est simple : le modèle OSI est un modèle théorique. Le modèle sur lequel s'appuie Internet aujourd'hui est le modèle TCP/IP. Or, ce modèle n'utilise pas les couches 5 et 6, donc... on s'en fiche !

Bon, je crois que vous avez compris ! Par contre, la couche 7 existe bien. Et c'est pour elle que nous mettons tout cela en place, le grand manitou, le patron, **l'application** !

La couche 7 ou **couche application** :

* Nom : application.
* Rôle : RAS.
* Rôle secondaire : RAS.
* Matériel associé : le proxy.

Règles d’or du modèle OSI

**Chaque couche est indépendante**

L'impact sera que les informations utilisées par une couche ne pourront pas être utilisées par une autre couche.

**Chaque couche ne peut communiquer qu'avec une couche adjacente**

Pour comprendre cette règle, vous allez devoir comprendre comment les machines se servent du modèle OSI pour communiquer.  
Vous êtes devant votre ordinateur et votre navigateur préféré. Vous entrez l'adresse d'un site dans la barre d'adresses, et le site apparaît aussitôt.  
Sans le savoir, vous avez utilisé le modèle OSI !  
En gros, l'application (le navigateur) de couche 7, s'est adressée aux couches réseau pour que celles-ci transmettent l'information à l'application demandée sur la machine demandée (le serveur web sur la machine google.com par exemple).

Lors d'un envoi, nous parcourons donc les couches du modèle OSI de haut en bas, de la couche 7 à la couche 1

En réalité la couche 7 s'adresse directement aux couches réseau pour communiquer, soit directement à la couche 4, qui s'adresse à la couche 3, qui s'adresse à la couche 2...

**Ce qu'il faut retenir**

* Le modèle OSI est une norme précisant comment les machines doivent communiquer entre elles.
* C'est un modèle théorique, le modèle réellement utilisé étant le modèle TCP/IP.
* Le modèle OSI possède 7 couches.
* Chaque couche a un rôle particulier à accomplir.
* Les couches 1 à 4 sont les couches réseau.
* Les couches réseau offrent le service de communication à la couche applicative.
* Chaque couche est indépendante des autres.
* Chaque couche ne peut communiquer qu'avec une couche adjacente.
* Lors de l'envoi de données, on parcourt le modèle OSI de haut en bas, en traversant toutes les couches.
* vous connaissez et comprenez maintenant le modèle OSI ;
* il est composé de 7 couches chacune dédiée à un rôle particulier ;
* Il y a deux règles d'or associées à ce modèle qui permettent de garantir la bonne utilisation du modèle OSI.

# Brancher les machines, la couche 1

## La couche 1 et ses rôles

Le rôle principal de la couche 1 est de **fournir le support de transmission de la communication**.  
Eh oui, pour pouvoir communiquer il va bien falloir avoir un support. Vous en connaissez déjà un si vous êtes connectés à Internet : un câble si vous êtes connectés directement à votre box, l'air libre si vous utilisez le wifi.

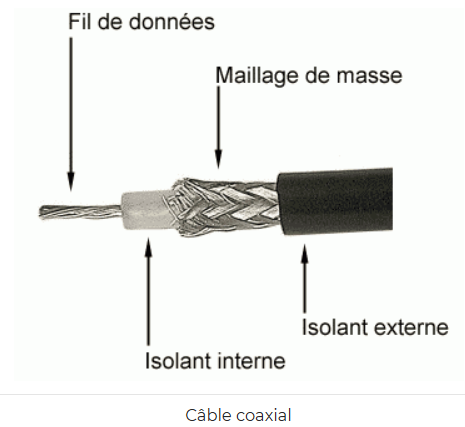
La couche 1 aura donc pour but d'acheminer des signaux électriques, des 0 et des 1 en gros.

0 = 0V et 1 = 5V

## Les matériels, câbles, etc.

Les 0 et les 1 vont circuler grâce aux différents supports de transmission

**Les câbles coaxiaux**



Le principe est de faire circuler le signal électrique dans le fil de données central. On se sert du maillage de masse, autrement appelé *grille*, pour avoir un signal de référence à 0 V. On obtient le signal électrique en faisant la *différence de potentiel* entre le fil de données et la masse.

Le nom scientifique donné au câble coaxial est donc le 10B2 ou 10B5 pour sa version encore plus ancienne.

* le 10 indique le débit en Mbps (mégabits par seconde) ;
* le B indique la façon de coder les 0 et les 1, soit ici la bande de Base ;
* le dernier chiffre indique la taille maximale du réseau, exprimée en mètres et divisée par 100.

Cette taille est de 200 m pour le 10B2, et 500 m pour le 10B5. Par exemple, pour une longueur de 200 m, si je divise par 100, cela me donne 2. Le nom scientifique est donc bien 10B2 !

**Le câble coaxial 10B2**

Le câble coaxial 10B2 possède la même structure que le 10B5, mais en plus fin. La connectique utilisée est aussi très différente, car la propagation de l'information ne se fait pas de la même façon.  
Pour mettre en place un réseau en 10B2, il fallait :

* des câbles 10B2 équipés de prises BNC ;
* des tés BNC ;
* des bouchons.

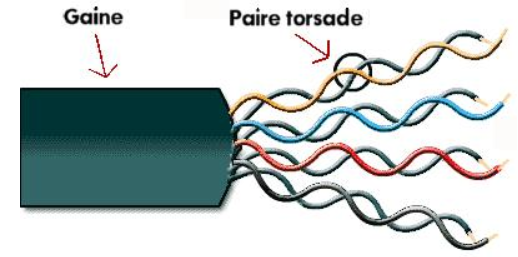
Voici aux figures suivantes, dans l'ordre de haut en bas, le câble équipé d'une prise BNC, le té BNC et le bouchon BNC.



Heureusement pour nous, le réseau a évolué, et Zorro la paire torsadée est arrivée !

**La paire torsadée**

Le câble à paires torsadées n'est plus un câble coaxial. Il n'y a plus un unique fil dans le câble mais huit ! De quoi faire passer de l'information dans tous les sens !  
Le câble à paires torsadées est donc composé de huit fils, torsadés deux à deux par paires, d'où le génie des chercheurs quand ils ont trouvé son nom, la paire torsadée ! Vous pouvez en voir un exemple sur la figure suivante.



Aujourd'hui, dans la plupart des réseaux, nous utilisons 2 paires, soit 4 fils, car nous utilisons une paire pour envoyer les données, et une paire pour les recevoir. Nous n'utilisons donc que 4 fils sur 8.

Est-ce que la paire torsadée a un nom compliqué comme le 10B2 ?

Oui, on l'appelle aussi le 10BT, ou 100BT ou 1000BT, selon le débit utilisé (10 Mbps, 100 Mbps, 1000 Mbps) le T étant là pour « torsadé », ou *twisted* en anglais.

Le câble coaxial n'est plus utilisé, mais qu'en est-il de la paire torsadée ?

Eh bien on l'utilise partout, dans 90 % des cas ! C'est la *number one* de la connexion

Comment branche-t-on les machines avec ?

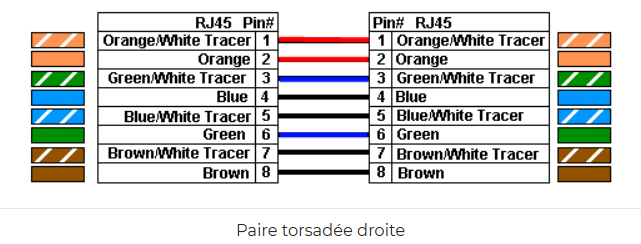
On les branche à l'aide de prises RJ45.

Et ne confondez pas le câble à paires torsadées avec les prises de ce câble, RJ45 ! Ne me parlez donc pas de câble RJ45, cela n'existe pas !

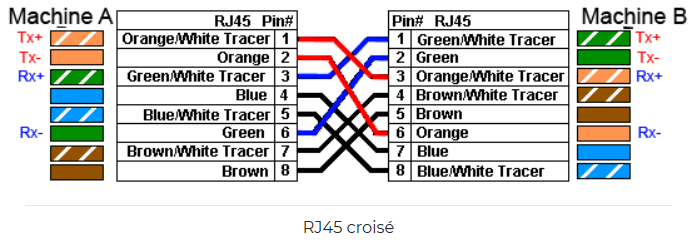
Voici à la figure suivante une prise RJ45. On peut voir les 8 petits connecteurs en cuivre qui sont reliés aux 8 fils.



Il faut utiliser des fils spécifiques, qui sont les fils 1, 2, 3 et 6. Voici en figure suivante le branchement d'un câble et les fils utilisés (avec les couleurs).



Pour pouvoir relier la transmission de la machine A avec la réception de la machine B, il faudrait que les fils 1 et 2 soient en relation avec les fils 3 et 6... Ce qui reviendrait à *croiser* les fils... Eh bien voilà, nous venons d'inventer le câble croisé !



**Pourtant je connecte mon ordinateur sur ma box et j'utilise un câble droit !**

Il peut y avoir deux raisons à cela :

* la prise femelle sur la box a déjà ses connexions transmission et réception inversées ;
* les prises femelles de ma box et de mon ordinateur sont capables de s'adapter et d'inverser les connexions de transmission et réception si besoin.

Il y a une règle simple, mais pas toujours facile à comprendre : **je dois utiliser un câble croisé pour connecter deux matériels de même type**.

Pour pouvoir relier plusieurs machines entre elles sur un réseau, il faut utiliser un matériel de connexion. Et pour la couche 1, il s'agit du *hub* (ou concentrateur en français). Le hub est une machine composée de plusieurs prises RJ45 femelles et qui a pour rôle de relier les machines entre elles (voir la figure suivante).



Le hub a un fonctionnement particulier. Imaginez qu'il y ait 5 machines branchées au hub, les machines A, B, C, D et E. Si A veut parler à C, elle va envoyer l'information au hub. Mais lui ne sait pas lire ! Il va donc envoyer l'information à toutes les machines en se disant qu'il y en aura bien une dans le tas qui sera la bonne !  
Les machines B, D et E vont voir que l'information n'est pas pour elles et vont la jeter, alors que la machine C va pouvoir la lire ! (*on voit tout de suite qu'un hub n'est pas top pour la confidentialité des données...*).

**La fibre optique** est amenée à remplacer la paire torsadée, notamment en raison des débits qu'elle peut offrir. Mais ce n'est pas pour tout de suite...

**La fibre optique**

Avec la fibre optique, nous transportons des 0 et des 1, non plus avec de l'électricité mais **avec de la lumière** !  
Ce sera en gros, allumé, éteint, allumé, éteint... On envoie de la lumière dans le fil, et elle ressort quelques mètres/kilomètres plus loin.

Il existe aussi aujourd'hui du câblage virtuel, j'ai nommé le **wifi** !

## La topologie réseau

**Les 3 topologies**

En réseau, la topologie est **la manière selon laquelle on branche les machines entre elles**.

Il y a trois topologies principales :

* la topologie en bus ;
* la topologie en anneau ;
* la topologie en étoile.

Celles en bus et en anneau sont voués à disparaître.

**Caractéristiques de l'étoile**



En étoile, toutes les communications passent par le point central.  
On lui envoie l'information avec le nom du destinataire, et le point central aiguille l'information vers la bonne machine.

## Le CSMA/CD

Dans une topologie en bus, il n'y a qu'un câble pour tout le monde, donc une seule machine peut parler à un instant t. Si deux machines parlent en même temps, il se produit **une collision**.

L’objectif du CSMA/CD est de **limiter le nombre de collisions en organisant le droit à la parole**. L'idée est de mettre en place une règle qui permettrait de n'avoir presque plus de collisions.

# Faire communiquer les machines entre elles, la couche 2

## La couche 2 et ses rôles

Comme nous l'avons vu dans un chapitre précédent, la couche 2 se nomme la couche liaison, ou plus précisément, **liaison de données**.

Le rôle donné à la couche 2 est de **connecter des machines sur un réseau local**.

La couche 2 possède un autre rôle important qui est **la détection des erreurs de transmission**. J'ai bien dit *détection*, et non pas *correction*, la différence est importante, car la couche 2 verra les erreurs, et fermera les yeux sur celles-ci.

Un identifiant, l’adresse MAC

L'adresse MAC est donc l'adresse d'une carte réseau.

Codage de l’adresse MAC

L'adresse MAC est codée sur 6 octets.

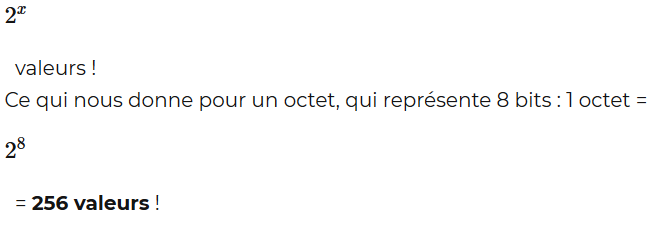
Un octet est une unité informatique indiquant une quantité de données.

Un disque 40 Go fera 40 gigaoctets, soit 40 000 000 000 octets !

Un octet représente 8 bits. Un bit est une valeur binaire.

Une valeur binaire peut être soit 0, soit 1. Un bit peut donc coder deux valeurs, deux bits peuvent coder quatre valeurs, trois bits 8 valeurs, etc. Dans l'exemple de deux bits, chacun d'eux peut prendre les valeurs 0 ou 1 ; quand on les couple on peut donc prendre les valeurs : 00, 01, 10, 11  
Ceci donne bien 4 valeurs différentes.

En fait, on en déduit que x bits peuvent coder



Un octet est donc compris entre 0 et... 255 (puisqu'on démarre à 0)  
Notre adresse MAC est codée sur 48 bits. Combien cela représente-t-il d'octets et de valeurs possibles (en puissances de 2) ?

1 octet = 8 bits, donc 48 bits = 48/8 octets = 6 octets.  
L'adresse MAC est codée sur 6 octets.

Vu que l'adresse MAC est codée sur 48 bits, elle peut prendre donc :



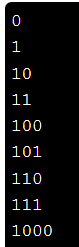
Soit... 281 474 976 710 656 valeurs ! Soit plus de 280 mille milliards d'adresses MAC possibles !

Chaque adresse MAC est unique.

**Rappel sur le binaire**

Le binaire est un système de numération en base 2. Globalement, cela veut dire qu'on ne peut compter qu'avec 1 et 0, contrairement au système de numération décimal que nous avons l'habitude d'utiliser dans lequel on se sert des chiffres de 0 à 9.

Si je compte en binaire, cela donne le résultat suivant :



Ce qui est équivalent en décimal à :



On compte en binaire car les informations électriques passaient sous la forme de 0 V ou 5 V, soit deux états différents 0 ou 1.

**Comment calculer en binaire ?**

Il y a plusieurs façons de faire, je vais vous en présenter une qui est *relativement* facile à utiliser.

tout nombre décimal peut s'écrire comme une somme de puissances de 2.

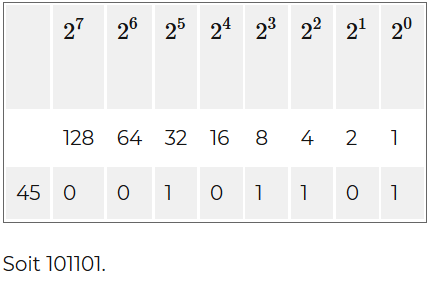
Prenons un exemple avec le nombre 45. Il peut s'écrire :  
45 = 32 + 8 + 4 + 1

=(1×25)+(0×24)+(1×23)+(1×22)+(0×21)+(1×20)

On peut donc écrire 45 en binaire : 101101

Tout nombre décimal peut s'écrire comme une somme de puissances de 2.  
On peut donc faire un tableau de puissances de 2 qui nous aidera à faire nos calculs :

Donc pour le nombre 45 ça donne :



Ce que nous allons faire pour un calcul, c'est de regarder si la puissance de 2 la plus élevée peut être contenue dans notre nombre, et recommencer avec la puissance de 2 suivante.

Pour notre exemple, est-ce que 128 peut être contenu dans 45 ? Non, je mets 0 dans la colonne 128. On passe à la puissance de 2 suivante :  
Est-ce que 64 peut être contenu dans 45 ? Non, je mets 0 dans la colonne 64. Est-ce que 32 peut être contenu dans 45 ? Oui ! Je mets 1 dans la colonne 32 ET j'ôte 32 à 45. Donc 45 - 32 = 13

Je continue maintenant avec ce nouveau chiffre. Est-ce que 16 peut être contenu dans 13 ?   
Non, je mets 0 dans la colonne 16.

Est-ce que 8 peut être contenu dans 13 ? Oui ! Je mets 1 dans la colonne 8 ET j'ôte 8 à 13.  
13 - 8 = 5

Est-ce que 4 peut être contenu dans 5 ? Oui ! Je mets 1 dans la colonne 4 ET j'ôte 4 à 5.  
5 - 4 = 1

Est-ce que 2 peut être contenu dans 1 ? Non, je mets 0 dans la colonne 2.

Est-ce que 1 peut être contenu dans 1 ? Oui ! Je mets 1 dans la colonne 1 ET j'ôte 1 à 1.  
1 - 1 = 0 donc j'ai fini !

Comment écrire 109 en binaire ?

109 en décimal s'écrit 1101101 en binaire.

L'adresse MAC s'écrit en hexadécimal...

0, 1, 2, 3, 4, 5, 6, 7, 8, 9... a, b, c, d, e et f !

Par exemple, **10** en hexadécimal s'écrit **a**.  **11** s'écrit **b**, etc.

Notre adresse MAC s'écrira donc en hexadécimal.  
En voici une pour l'exemple : 00:23:5e:bf:45:6a

Récapitulons :

* L'adresse MAC est l'adresse d'une carte réseau.
* Elle est unique au monde pour chaque carte.
* Elle est codée sur 6 octets (soit 48 bits).

Grâce à l'adresse MAC, je suis donc capable d'envoyer des informations à la carte réseau d'une machine !

**Une adresse MAC spéciale**

Parmi les adresses MAC, il y en a une particulière, c'est l'adresse dans laquelle tous les bits sont à 1, ce qui donne ff:ff:ff:ff:ff:ff.  
Cette adresse est appelée **l'adresse de broadcast**.

L'adresse de broadcast est une adresse universelle qui **identifie n'importe quelle carte réseau**.  
Elle me permet ainsi d'envoyer un message à toutes les cartes réseaux des machines présentes sur mon réseau, en une seule fois.

Toute machine qui reçoit une trame qui a, comme adresse MAC de destination, l'adresse de broadcast, considère que la trame lui est destinée.

## Un protocole, Ethernet

Nous allons devoir définir un langage pour communiquer entre machines. Ce langage permettra de définir le format des messages que les ordinateurs vont s'échanger. Et le gagnant est... **Ethernet** ! En réseau, on traduit langage par **protocole**

**À quoi sert un protocole ?**

L'objectif des réseaux est de pouvoir s'échanger des informations. Étant donné que nous discutons entre des machines très différentes, qui elles-mêmes ont des systèmes d'exploitation très différents (Windows, Mac OS, Linux, etc.), nous devons créer un langage de communication commun pour se comprendre. C'est le protocole.

Dans notre message, nous allons au moins devoir envoyer :

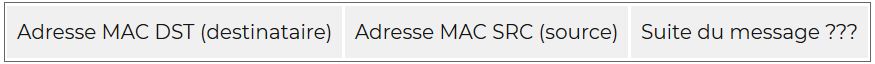
* l'adresse de l'émetteur ;
* l'adresse du destinataire ;
* le message proprement dit.

Le protocole va donc définir le format des messages envoyés sur le réseau.

Plus exactement, nous allons appeler ce message, **une trame**.

La trame est le message envoyé sur le réseau, en couche 2

**Format d'une trame Ethernet**



En descendant les couches du modèle OSI nous sommes donc passés par la couche 3 **avant** de passer par la couche 2. La couche 3 peut donc indiquer à la couche 2 quel est le protocole qui a été utilisé en couche 3.

Et c'est utile, car à l'arrivée, quand la couche 2 de la machine réceptrice reçoit les données, qu'elle voit que l'adresse MAC de destination est bien la sienne, elle doit envoyer les informations à la couche 3, et donc au bon protocole de couche 3.

Il est donc nécessaire d'indiquer dans la trame quel protocole de couche 3 a été utilisé quand le message a été envoyé et qu'il a traversé les couches du modèle OSI de haut en bas.

Notre trame devient donc :



On y ajoute les infos à envoyer et le CRC

**Qu'est-ce que le CRC ?**

Le CRC est une valeur mathématique qui est représentative des données envoyées.

En gros cela veut dire que c'est un nombre qui sera différent pour chaque message.  
Imaginons qu'une machine A envoie un message à une machine B.

* Lors de l'envoi, A calcule le CRC (une valeur X) et le met à la fin de la trame.
* B reçoit le message et fait le même calcul que A avec la trame reçue (une valeur Y).
* B compare la valeur qu'elle a calculée (Y) avec la valeur que A avait calculée et mise à la fin de la trame (X).

Si elles sont égales, bingo ! La trame envoyée par A est bien identique à la trame reçue par B.

Si elles sont différentes, gloups ! Il y a eu une erreur lors de la transmission. La trame reçue par B n'est apparemment pas la même que celle envoyée par A. Il y a eu un problème quelque part, mais nous l'avons détecté !

**La trame complète**

Nous avons maintenant tous les éléments de la trame et avons donc **la trame complète** :



**Quelle taille pour la trame ?**

Il y a des éléments qui ne varient jamais d'une trame à l'autre. L'ensemble de ces éléments est appelé en-tête ou, dans le cas de la couche 2, **en-tête Ethernet**. Ils sont indiqués **en gras dans la figure au-dessus**.

Cet en-tête ne variant pas, nous pouvons définir sa taille :

* les adresses MAC font chacune 6 octets ;
* le protocole de couche 3 est codé sur 2 octets ;
* le CRC est codé sur 4 octets.

Ce qui donne un total de **18 octets pour l'en-tête Ethernet**.

La taille minimale pour une trame Ethernet est de 64 octets et la taille maximale est de 1518 octets.

Au passage, on se rend compte que si on enlève les 18 octets d'en-tête à la taille maximale, nous tombons sur un chiffre rond de 1500 octets de données pour les données à envoyer !

Récapitulons un peu, en observant un échange de données entre deux machines A et B.

* Une application sur la machine A veut envoyer des données à une autre application sur une machine B.
* Le message parcourt les couches du modèle OSI de haut en bas.
* La couche 3 indique à la couche 2 quel protocole a été utilisé.
* La couche 2 peut alors former la trame et l'envoyer sur le réseau.
* La machine B reçoit la trame et regarde l'adresse MAC de destination.
* C'est elle ! elle lit donc la suite de la trame.
* Grâce à l'information sur le protocole de couche 3 utilisé, elle peut envoyer les données correctement à la couche 3.
* Le message remonte les couches du modèle OSI et arrive à l'application sur la machine B.
* on sait maintenant que le rôle principal de la couche 2 est de **connecter les machines sur un réseau local** ;
* elle permet aussi de détecter les erreurs ;
* le protocole utilisé en couche 2 est **le protocole Ethernet** ;
* les cartes réseau ont une adresse qui est l'adresse MAC ;
* **l'adresse MAC** est codée sur 6 octets, soit 48 bits ;
* chaque adresse MAC est **unique au monde** ;
* il existe une adresse particulière, l'adresse de broadcast qui permet de parler à tout le monde, ff:ff:ff:ff:ff:ff ;

Vous connaissez les principes de la couche 2 et nous allons maintenant étudier en détail l'équipement qui permet de connecter les machines entre elles, le switch.

# Le matériel de couche 2, le commutateur ou switch

### Un matériel, le commutateur

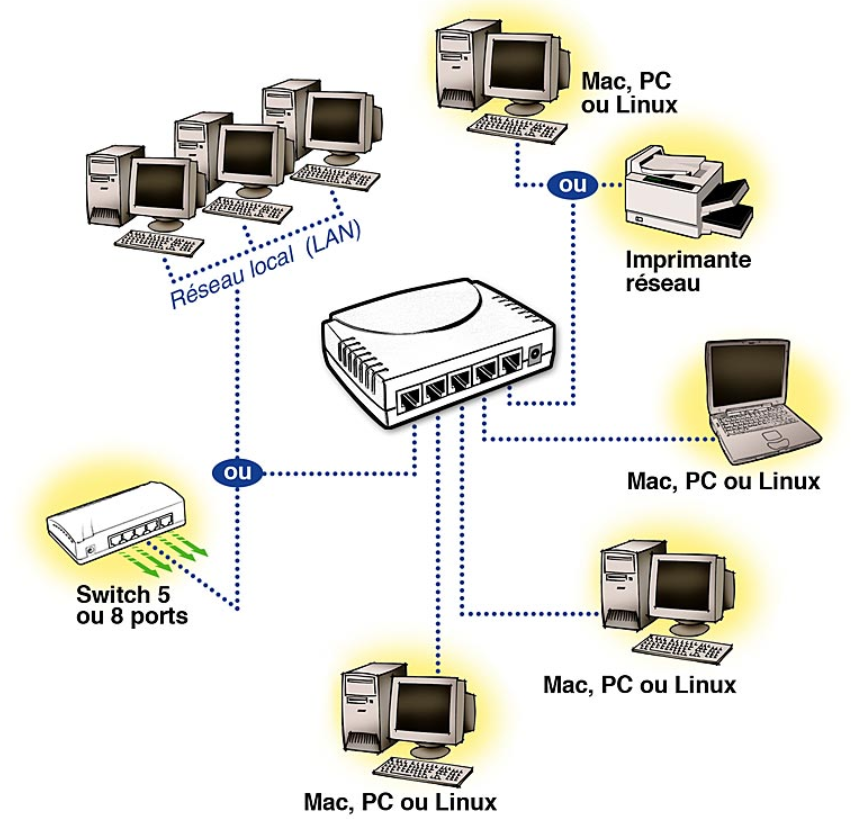
Le commutateur est un matériel qui va pouvoir nous permettre de relier plusieurs machines entre elles.  
On l'appelle aussi switch en anglais. Ce terme étant très souvent utilisé en français, nous pourrons donc utiliser les deux.

Vous entendrez parfois parler de **pont ou** bridge en anglais. Un pont n'est rien d'autre qu'un **switch avec seulement deux ports**. Donc si vous connaissez le switch, vous connaissez le pont !

Un commutateur est un boîtier sur lequel sont présentes plusieurs prises RJ45 femelles permettant de brancher dessus des machines à l'aide de câbles à paires torsadées.



Nous allons donc brancher nos machines au switch, voire d'autres switchs à notre switch

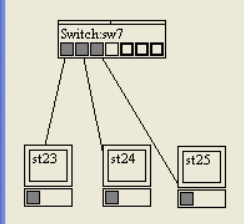


Si tout le monde est connecté ensemble, comment le switch sait à qui envoyer la trame ?

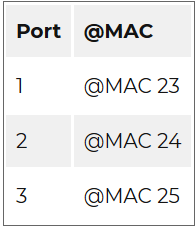
**L'aiguillage des trames**

Pour envoyer la trame vers la bonne machine, le switch se sert de l'adresse MAC destination contenue dans l'en-tête de la trame.  
Il contient en fait une table qui fait l'association entre **un port du switch** (une prise RJ45 femelle) et **une adresse MAC**. Cette table est appelée **la table CAM**.

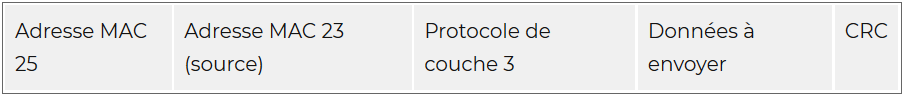
Prenons un exemple, avec le schéma de la figure suivante.



La table CAM de notre switch sera la suivante :



Quand la machine 23 voudra envoyer une trame à la machine 25, le switch lira l'adresse destination et saura alors vers quel port renvoyer la trame :



Le switch va donc envoyer la trame sur le port 3, et elle arrivera bien à la machine 25 qui est branchée sur ce port, et à elle seule !

Le switch aiguille donc les trames **grâce à l'adresse MAC de destination** située dans l'en-tête et à sa table CAM qui lui dit sur quel port renvoyer cette trame.

Donc un switch sait aiguiller une trame vers la bonne machine.

Comment cette table CAM est fabriquée ? Si je branche une nouvelle machine, comment le switch la connaît ?

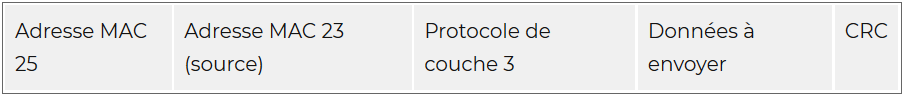
**Mise à jour de la table CAM**

La table CAM du switch va être fabriquée de façon **dynamique**. Cela veut dire que le switch va apprendre, au fur et à mesure qu'il voit passer des trames, quelle machine est branchée à quel port.

Prenons le cas d’un swich ayant une table CAM vide

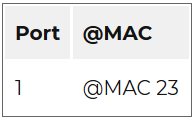


Imaginons maintenant que la machine 23 envoie une trame à la machine 25.



* La trame arrive au switch.
* Il lit l'adresse MAC source et voit l'adresse MAC de la machine 23.
* Vu que la trame vient du port 1, il met en relation le port 1 et l'adresse MAC de la machine 23 dans sa table CAM.
* Il met à jour sa table CAM.

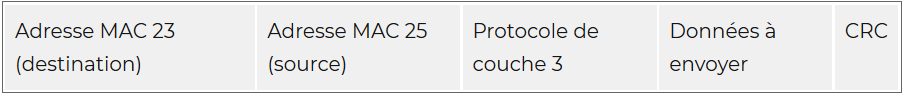
Table CAM mise à jour



Par contre, l'adresse MAC destination n'est pas présente dans sa table CAM, il ne sait donc pas où envoyer la trame. Pour être sûr que la machine destination va recevoir la trame, il lui suffit de l'envoyer **à tout le monde**, donc de renvoyer la trame sur tous les ports actifs du switch !

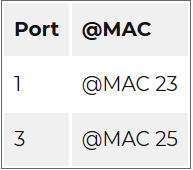
Attention, **ceci n'est pas un broadcast**, car l'adresse de destination dans la trame est l'adresse MAC de la machine 25. La trame est envoyée à tout le monde, mais pas en broadcast.

La machine 25 va donc recevoir la trame et va pouvoir répondre à la machine 23. Elle va donc envoyer une trame à la machine 23.



*Trame de réponse envoyée de 25 à 23*

* La trame arrive au switch.
* Il lit l'adresse MAC source et voit l'adresse MAC de la machine 25.
* Vu que la trame vient du port 3, il met en relation le port 3 et l'adresse MAC de la machine 25 dans sa table CAM.
* Il met à jour sa table CAM.



**Le TTL de la table CAM**

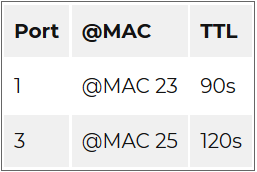
En réseau, nous allons très, très souvent parler de TTL.

Le TTL veut dire Time To Live en anglais, soit Durée De Vie. Il représente donc une durée.

Le principe est de considérer qu'une donnée est valable pendant un certain temps, mais qu'au-delà de ce temps, elle ne l'est plus.

Pour une information dans la table CAM, c'est pareil. On va considérer que cette information est valable un certain temps, mais une fois ce temps dépassé, on enlèvera l'information de la table CAM. Ainsi la table CAM sera mise à jour régulièrement et les données les plus anciennes seront effacées.

Nous ajoutons donc la colonne pour le TTL :



*Table CAM avec le TTL mis à jour*  
Ainsi, la table CAM du switch se remplira ou se mettra à jour **après chaque réception d'une trame**, et elle se videra quand elle n'aura pas reçu de trame depuis longtemps.

**Le switch a-t-il une adresse MAC ?**

Non car personne n'a besoin de parler avec le switch, donc il ne nécessite pas d'adresse MAC.

Cependant, certains switchs sont dits « administrables », ce qui veut dire que l'on peut se connecter dessus pour les configurer. Et dans ce cas, ils ont une adresse MAC pour être identifiés sur le réseau.

Tu peux avoir plusieurs machines sur le même port, en fait cela signifie que tu as branché un switch sur le port de mon switch. Et donc toutes les adresses MAC des machines connectées à ce switch seront susceptibles d'apparaître sur le port du premier switch.

**Trucs et astuces (de vilains...)**

Connaissant maintenant le fonctionnement d'un switch, comment pensez-vous qu'on puisse faire pour gêner son fonctionnement s'il nous en prend l'envie ? Il y a plusieurs façons de le faire.

**Méthode 1, saturation par envoi massif intelligent.**  
Si l'on envoie des tonnes de trames vers des adresses MAC inexistantes, que se passe-t-il ?

Le switch ne sachant pas vers quel port les envoyer, il va les envoyer vers tous les ports actifs... et va donc vite saturer !

**Méthode 2, saturation de la table CAM.**  
Si l'on envoie des tonnes de trames en utilisant à chaque fois une adresse MAC de source différente, que se passe-t-il ?

La table CAM du switch va se remplir progressivement. Plus elle sera remplie, plus sa lecture par le switch sera longue, et plus cela induira des temps de latence importants... jusqu'à provoquer l'écroulement du switch. Quand il sera saturé et n'aura plus le temps de lire sa table CAM, il enverra directement les trames sur tous les ports. Ceci permettrait à un pirate de voir tout le trafic du switch.

## La révolution du commutateur

**Qu'a apporté la commutation ?**

A priori, on peut se dire que par rapport à un hub, un commutateur permet d'isoler les conversations. Ceci dit, les conséquences de l'isolation des communications sont énormes !

Il n'y a pas de collisions sur un switch ?

Non ou alors, c'est qu'on l'a configuré pour qu'il y en ait.

En paire torsadée, nous **utilisons des fils différents pour la transmission et la réception**, donc deux messages envoyés en même temps vont se croiser, mais sur des fils différents !

Imaginons avec le schéma précédent que les machines 23 et 25 parlent en même temps à la machine 24.  
Dans ce cas, les deux messages vont arriver en même temps sur la paire de réception de la machine 24.

Ce cas a été prévu et les switchs imaginés en conséquence.  
En fait, **le switch possède une mémoire** dans laquelle il peut stocker une ou plusieurs trames quand il les reçoit. Il ne renvoie cette trame que si la paire de transmission de la machine à qui elle est destinée est libre. Ainsi, quand il a deux trames à envoyer sur la même paire de réception, il envoie la première, puis la seconde. Il n'y a alors pas de collision.

Et donc s'il n'y a plus de collisions, ce n'est plus la peine de faire du CSMA/CD ?

Non plus ! Fini, exit le CSMA/CD ! Avant, les machines devaient écouter avant d'envoyer une trame pour vérifier que le réseau était libre, c'était le CSMA/CD. Maintenant, dès qu'une machine veut envoyer une trame, elle l'envoie, sans se soucier de savoir si quelqu'un d'autre est en train de parler, car elle est sûre et certaine que cela ne provoquera pas de collision !

Le fait d'abandonner le CSMA/CD porte un nom. On dit que la carte réseau fonctionne en **full duplex**.  
À l'inverse, quand on fait du CSMA/CD sur un hub ou un câble coaxial, la carte réseau fonctionne en **half duplex**.

Le full duplex, c'est super ! Encore faut-il qu'il soit utilisé à bon escient, et ce n'est pas toujours le cas. Il peut parfois faire des ravages...  
Imaginez qu'on branche 10 machines sur un hub. Nous sommes sur un hub, donc sur une topologie en bus, donc les machines doivent être en half duplex et faire du CSMA/CD.

Que se passe-t-il si la carte réseau de l'une d'entre elles est configurée en full duplex ?

Eh bien cela est très, très gênant. Tout simplement, car les neuf autres machines attendent que le hub soit libre avant de pouvoir parler, et si jamais quelqu'un parle en même temps qu'elles, elles considèrent qu'il y a une collision.  
Alors que notre machine en full duplex ne se soucie de rien, parle quand elle veut, ne détecte aucune des collisions qui se produisent. Bref, c'est la catastrophe…

Pire encore, si cette machine est en train de télécharger un gros fichier, elle parle en permanence et empêche toutes les autres de parler. Le réseau est alors inutilisable pour nos neuf machines !

Nous voyons donc que toute machine connectée à un hub doit automatiquement avoir sa carte réseau configurée en half duplex.

Aujourd’hui les cartes réseau **sont intelligentes et capables de déterminer seules le duplex à utiliser**. Ainsi, quand une carte réseau est branchée, elle est capable de déterminer si elle doit fonctionner en full duplex ou en half duplex. Branchée à un hub, elle se mettra en half duplex ; branchée à un switch, elle se mettra en full duplex.

## Pour aller plus loin, les VLANs

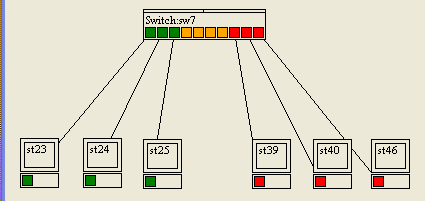
**Qu'est-ce qu'un VLAN ?**

Un VLAN est un LAN virtuel (ou virtual LAN en anglais).  
Sachant qu'un LAN est un réseau local (ou Local Area Network en anglais) un VLAN est donc un **réseau local virtuel**.

Un VLAN est la capacité de séparer des ports d'un switch dans des réseaux différents.

Cela revient à séparer certains ports d'un switch. Ils ne pourront donc plus communiquer ensemble.

L'idée du VLAN est de **couper notre switch en plusieurs morceaux**, comme si l'on avait plusieurs switchs.



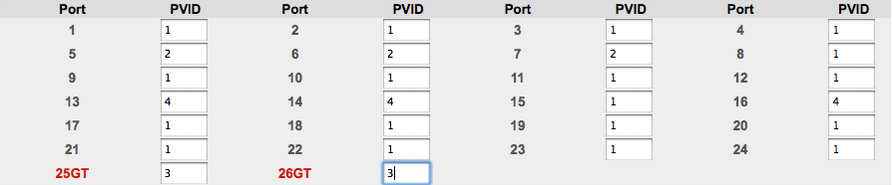
Nous voyons ici en vert et en rouge les deux VLANs. Ainsi, les machines connectées aux ports appartenant au VLAN vert ne peuvent communiquer qu'avec le VLAN vert. Et de même pour les machines connectées aux ports appartenant au VLAN rouge. Par contre, il est impossible pour une machine connectée au VLAN vert de communiquer avec une machine connectée au VLAN rouge. C'est comme si l'on avait séparé le switch en deux petits switchs, chacun ayant sa propre table CAM.

**Quel est l'intérêt des VLANs ?**

Dans l'exemple que nous avons choisi, l'intérêt n'est pas flagrant, mais imaginons que nous ayons à gérer une école, avec une administration, 100 enseignants et 1000 élèves. Nous avons alors plusieurs switchs répartis dans l'école. Des gros switchs de 256 ports ! (on appelle cela souvent des châssis.)  
Il est intéressant de pouvoir segmenter ces switchs pour séparer les trois populations, pour que les élèves n'aient pas accès au réseau administratif ou à celui des enseignants, et que les enseignants n'aient pas accès au réseau administratif (pour changer leur fiche de paye par exemple). Plutôt que d'acheter 25 petits switchs de 48 ports, on en achète 5 gros de 256 ports.

En plus de la sécurité offerte par la séparation des réseaux, cela apporte de la facilité de configuration. Si je veux qu'un port passe d'un VLAN à un autre, il me suffit de le configurer sur le switch.

Je peux faire tout cela sans bouger de mon bureau d'administrateur réseau à travers une interface web d'administration du switch, comme vous pouvez le voir sur la figure suivante.



On voit ici que chaque port peut être positionné dans un VLAN donné.  
Ici le port 1 est dans le VLAN 1 alors que le port 5 est dans le VLAN 2. Les machines connectées sur ces ports ne pourront pas communiquer ensemble.

Un VLAN permet donc d'isoler certains ports d'un switch par rapport aux autres, comme si l'on coupait le switch en deux.

Dans les propriétés avancés de la carte réseau, tu peux modifier l'adresse MAC.

Dans certains cas, il peut être intéressant de modifier son adresse MAC, notamment si une authentification wifi se base sur l'adresse MAC d'une machine...

Attention ne jamais faire de boucles sur, ou entre, des switchs !

# La couche 3

## La couche 3 ses rôles

La couche 3 est la couche **réseau**.

Le rôle de la couche 3 est donc d'**interconnecter les réseaux**.

La couche 3 va donc me permettre de joindre n'importe quel réseau sur Internet, en passant à travers d'autres réseaux. Ma connexion à une machine sur un autre réseau se fera à travers des réseaux, de proche en proche.

La commande traceroute sous Linux ou tracert sous Windows permet d'indiquer par quelles machines nous passons pour aller d'un point à un autre sur Internet.

## Un identifiant, l'adresse IP

L'adresse IP est en fait l'adresse **du réseau ET de la machine**.

Plus exactement, une partie de l'adresse représentera l'adresse du réseau, et l'autre partie l'adresse de la machine.

Une adresse IP **est codée sur 32 bits** soit 4 octets.

Cela donne par exemple : 192.168.0.1

**Le masque de sous-réseau**

C'est le masque qui va indiquer quelle est la partie réseau de l'adresse, et quelle est la partie machine.

**Définition : Les bits à 1 dans le masque représentent la partie réseau de l'adresse IP.**

On en déduit que les bits à 0 représentent la partie machine de l'adresse.

Prenons un exemple : on associe l'adresse IP 192.168.0.1 au masque 255.255.0.0.  
Écrivons maintenant ces deux adresses en binaire pour y voir plus clair :

255.255.0.0 -> 11111111.11111111.00000000.00000000  
192.168.0.1 -> 11000000.10101000.00000000.00000001

Donc la partie réseau de l'adresse est 192.168, et la partie machine est 0.1

L'exercice que nous venons de faire était très facile, car la coupure entre les deux parties de l'adresse se faisait entre deux octets. Or, il arrive très souvent que **la coupure se fasse en plein milieu d'un octet.**

## Le masque de sous-réseau et les difficultés associées...

**Calcul de la partie réseau et de la partie machine d'une adresse**

**La contiguïté des bits**

Dans un masque en binaire, il doit y avoir **les 1 à gauche et les 0 à droite**.  
On ne peut pas mélanger les 1 et les 0.

Par exemple, ce masque est **correct**: 11111111.11111000.00000000.00000000

.  
Mais celui-ci est **incorrect**: 11111111.11100011.00000000.00000000.

Ainsi, on retrouvera toujours les mêmes valeurs pour les octets d'un masque, qui sont les suivantes :

00000000 -> 0  
10000000 -> 128  
11000000 -> 192  
11100000 -> 224   
11110000 -> 240   
11111000 -> 248   
11111100 -> 252   
11111110 -> 254   
11111111 -> 255

Donc ce masque est **correct**: 255.255.128.0.  
Et ce masque est **incorrect**: 255.255.173.0.  
Et ce masque est encore **incorrect**: 255.128.255.0 (car il mélange des 0 et des 1).

Une plage d'adresse est l'ensemble des adresses définies par l'association d'une adresse et d'un masque, de la plus petite adresse à la plus grande.

**Calcul de plages d'adresses**

L'adresse 192.168.0.1 associée au masque 255.255.240.0.

Votre mission, si vous l'acceptez, est de trouver la première et la dernière adresse du réseau auquel appartient cette adresse.

Nous avons le masque et l'adresse :

255.255.240.0 -> 11111111.11111111.11110000.00000000  
192.168.0.1 -> 11000000.10101000.00000000.00000001

Toutes les machines appartenant à un même réseau ont un point commun : **tous les bits de leur partie réseau sont identiques** !

Donc pour notre calcul, on en déduit que toutes les machines appartenant à notre réseau vont avoir leur partie réseau égale à 11000000.10101000.0000.

Par contre, les bits de la partie machine de l'adresse vont pouvoir varier pour toutes les machines du réseau.

Dans ce réseau, les adresses des machines pourront prendre beaucoup de valeurs, selon que l'on met certains bits de la partie machine à 0 ou 1.  
Globalement, les adresses seront :

 11000000.10101000.00000000.00000000

-> 192.168.0.0  
 11000000.10101000.00000000.00000001 -> 192.168.0.1  
 11000000.10101000.00000000.00000010 -> 192.168.0.2  
 11000000.10101000.00000000.00000011 -> 192.168.0.3

...  
 11000000.10101000.00001111.11111110 -> 192.168.15.254  
 11000000.10101000.00001111.11111111

‌ -> 192.168.15.255

En faisant varier les bits de la partie machine de l'adresse, nous avons pu trouver toutes les adresses du réseau.

La première adresse du réseau est celle dont tous les bits de la partie machine sont à 0 ; la dernière adresse du réseau est celle dont tous les bits de la partie machine sont à 1.

**Nombre d'adresses dans un réseau**

Si jamais nous n'avions qu'un seul bit pour la partie machine, nous aurions deux possibilités sur ce bit, 0 ou 1. Si nous en avions deux, il y aurait 2²  adresses possibles, soit 4 adresses (00, 01, 10, 11) et ainsi de suite.  
Si nous avions 10 bits pour la partie machine, nous aurions 210  adresses possibles, soit 1024 adresses.

Or, vu que la partie machine est définie par le masque, **le nombre de machines disponibles dans un réseau est directement dépendant du masque !**

La relation est même encore plus explicite : nombre d'adresses dans un réseau = 2nombre de 0 danslemasque

Le masque s'écrit :  
255.255.240.0 -> 11111111.11111111.11110000.00000000

Dans lequel nous voyons douze 0 qui identifient la partie machine de l'adresse.  
Nombre d'adresses = 212  = 4096 adresses !

**Adresse de réseau, adresse de broadcast**

Parmi la plage d'adresses définie par une adresse IP et un masque, deux adresses sont particulières, la première et la dernière.

**La première adresse d'une plage est l'adresse du réseau lui-même.**  
Cette adresse ne pourra donc pas être utilisée pour une machine.

**La dernière adresse d'une plage est une adresse spéciale, l'adresse de broadcast.**  
Cette adresse ne peut pas non plus être utilisée pour une machine. Elle est en fait utilisée pour identifier toutes les machines de mon réseau.  
Quand nous envoyons un message à l'adresse de broadcast, ce message va être reçu par toutes les machines de notre réseau.

Donc dans un réseau ayant 16 adresses disponibles, seules 14 adresses seront utilisables par les machines du réseau, car la première et la dernière seront réservées pour le réseau et le broadcast. Et cela est vrai pour tout réseau. Pour chaque réseau, il y a deux adresses non utilisables pour les machines.

## Le masque mis en pratique

**Adresse de réseau, de machine ou de broadcast ?**

Le principe de l'exercice est simple.  
Je vais vous donner un couple adresse/masque, et vous devrez me dire si l'adresse est une adresse de réseau, de machine ou de broadcast.

Exemple :

192.168.0.15 -> 11000000.10101000.00000000.00001111  
255.255.255.240 -> 11111111.11111111.11111111.11110000

Je fixe la partie réseau dans l'adresse :  
 11000000.10101000.00000000.00001111

Et je fais varier les bits de la partie machine en mettant tout à 0, puis tout à 1.  
 11000000.10101000.00000000.00000000

-> 192.168.0.0  
 11000000.10101000.00000000.00001111

-> 192.168.0.15

Nous avons donc trouvé 192.168.0.0 comme adresse de réseau. Il reste donc que 4 bits pour la partie machine. Tous les bits de la partie machine sont à 1 donc 192.168.0.15 est l’adresse de broadcast.

**Des exemples plus complexes**

Alors allons-y !

Voici les données, à vous de faire l'exercice et de voir par la suite si vous avez la bonne réponse !

*Citation*

192.168.0.15/255.255.255.0  
192.168.1.0/255.255.255.0  
192.168.1.0/255.255.254.0  
10.8.65.29/255.255.255.224  
10.8.65.31/255.255.255.224  
10.0.0.255/255.255.254.0

Et voici les solutions plus bas

192.168.0.15/255.255.255.0  
Réseau allant de 192.168.0.0 à 192.168.0.255 -> Adresse de machine !

192.168.1.0/255.255.255.0  
Réseau allant de 192.168.1.0 à 192.168.1.255 -> Adresse de réseau !

192.168.1.0/255.255.254.0  
Réseau allant de 192.168.0.0 à 192.168.1.255 -> Adresse de machine !

10.8.65.29/255.255.255.224  
Réseau allant de 10.8.65.0 à 10.8.65.31 -> Adresse de machine !

10.8.65.31/255.255.255.224  
Réseau allant de 10.8.65.0 à 10.8.65.31 -> Adresse de broadcast !

10.0.0.255/255.255.254.0  
Réseau allant de 10.0.0.0 à 10.0.1.255 -> Adresse de machine !

Une adresse de broadcast est toujours impaire ; une adresse de réseau est toujours paire.

Trucs et astuces !

Après ces exercices, vous avez peut-être remarqué des informations intéressantes.

* Une adresse qui finit en 255 n'est pas obligatoirement une adresse de broadcast.
* Une adresse qui finit en 0 n'est pas obligatoirement une adresse de réseau.

**Des adresses particulières.**

**Les RFC**

Une RFC est un document qui propose et présente une technologie que l'on souhaite voir utiliser sur Internet.

Par exemple, il y a une RFC qui présente le protocole IP, [la RFC 791](https://www.ietf.org/rfc/rfc791.txt).

**La RFC 1918**

Cette RFC précise des plages d'adresses, soit des réseaux, qui ont une utilité particulière.  
En effet, ces plages d'adresses sont réservées pour une utilisation privée. Cela veut dire que si vous faites un réseau chez vous, ou dans une entreprise, il vous faudra obligatoirement utiliser ces adresses.

Les plages définies sont :

1. 10.0.0.0/255.0.0.0
2. 172.16.0.0/255.240.0.0
3. 192.168.0.0/255.255.0.0

Vu que ces adresses n'appartiennent à personne sur Internet, je serai sûr de pouvoir joindre n'importe quel site sur Internet.  
C'est aussi pour cela que, très souvent, les adresses qui sont données par les opérateurs sont dans ces plages.

# Découpage d'une plage d'adresses

A continuer ….

* + 1. **Découpage avec la méthode de base**

192.168.0.1/24. = IP avec un masque ayant 24 bits à 1 soit 255.255.255.0

on pourra donc écrire /20 au lieu de 255.255.240.0.

* + - 1. **Un premier découpage**

Prenons une entreprise possédant la plage 10.0.0.0/16. Nous allons essayer de découper cette plage.  
L'entreprise compte 1000 techniciens, 200 commerciaux et 20 directeurs. Il va donc falloir définir trois petites plages au sein de notre grande plage d'origine.

* + - * 1. **Vérification du nombre d'adresses**

Dans un premier temps, nous allons regarder si notre plage de départ contient assez d'adresses pour nos 1220 employés (1000 + 200 + 20).  
Le masque contient 16 bits à 1, donc 16 bits à 0 (puisqu'il contient au total 32 bits).  
Or, nous connaissons une formule qui nous permet de connaître le nombre d'adresses dans un réseau en fonction du nombre de bits à 0 dans le masque.

Rappel : **Nombre d'adresses dans un réseau =**2*Nombrede*0*dan**s**l**e**m**a**s**q**u**e*

Nous allons donc avoir dans ce réseau 216

 adresses, soit 65 536 adresses dans notre plage ! On en a largement plus que les 1220 nécessaires.

Par exemple pour les techniciens qui sont 1000, il me faudra un réseau avec au moins 1000 adresses.

Donc avec 2^10 on aura 1024 adresses.

Donc il faudrait dix 0 dans la partie machine et il restera donc 22 1 dans la partie réseau, le masque sera donc ainsi :

11111111 11111111 11111100 00000000, soit un masque de 255.255.252.0 ou /22

C'est 252 car on a déduit 3 à 255.

Nous pouvons faire le même calcul pour les 200 commerciaux :  
2^8 = 256, le masque pour les commerciaux sera donc 255.255.255.0.

Et enfin pour les 20 directeurs :  
2^5 = 32; le masque pour les directeurs sera donc 255.255.255.224.

* + - * 1. **Choix des plages d'adresses**

Nous avons donc la grande plage 10.0.0.0/16 de 65536 adresses à notre disposition, et nous souhaitons trouver une plage de 1024 adresses pour nos techniciens parmi ces 65536 adresses.  
Le choix le plus simple qui s'offre à nous est de commencer à l'adresse la plus basse, même si ce n'est pas le seul.

Donc nous choisissons de commencer notre plage d'adresses pour les techniciens à l'adresse 10.0.0.0.

Nous pouvons d'ores et déjà identifier le réseau des techniciens par le couple 10.0.0.0/255.255.252.0.

Donc au lieu de calculer en binaire toute l'adresse, nous n'allons travailler que sur le troisième octet   
Masque : 252 -> 11111100  
Adresse: 0 -> 00000000

Ainsi, d'après le masque, toutes les adresses des machines de mon réseau commenceront par 000000 sur le troisième octet.  
La dernière adresse sera donc celle où tous les bits de la partie machine sont à 1, soit 00000011 sur le troisième octet (3 en décimal), et 11111111 sur le quatrième octet qu'il ne faut pas oublier ! (255 en décimal)

La première adresse de la plage des techniciens est donc **10.0.0.255**.

La dernière adresse de la plage des techniciens est donc **10.0.3.255**.

Il nous faut maintenant en choisir une pour les 200 commerciaux.

Cependant, nous avons une contrainte supplémentaire sur le choix de notre plage d'adresses, c'est que les techniciens occupent déjà un certain nombre d'adresses de notre plage, de 10.0.0.0 à 10.0.3.255.  
Nous pouvons par exemple choisir de démarrer la plage d'adresses des commerciaux juste après celle des techniciens, en 10.0.4.0.  
Nous pouvons d'ores et déjà identifier le réseau des commerciaux par le couple 10.0.4.0/255.255.255.0. Comme pour les techniciens, il serait bien aussi de connaître la dernière adresse de cette plage.

La première adresse sera **10.0.4.0**.

La dernière adresse sera **10.0.4.255**.

Nous pouvons faire le même raisonnement pour les directeurs en commençant à 10.0.5.0.  
En associant le masque à cette adresse, nous trouvons comme dernière adresse **10.0.5.31**.

La première adresse sera **10.0.5.0**.

La dernière adresse sera **10.0.5.31**.

Nous avons bien découpé la grande plage d'adresses qui nous était donnée : 10.0.0.0/16 -> 10.0.0.0 à 10.0.255.255 en trois plages d'adresses plus petites :

* 10.0.0.0/22 -> 10.0.0.0 à 10.0.3.255 pour les techniciens ;
* 10.0.4.0/24 -> 10.0.4.0 à 10.0.4.255 pour les commerciaux ;
* 10.0.5.0/27 -> 10.0.5.0 à 10.0.5.31 pour les directeurs.

Une **méthode très simple** : faites toujours vos calculs **en prenant en premier les plages les plus grandes.**

* + 1. **Découpage avec la méthode magique**

La méthode magique est une méthode qui va nous permettre de calculer très facilement des plages d'adresses réseau, et bien plus encore !

* + - 1. **Le nombre magique**

Pour utiliser la méthode magique, nous allons devoir utiliser **le nombre magique**...

Le nombre magique est simplement un calcul fait à partir de l'octet significatif du masque. Il est égal à **256 - octet significatif**.

Par exemple, si l'on choisit le masque 255.224.0.0, on voit vite que l'octet significatif (celui où la séparation a lieu) est 224.  
Notre nombre magique vaut donc 256 - 224 = **32**.

* + - 1. **Que faire avec le nombre magique ?**

Il va nous permettre de calculer instantanément la première et la dernière adresse de notre plage. Pour cela, il va falloir écrire tous les multiples du nombre magique (jusqu'à 256 bien sûr).  
Allons-y pour les multiples de 32 : 0, 32, 64, 96, 128, 160, 192, 224, 256.

Et maintenant, nous allons simplement appliquer les deux règles suivantes :  
La première adresse du réseau sera le multiple du nombre magique, inférieur ou égal à l'octet correspondant dans l'adresse.  
La dernière adresse du réseau sera le multiple suivant, moins 1.

Un exemple sera plus parlant. On associe l'adresse 192.168.0.1 et le masque 255.224.0.0.  
Dans notre masque, l'octet significatif est le deuxième (255.**224**.0.0).  
Nous allons donc prendre le deuxième octet de notre adresse (192.**168**.0.1), soit 168.

La première adresse du réseau sera donc le multiple du nombre magique, inférieur ou égal à 168.  
En regardant la liste des multiples, on trouve très vite 160

La dernière adresse du réseau sera le multiple suivant, moins 1.  
Le multiple suivant est 192, auquel on enlève 1 pour trouver 191.

La première adresse de la plage est donc 192.160.0.0 et la dernière 192.191.255.255.  
On a ajouté les 0 pour la première et les 255 pour la dernière, car tous les bits qui suivent sont à 0 ou à 1, selon qu'on veut la première ou la dernière.

Prenons un nouvel exemple : 10.45.185.24/255.255.248.0

Le nombre magique vaut : 256 - 248 = 8 ; l'octet significatif du masque est le troisième, ce qui correspond à 185 dans l'adresse

Nous devons donc trouver le multiple de 8 inférieur ou égal à 185... Pas la peine de commencer à 0 !   
8 \* 10 = 80, on est en dessous de 185.  
8 \* 20 = 160, on est en dessous, mais on se rapproche.  
Commençons donc à 160 :  
160, 168, 176, 184, 192... STOP ! On est au-dessus de 185.

Le multiple inférieur ou égal est 184, celui du dessus moins un vaut 191. Ce qui nous donne pour la première adresse 10.45.184.0, et pour la dernière 10.45.191.255.

* + - 1. **Un exemple concret de découpage**

Vous avez en charge le réseau d'une petite entité d'une entreprise. L'administrateur général vous laisse à disposition le réseau : 192.168.160.0/255.255.224.0.

Vous avez dans votre entité trois catégories de personnel :

* 550 techniciens ;
* 130 commerciaux ;
* 10 directeurs.

l vous faut donc découper la plage d'origine en **trois sous-réseaux** pour chacune de ces populations.

* + - * 1. **Étape 1: Calcul de la plage d'origine**

Vous allez voir ici que la méthode magique est vraiment rapide par rapport à la méthode classique.

Allons-y !

1. Le nombre magique vaut: 256 - 224 = 32.
2. L'octet significatif de l'adresse vaut 160, qui est un multiple de 32 ! Ce sera donc la première adresse, la dernière étant 160 + 32 - 1 = 191.
3. La première adresse est 192.168.160.0 et la dernière est 192.168.191.255.
   * + - 1. **Étape 2: Calcul des masques**

Pour les techniciens, qui sont 550, le réseau devra contenir 1024 adresses (la puissance de 2 supérieure) soit

210

.  
Le masque contiendra donc 10 bits à 0, soit : 11111111.11111111.11111100.00000000.  
Soit en décimal : 255.255.252.0.

Nous pouvons faire pareil pour les commerciaux : 130 < 28

.  
Le masque est donc : 255.255.255.0.

Et pour les directeurs, nous trouvons : 10 < 24

.  
Le masque est donc : 255.255.255.240.

Nous avons les masques pour nos trois populations, il ne nous reste plus qu'à y associer des adresses pour avoir nos plages.

* + - * 1. **Étape 3: Calcul des plages**

C'est ici que la méthode magique va nous être utile, car elle permet facilement de trouver la première et la dernière adresse d'une plage.

Nous allons commencer par les techniciens. Notre plage de départ démarre en 192.168.160.0. Eh bien nous allons commencer la plage des techniciens à cette adresse, et allons trouver l'adresse de fin grâce au masque.

Calculons le nombre magique : 256 - 252 = 4.  
Le prochain multiple de 4 après 160 est 164 - 1 = 163.  
La dernière adresse pour les techniciens est donc 192.168.163.255.

Pour les commerciaux, nous allons donc démarrer à l'adresse juste après pour ne pas empiéter sur la plage des techniciens, soit 192.168.164.0.  
Nous allons nous passer du nombre magique pour les commerciaux, car la coupure se fait parfaitement entre deux octets sur le masque. L'adresse de fin est donc facilement calculée à 192.168.164.255.

Nous démarrons après pour les directeurs, à l'adresse 192.168.165.0. Le nombre magique vaut 256 - 240 = 16  
La dernière adresse est donc 192.168.165.15 !

* + - * 1. **Résultat**

Nous avons donc défini les trois plages :

* Tech: 192.168.160.0/255.255.252.0, soit les adresses allant de 192.168.160.0 à 192.168.163.255.
* Comm: 192.168.164.0/255.255.255.0, soit les adresses allant de 192.168.164.0 à 192.168.164.255.
* Dirs: 192.168.165.0/255.255.255.240, soit les adresses allant de 192.168.165.0 à 192.168.165.15.

Nous remarquons que pour le réseau des directeurs, l'adresse 192.168.165.15 est une adresse de broadcast même si elle ne finit pas par 255...

* + - 1. **Quand ça se complique**

Imaginons que nous ayons 120 secrétaires qui débarquent sur notre réseau  
Nous voulons leur créer une nouvelle plage, mais sans toucher aux réseaux existants.

Eh bien il suffit de prendre le multiple du nombre magique suivant !  
Nous allons commencer notre plage non pas en 192.168.165.16, mais en 192.168.165.128, et donc finir en 192.168.165.255.

Et là, nous avons bien défini un réseau d'au moins 120 adresses et qui n'empiète pas sur le réseau des directeurs !

Cependant, nous avons laissé un trou... Les adresses de 16 à 127 ne sont pas utilisées. C'est normal, et ce n'est pas grave de toute façon. Nous pourrons utiliser ces adresses pour des petits réseaux par la suite si nous le souhaitons.

Quand on place un réseau plus grand que le précédent dans une plage, il est nécessaire de sauter une certaine plage d'adresses et de laisser un "trou" dans la plage d'adresses.

Le principe est simple : vu que nous travaillons avec des réseaux dont la taille est un multiple de 2, un petit réseau pourra toujours démarrer sur un multiple d'un grand réseau.

Par exemple, tout multiple de 16 est un multiple de 8 :  
0, 16, 32, 48...  
0, 8, 16, 24, 32, 40, 48

On pourra donc toujours placer une petite plage d'adresses derrière une plage précédente plus grande. Et on pourra seulement parfois placer une grande plage derrière une petite, mais dans ce cas il faudra faire attention et bien utiliser la méthode magique.

* + - 1. **Exercices**

Ici encore, je vous conseille de ne pas négliger ces exercices. Faites-les **avant** de regarder les solutions.

* + - * 1. **Premier exemple**

Découpez la plage suivante en trois sous-réseaux : 10.47.192.0/255.255.240.0, avec les populations suivantes :

* 880 techniciens ;
* 400 commerciaux ;
* 60 directeurs.
  + - * 1. **Second exemple... le même que le premier !**

En fait l'énoncé est le même, mais l'on vous demande de commencer par les directeurs, puis les commerciaux, et enfin les techniciens.

La bonne nouvelle, c'est que les masques restent les mêmes !  
Techniciens : 255.255.252.0.  
Commerciaux : 255.255.254.0.  
Directeurs : 255.255.255.128.

**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 :

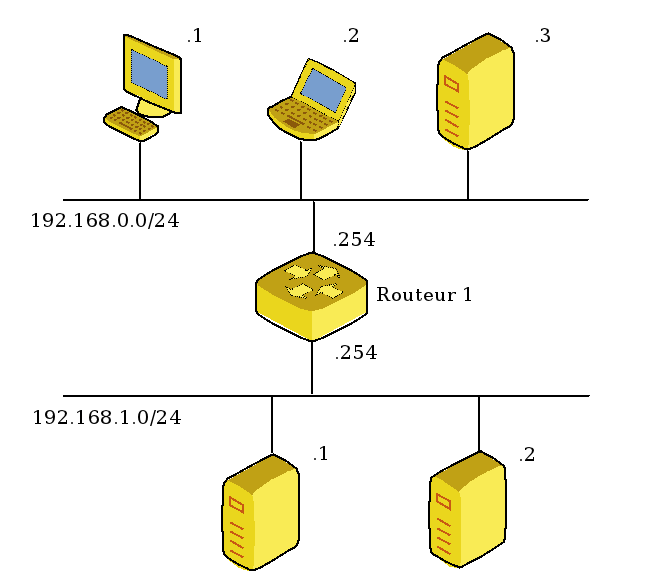
* un routeur est une machine possédant **plusieurs interfaces** ;
* chaque interface d'un routeur est connectée à un réseau, **le routeur relie ainsi plusieurs réseaux entre eux** ;
* toute machine ayant plusieurs interfaces peut jouer le rôle de routeur, même le vieux PC de mamie ;
* un routeur se différencie 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

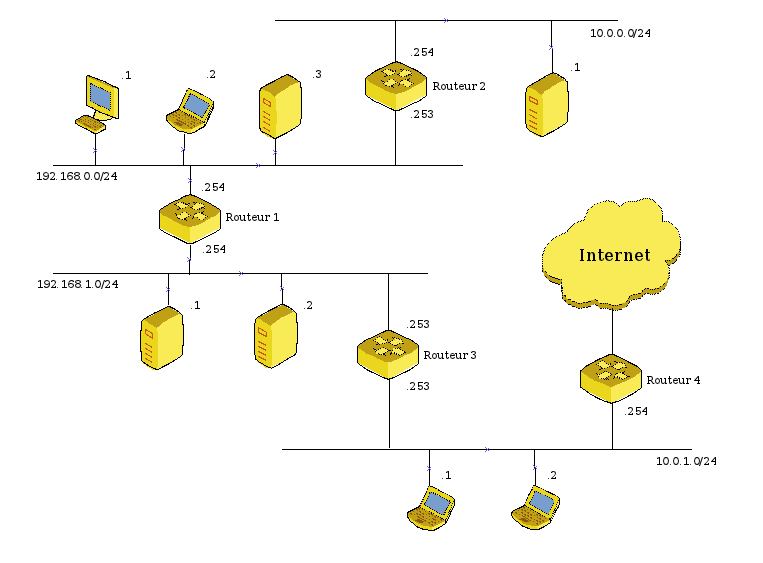


Table de routage du routeur 1

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

Extras

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.