Chapitre 4

Transmission de données dans un réseau

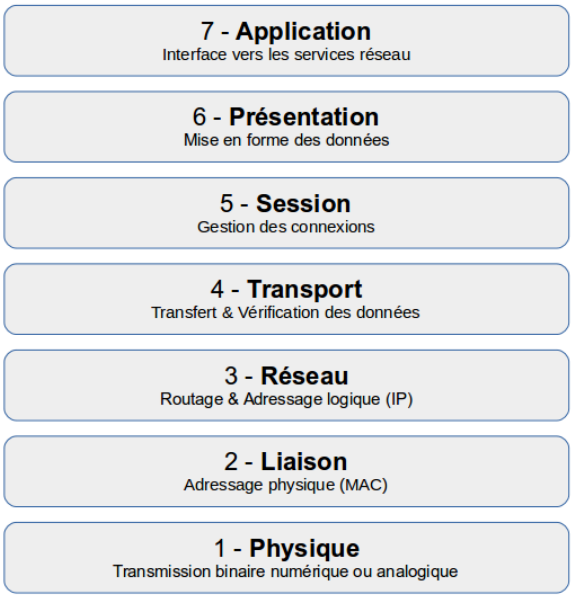
4.1 Comment communiquer ? Un peu de théorie...

Imaginons un groupe de 4 personnes qui échangent des informations. Avec un minimum de règles, chacun devrait pouvoir transmettre et recevoir des informations sans problème. Mais qu'en est-il d'un groupe de 100 personnes ? De plusieurs millions de personnes ?

Cette problématique, transposée aux machines (ordinateurs, smartphones, tablettes...), a été résolue dans les années 80. C'est le fonctionnement d'Internet tel que vous l'utilisez aujourd'hui. Mais comment cela fonctionne-t-il ?

4.1.1 Le modèle OSI

Le **modèle OSI** est apparu en 1984. C'est un modèle théorique qui permet de normaliser les communications entre les ordinateurs, autrement dit de fixer des règles et des protocoles de communication communs.
Selon ce modèle, toute communication peut se décomposer en 7 couches :



Pour utiliser ce modèle, deux règles doivent être respectées :

- **Chaque couche est indépendante** : que la liaison physique (couche 1) soit filaire ou non est totalement indépendant de l'utilisation d'adresses IPV4 ou IPV6 lors du routage (couche 3).
- **Chaque couche ne peut communiquer qu'avec une couche adjacente** : ainsi, toutes les couches du modèle sont parcourues.

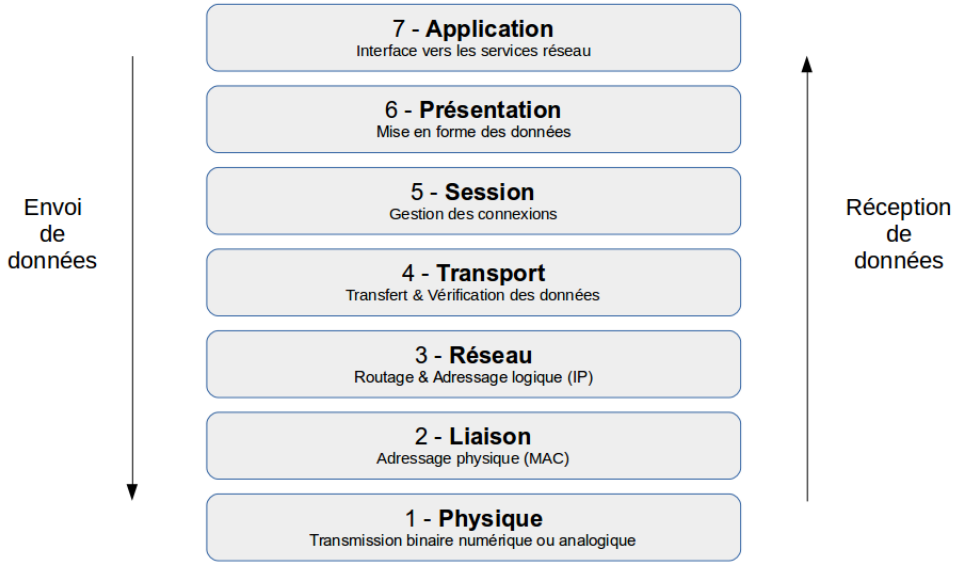
▷ ***À quoi correspondent ces différentes couches ?***

Le modèle OSI est un modèle relativement abstrait, dont le rôle de chaque couche peut être défini de la manière suivante :

1. La **couche physique** définit la façon dont les données sont physiquement converties en signaux numériques sur le média de communication (impulsions électriques, modulation de la lumière, ...).
2. La **couche liaison** définit l'interface avec la carte réseau et le partage du média de transmission.
3. La **couche réseau** permet de gérer l'adressage et le routage des données, c'est-à-dire leur acheminement via le réseau.
4. La **couche transport** est chargée du transport des données, de leur découpage en paquets et de la gestion des éventuelles erreurs de transmission.
5. La **couche session** définit l'ouverture et la destruction des sessions de communication entre les machines du réseau.
6. La **couche présentation** définit le format des données manipulées par le niveau applicatif (leur représentation, éventuellement leur compression et leur chiffrement) indépendamment du système.
7. La **couche application** assure l'interface avec les applications. Il s'agit donc du niveau le plus proche des utilisateurs, géré directement par les logiciels.

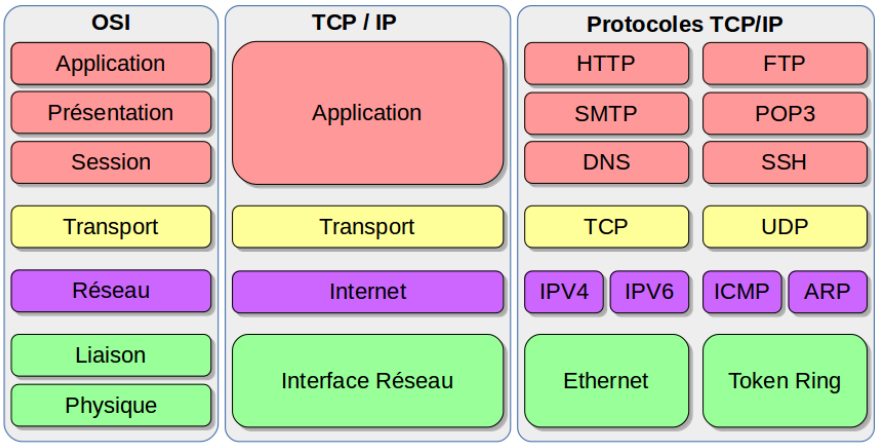
▷ ***Dans quel sens parcourt-on les couches du modèle OSI ?***

Cela dépend que l'on soit émetteur ou récepteur. Dans le cas où l'on envoie des données, on parcourt les couches de « haut en bas », et de « bas en haut » si on reçoit des données.



4.1.2 Le modèle TCP/IP

Internet s'appuie sur une version « allégée » du modèle OSI, et chronologiquement antérieure (1974) : c'est le modèle **TCP/IP**.



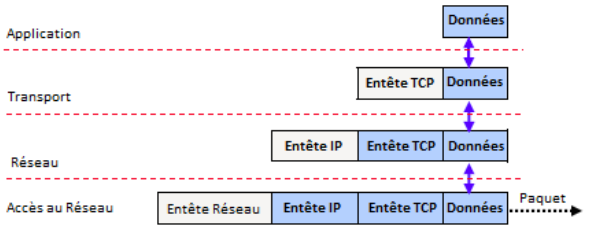
Contrairement au modèle OSI, ce modèle ne dispose que de 4 couches. Chaque couche est associée à plusieurs **protocoles** spécifiques, dont les deux plus importants sont TCP et IP. Un protocole est un **ensemble de règles** à suivre pour communiquer. Par exemple, le protocole IP définit, dans sa version 4 (IPV4), que chaque réseau et chaque ordinateur est identifié par une adresse de 32 bits (4 octets), généralement écrite à l'aide de 4 nombres compris entre 0 et 255 en notation décimale pointée, comme **192.168.1.1**.

Le modèle TCP/IP est conçu pour répondre à un certain nombre de critères parmi lesquels :

- Le fractionnement des messages en **paquets**
- L'utilisation d'un système d'**adresses**
- L'acheminement des données sur le réseau, appelé **routage**
- Le **contrôle des erreurs** de transmission des données

Encapsulation des données

Lors d'une transmission, les données traversent les 4 couches du modèle TCP/IP (de haut en bas dans le cas d'une émission) et forment un **paquet** de données. À chaque couche, une information est ajoutée au paquet de données, il s'agit d'un **en-tête**, ensemble d'informations qui garantit la transmission. On parle d'**encapsulation** des données.



Lors de la réception d'un paquet, l'entête de chaque couche est lu, supprimé, et le reste du message est transmis à la couche suivante. À la fin, on récupère le message original.

4.2 Le modèle TCP/IP en détails

4.2.1 Couche Interface Réseau

Brancher deux machines entre elles (couche 1 du modèle OSI)

Le rôle principal de la couche 1 est de fournir le **support de transmission de la communication**. Ce support peut être :

- un câble : dans le cas où la machine est reliée « physiquement ».

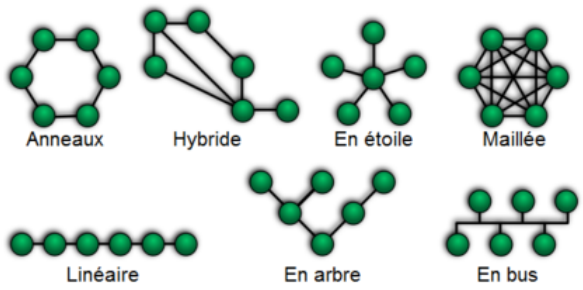
  Les premiers câbles utilisés étaient des **câbles coaxiaux**. Aujourd'hui, on utilise des **câbles à paires torsadées** (appelés vulgairement câbles Ethernet) ou encore de la **fibre optique**.

- l'air : dans le cas où la machine transmet et reçoit ses données par ondes (Wi-Fi, Bluetooth...)

Les données sont acheminées sous **forme binaire** via le support de transmission.

Topologie du réseau

En réseau, la **topologie** est la manière selon laquelle on branche les machines entre elles.
Le schéma ci-contre présente les principales topologies.
Certaines ne sont plus utilisées comme la topologie en bus ou en anneau. D'autres, comme les topologies en étoile ou maillée sont utilisées majoritairement sur Internet.



Avec une **topologie en bus**, une seule machine peut envoyer des données. C'est comme si plusieurs personnes discutent ensemble dans une même pièce : si tout le monde parle en même temps, on ne comprend rien. Une seule personne peut parler à la fois.
On considère qu'au-delà de 50 machines, la probabilité de parler en même temps qu'une autre machine est plus forte que celle de parler seul, et donc que le réseau ne marchera plus...

Avec une **topologie en anneau**, chaque machine ne peut correspondre qu'avec 2 voisins. Pour transmettre un message à un ordinateur distant, il faut le faire transiter par tous les ordinateurs intermédiaires.
Sur un tel réseau, un **jeton** circule et permet à la machine qui le récupère d'émettre ou de recevoir un message.

Avec une **topologie en étoile**, le noeud central joue le rôle d'aiguilleur. Dans ce cas, chaque machine ne reçoit que les messages dont elle est le destinataire.

Question 01 Citer des avantages et des inconvénients pour les topologies en anneau, en étoile et maillée.

Faire communiquer des machines sur un réseau local (couche 2 du modèle OSI)

Le rôle de la couche 2 du modèle OSI est de **connecter des machines sur un réseau local**, mais également de **détecter les erreurs de transmission** (sans chercher à les corriger).

Un identifiant : l'adresse MAC

Pour distinguer deux machines sur le réseau, il est nécessaire d'attribuer à chacune d'entre elles un identifiant unique.
Cet identifiant, c'est l'**adresse MAC** (pour *Media Access Control*), encore appelée **adresse physique**. C'est l'adresse liée à la **carte réseau** utilisée. Elle est **unique au monde**.

Tout objet possédant une carte réseau - ordinateur, smartphone, montre connectée, décodeur TV... - possède une adresse MAC unique, codée sur 6 octets (soit 48 bits), et souvent écrite sous forme héxadécimale de la manière suivante :

F8 : AC : B8 : 05 : 64 : C2

Les trois premiers octets représentent le constructeur de la carte réseau, et les trois suivants une adresse unique choisie par le fabricant.

Question 02 Combien d'adresses MAC différentes existe-t-il ?

Question 03 Combien d'adresses MAC différentes un constructeur peut-il former ?

En réalité, certaines adresses MAC spéciales sont réservées.

C'est le cas de l'adresse FF : FF : FF : FF : FF : FF dite **adresse de broadcast**. Un message dont le destinataire est l'adresse de broadcast sera lu par **toutes les machines du réseau**.

Question 04 Lire l'adresse MAC de la machine sur laquelle vous lisez ce document à l'aide de la commande **ipconfig** depuis un invite de commande Windows ou **ifconfig** depuis un terminal Linux.

Un protocole : Ethernet

Une fois nos machines correctement identifiées, il est nécessaire de mettre en place un **protocole de communication**, c'est à dire un ensemble de règles à respecter pour communiquer. Ce protocole est un langage de communication commun à toutes les machines du réseau.

Le protocole le plus couramment utilisé (ce n'est cependant pas le seul) est le **protocole Ethernet**. Avec ce protocole, tout message transmis sur le réseau doit contenir :

- l'adresse MAC du destinataire
- l'adresse MAC de l'émetteur (ou de la source)
- le message (!)

Question 05 Pourquoi l'adresse MAC de l'émetteur doit-elle être renseignée ?

En plus de ces informations, il est également indispensable de rajouter :

- le protocole de couche 3 utilisé, aussi appelé *Ether Type* : la couche 2 faisant le lien entre les couches 1 (physique) et 3 (réseau), elle contient donc des informations sur la couche directement supérieure, à savoir le protocole réseau utilisé (IPv4, IPv6, ARP...). Nous y reviendrons dans la suite de ce cours.
- un code de détection d'erreur, appelé **CRC** : ce code est calculé par la machine qui émet le message (c'est une opération mathématique sur les bits d'informations transmis) et inscrit dans le message Ethernet. La machine qui reçoit le message recalcule le CRC et le compare à la valeur donnée. Si les codes correspondent, c'est que le message n'a pas été altéré durant la transmission.

Toutes ces informations mises bout à bout forment une **trame Ethernet** :

MAC DST (destinataire)	MAC SRC (source)	Protocole couche 3	Données	CRC
6 octets	6 octets	2 octets	46 - 1500 octets	4 octets

L'**en-tête Ethernet** est formé des adresses MAC destinataire et source, du protocole de couche 3 et du CRC, pour un total de **18 octets**. Les données transmises ne peuvent dépasser 1500 octets, soit une trame Ethernet maximale de 1518 octets !

Question 06 Voici un exemple de trame Ethernet (le CRC n'y figure pas), transcrite sous forme hexadécimale :

```
d8 a7 56 b5 b1 3a c4 8e 8f fc 51 7b 86 dd 60 0f
10 b1 00 29 11 40 2a 01 cb 1c 0c f1 91 00 22 ba
58 56 72 d0 69 13 2a 01 cb 1c 0c f1 91 00 da a7
56 ff fe b5 b1 3a 81 90 00 35 00 29 37 37 fc 64
01 00 00 01 00 00 00 00 00 00 06 67 6f 6f 67 6c
65 03 63 6f 6d 04 68 6f 6d 65 00 00 01 00 01
```

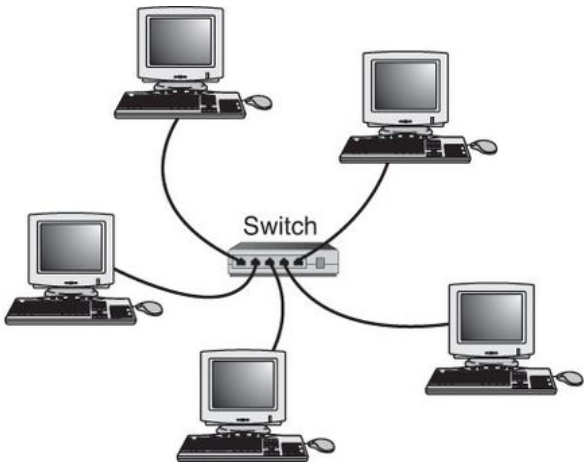
Code	Ether Type
0800	IPV4
0806	ARP
86DD	IPV6

Quelle est l'adresse MAC de la machine ayant envoyé ce message ?
Quel est le protocole de couche 3 utilisé ?

Un matériel, le commutateur

Dans un réseau en étoile typique, le noeud central est une machine particulière, appelée **commutateur** (ou *switch* en anglais).

Un commutateur est un boîtier sur lequel sont branchées toutes les machines de notre réseau, et dont le rôle est d'aiguiller les différentes trames Ethernet émises par les différentes machines.



Le commutateur étant un matériel de couche 2 du modèle OSI, il est ainsi capable de lire un en-tête Ethernet. Ainsi, lorsqu'il reçoit un message sur une de ses entrées :

- il associe l'entrée sur laquelle il reçoit le message à l'adresse MAC source. Les adresses MAC sont stockées dans une partie de sa mémoire, appelée **table CAM** (pour *Content-Addressable Memory*)
- il transmet le message à l'adresse MAC destinataire

▷ ***Et comment fait le switch s'il ne connaît pas l'adresse MAC destinataire ?***

C'est simple : il envoie alors le message sur tous ses ports.
Si la machine destinataire est branchée, elle recevra le message. Très souvent, elle répondra, et le switch saura alors sur quel port cette dernière est branchée, et associera son adresse MAC.

▷ ***Le commutateur dispose-t-il d'une adresse MAC lui aussi ?***

Généralement non, car son rôle n'est pas de recevoir et d'envoyer des messages mais d'aiguiller ces derniers sur le réseau.
On ne peut donc pas envoyer de message à destination du commutateur lui-même.

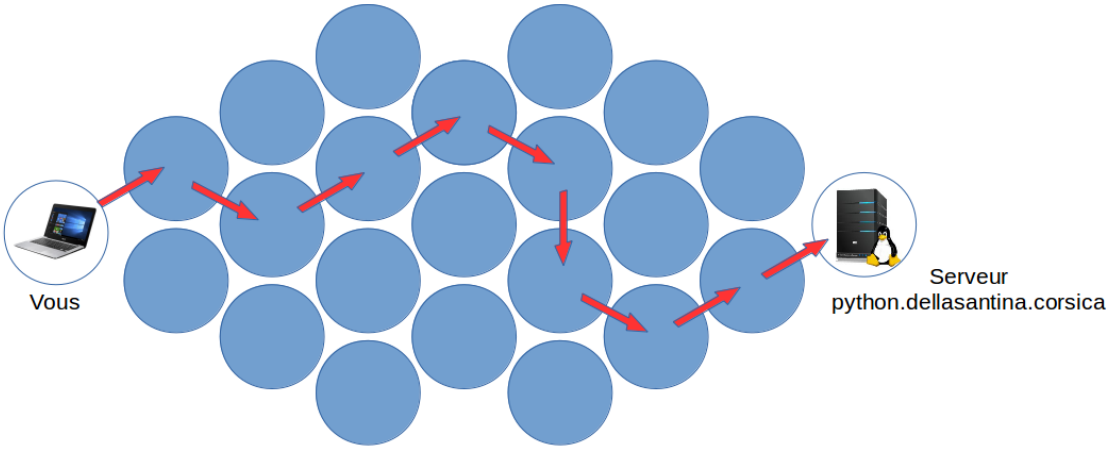
4.2.2 Couche Internet

Dans la partie précédente, nous avons découvert comment identifier des machines dans un réseau (un ensemble d'ordinateurs connectés entre eux). Internet est LE réseau formé par tous les réseaux du monde, interconnectés entre eux.

▷ ***Comment connecte-t-on deux réseaux ensemble ? Avec un gros switch ?***

C'est plus compliqué ! En premier lieu, votre réseau domestique n'est pas relié *directement* à tous les réseaux existants. Le réseau Internet est de type **maillé**, il est donc complètement **décentralisé**, contrairement à votre réseau domestique où la box Internet joue le rôle de commutateur, et donc d'élément central.

Par exemple, pour rejoindre le réseau du site *python.dellasantina.corsica*, il est nécessaire de traverser un ensemble de réseaux intermédiaires, comme présenté dans le schéma ci-dessous.



Le chemin reliant votre réseau à celui de *python.dellasantina.corsica* n'est pas unique. D'ailleurs, il est tout à fait possible que ce chemin change à chaque nouvel échange d'informations avec le réseau distant. Le choix de cet « itinéraire » n'est cependant pas laissé au hasard : c'est ce que l'on appelle le **routage**. Nous y reviendrons plus loin.

Question 07 Déterminer le nombre de réseaux intermédiaires empruntés pour joindre le serveur de *python.dellasantina.corsica* grâce à la commande `traceroute` de Linux (ou `tracert` de Windows) :

```
1 traceroute python.dellasantina.corsica
```

☞ *Chaque réseau traversé correspond à une ligne dans le retour de la commande précédente*

Comme vous avez pu le remarquer, chaque réseau dispose d'un identifiant, un ensemble de 4 nombres séparés par des points : il s'agit de son **adresse IP**. De cette façon, tout réseau est identifié de manière unique sur Internet !

Un identifiant, l'adresse IP

Pour naviguer sur Internet, l'adresse MAC ne suffit plus. Il est nécessaire de donner une adresse à chaque réseau, et même plus : chaque machine de chaque réseau doit posséder une adresse unique afin d'être identifiable et accessible : l'**adresse IP**.

Dans le protocole **IPV4** (*Internet Protocol*, version 4), une adresse IP est codée sur **32 bits** (soit 4 octets). Un octet représentant un nombre entre 0 et 255, il est commun d'utiliser la **notation décimale pointée** pour représenter une adresse IP :

$$\underset{\downarrow}{1100\,0000}\;\underset{\downarrow}{1010\,1000}\;\underset{\downarrow}{0000\,0001}\;\underset{\downarrow}{0000\,0011}$$
$$192 \;.\; 168 \;.\; 1 \;.\; 3$$

Question 08 Combien existe-t-il d'adresses IP différentes ? Ce nombre vous paraît-il important ?

Question 09 Avec le protocole IPV6, on code les adresses IP sur 128 bits. Combien existe-t-il d'adresses IPV6 différentes ?

Question 10 Parmi les adresses IP suivantes, une seule est incorrecte. Laquelle ? Et pourquoi ?

192.168.90.109 123.234.345.0 10.11.12.13

Question 11 À l'aide d'un terminal, identifier l'adresse IP de la machine que vous utilisez avec la commande `ifconfig` pour Linux (et `ipconfig` pour Windows).

```
1 ifconfig
```

Masque de sous-réseau

En réalité, une adresse IP contient deux informations : une adresse de réseau et une adresse de machine. Pour savoir où s'arrête la première et où commence la deuxième, il nous faut une seconde information : le **masque de sous-réseau**.
Le masque de sous-réseau est une série de 32 bits (avec le protocole IPV4) permettant de séparer adresses réseau et machine.
La règle est simple : **les bits à 1 du masque de sous-réseau correspondent à la partie réseau de l'adresse IP**.

Exemple.

Adresse IP :	192.168.1.3	$\Longrightarrow$	1100 0000 1010 1000 0000 0001 0000 0011
Masque :	255.255.255.0	$\Longrightarrow$	1111 1111 1111 1111 1111 1111 0000 0000

Grâce au masque, on distingue la partie réseau de la partie machine .
Dans cet exemple, la partie réseau est **192.168.1** et la partie machine **3**.

Question 12 Dans chaque cas, donner la partie réseau et la partie machine des adresses IP suivantes :

1. Adresse : 10.0.37.1 Masque : 255.0.0.0
2. Adresse : 86.228.32.107 Masque : 255.255.0.0
3. Adresse : 128.10.31.22 Masque : 255.240.0.0

Dans ce dernier exemple, il est impossible de donner les adresses réseau et machine sous forme décimale, car les 0 et 1 du masque sont « coupés » au milieu d'un octet : on conservera donc une écriture binaire pour ces adresses.

De plus, les 0 et les 1 ne doivent pas être mélangés dans un masque de sous-réseau.
Les 1 sont toujours à gauche, et les 0 toujours à droite.
Voici un exemple de masques correct et incorrect :

- Correct : 1111 1111 1111 1111 1111 1111 0000 0000
- Incorrect : 1111 1111 1111 1111 0001 1111 0000 0011

Il n'y a de fait que 8 valeurs décimales correctes pour les octets d'un masque.

Enfin, il est courant d'utiliser la **notation CIDR** pour les masques de sous-réseau.
Avec cette notation, un masque est associé à un nombre entre 1 et 31, correspondant au nombre de 1 dans son écriture binaire.

1. 00000000 : **0**
2. 10000000 : **128**
3. 11000000 : **192**
4. 11100000 : **224**
5. 11110000 : **240**
6. 11111000 : **248**
7. 11111100 : **252**
8. 11111110 : **254**
9. 11111111 : **255**

Exemple. 255.255.240.0 $\Longrightarrow$ 1111 1111 1111 1111 1111 0000 0000 0000 $\Longrightarrow$ Nombre correspondant : 20
Ainsi, le couple IP/Masque 129.10.37.189/255.255.240.0 est encore noté 129.10.37.189/20.

Question 13 Donner la notation CIDR correspondante aux masques suivants :

255.240.0.0 255.255.255.0 255.255.255.192

Question 14 Donner l'écriture décimale pointée des masques dont le CIDR est :

23 8 17

Question 15 Donner la partie réseau et la partie machine des adresses IP suivantes :

1. Adresse : 22.84.1.12/24
2. Adresse : 86.228.32.107/16
3. Adresse : 128.10.31.22/20

Combien de machines sur mon réseau ?

Considérons l'adresse IP suivante : 202.52.27.21/24. Si on décompose réseau et machine, on obtient :

202 11001010	**52** 00110100	**27** 00011011	**21** 00010101
Réseau			*Machine*

Avec le masque réseau /24, on dispose donc de 8 bits (soit 1 octet) pour l'adresse de notre machine. On peut donc écrire $2^8 = 256$ adresses de machines différentes, en commençant par 202.52.27.**0** et en terminant par 202.52.27.**255** :

202 11001010	**52** 00110100	**27** 00011011	**0** 00000000
			1 00000001
			...
			255 11111111

En réalité, ces deux adresses « extrêmes » dans lesquels tous les bits « machine » sont à 0 ou 1 sont réservées :

- 202.52.27.**0** est l'**adresse du réseau** lui-même
- 202.52.27.**255** est l'**adresse de broadcast** : c'est une adresse particulière permettant de cibler **toutes les machines du réseau**.

Une machine ne pourra donc pas utiliser ces adresses. Ainsi, sur le réseau 202.52.27.0/24, nous disposons de 254 adresses pour nos machines, de 202.52.27.**1** à 202.52.27.**254**.

▷ ***Le nombre d'adresses disponibles dépend uniquement du masque utilisé !***

Question 16 Déterminer la formule mathématique donnant le nombre d'adresses machines disponibles en fonction du CIDR. Écrire la fonction Python équivalente.

Question 17 Pour chacune des adresses IP suivantes, déterminer l'adresse du réseau, l'adresse de broadcast ainsi que le nombre d'adresses machines disponibles :

1. 192.168.1.18/24
2. 10.20.30.40/8
3. 132.39.48.60/16
4. 128.10.31.22/20

Question 18 Parmi les adresses IP suivantes, dire si ce sont des adresses de réseau, de machine ou de broadcast :

1. 192.168.0.15/255.255.255.0
2. 10.8.65.31/255.255.255.224
3. 10.0.0.255/255.255.254.0

Le protocole IP

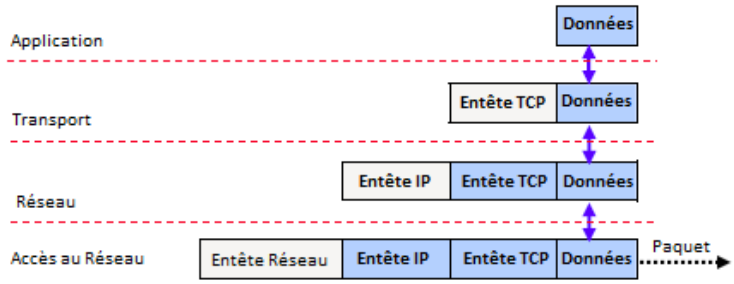
Maintenant que les adresses IP n'ont plus de secret pour nous, il est temps de découvrir de quelle manière elles sont utilisées pour acheminer les données entre les réseaux. Nous avons vu que pour la couche « Réseau », c'est le switch (ou commutateur) qui a pour rôle de répartir les informations aux différentes machines d'un même réseau, à l'aide du **protocole Ethernet**. Pour échanger des informations entre réseaux, nous allons utiliser le **protocole IP** (*Internet Protocol*).

Le protocole IP définit la forme du message envoyé sur le réseau. Ce message, appelé **datagramme** ou **paquet**, doit contenir en plus des données l'adresse IP du destinataire et l'adresse IP de la source. Il est de la forme suivante (en IPV4) :

En-tête	IP SRC (source)	IP DST (destinataire)	Données
12 octets	4 octets	4 octets	0 - 65536 octets

▷ ***Je suis perdu... Avec le protocole Ethernet, on avait des trames, et maintenant on a des datagrammes ?***

Les datagrammes sont **une partie** des trames Ethernet, ils sont **encapsulés** dans une trame lors de la « fabrication » de celle-ci dans la couche Ethernet. Rappelez-vous le schéma quelques pages plus haut :



☞ *Rappelez-vous également que pour créer un message, on parcourt les couches du haut vers le bas, en encapsulant les données. Lorsqu'un message est reçu, on « ouvre » petit à petit les différents en-têtes, du bas vers le haut.*

Si on reprend notre trame Ethernet :

MAC DST (destinataire)	MAC SRC (source)	IP Type	Données	CRC
6 octets	6 octets	2 octets	46 - 1500 octets	4 octets

La partie *Données* de notre trame contient en réalité notre datagramme IP :

Ethernet			IP				CRC
MAC DST	MAC SRC	IP Type	En-tête	IP SRC	IP DST	Données	CRC

Et la partie *Données* de notre datagramme IP contiendra des informations sur la couche supérieure, la couche Transport... Cependant, pour un switch (matériel de couche Réseau), ces informations n'ont aucune valeur : il se contente de lire les 12 premiers octets de la trame pour connaître les adresses MAC source et de destination. Le reste ne l'intéresse pas.

▷ ***Pourquoi l'adresse IP source est-elle mentionnée en premier ? Ce ne devrait pas être l'adresse IP du destinataire ?***

Peu importe ! En effet, si une machine reçoit une trame Ethernet (et si le switch a bien fait son travail), alors l'adresse MAC de destination est la sienne. Le message lui est donc nécessairement destiné. Sinon, la carte réseau ne reconnaît pas son adresse MAC et détruit purement et simplement la trame reçue !

▷ ***Mais à quoi bon inscrire une adresse IP dans notre message alors ?***

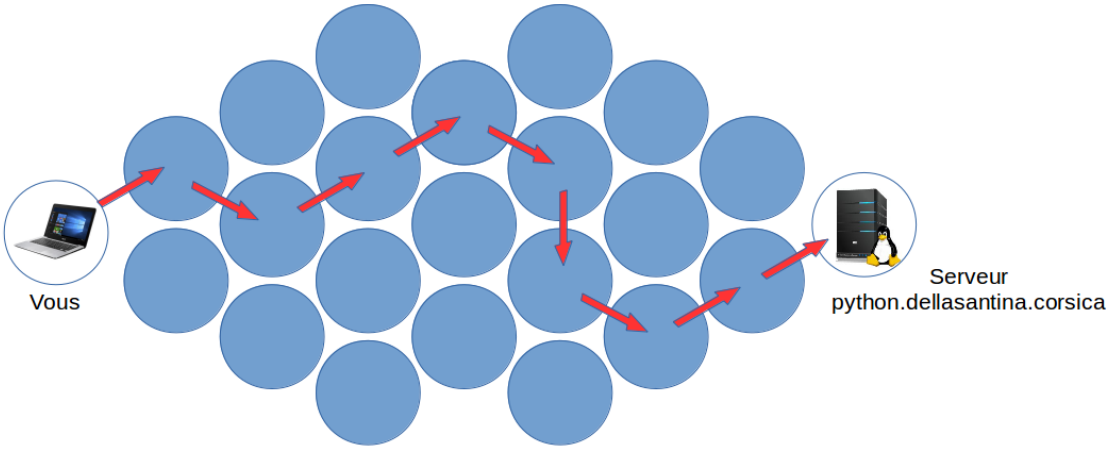
Afin qu'il puisse sortir de notre réseau ! Et pour sortir du réseau, il faut « parler Internet », autrement dit comprendre le protocole Internet, et donc utiliser des adresses IP. C'est le rôle du **routeur**, qui n'a rien à faire des adresses MAC, mais qui utilise le protocole IP.

▷ ***Les données d'un datagramme IP contiennent jusqu'à 65536 octets, et une trame Ethernet ne peut dépasser 1518 octets...***

Bonne remarque ! C'est pour cette raison que les données sont **fragmentées**, c'est à dire découpées en plusieurs morceaux. Et ça tombe bien : c'est encore le rôle du routeur de fragmenter les données.

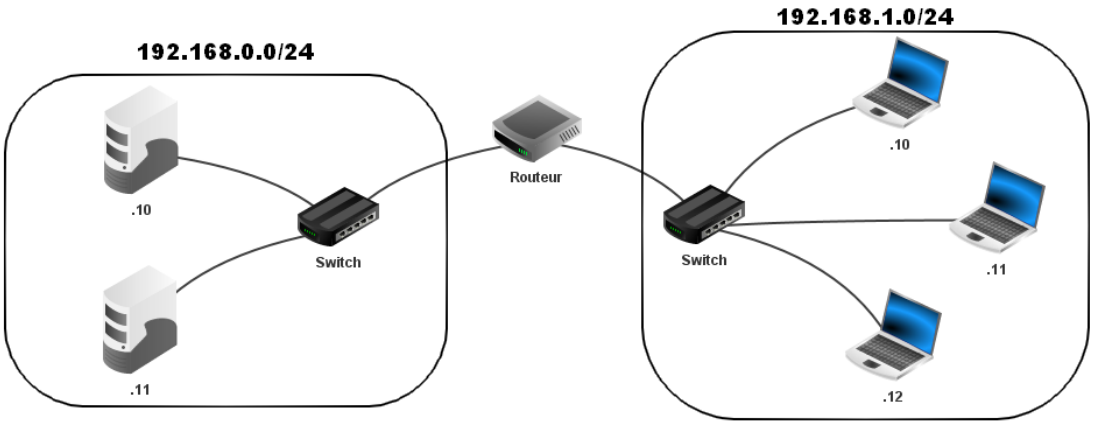
Le routage

Pour relier deux réseaux, on utilise un **routeur**, dont le rôle est d'aiguiller nos données vers le bon réseau. C'est un matériel de couche Internet, il peut donc lire l'entête Internet de notre message, et donc avoir accès aux adresses IP.
Un routeur possède **plusieurs interfaces** (plusieurs cartes réseaux) : une pour chaque réseau auquel il est connecté.
Il est également capable de **relayer** les paquets qui ne lui dont pas destinés. Et c'est souvent le cas : lorsque vous vous connectez au site *python.dellasantina.corsica*, vous traversez en réalité un grand nombre de routeurs, qui acceptent gentillement de transmettre vos datagrammes... Vous vous souvenez ?



Prenons l'exemple de deux réseaux :

- Le réseau A : 192.168.0.0/24
- Le réseau B : 192.168.1.0/24



Les ordinateurs de chaque réseau sont identifiés sur ce schéma avec la partie machine de leur adresse IP. Les switch n'ont pas d'adresse IP, ce sont des matériels de couche Réseau.

▷ ***Et le routeur, il n'en a pas ?***

Si, et il en a même 2, une pour chacun des deux réseaux.

Question 19 Réaliser le réseau précédent à l'aide du logiciel Filius. Donner à chaque machine son adresse IP.
Donner les adresses 192.168.0.1 et 192.168.1.1 au routeur (attention à l'interface choisie !)
Pour télécharger le logiciel (version exécutable) :
`http://nsi.dellasantina.corsica/documents/reseau/filius-1.8.2.zip`

Nous allons réaliser un « *ping* » entre deux machines du réseau A. Le *ping* est une commande souvent employée pour debugger un réseau. Elle utilise un protocole de couche Internet, le **protocole ICMP**. La machine 192.168.0.10 va « *pinguer* » la machine 192.168.0.11 en lui transmettant un *Echo Request* (ping) autrement dit « Es-tu là ?? » auquel cette dernière va répondre par un *Echo Reply* (pong) autrement dit « Oui oui, je suis là ! ».

Question 20

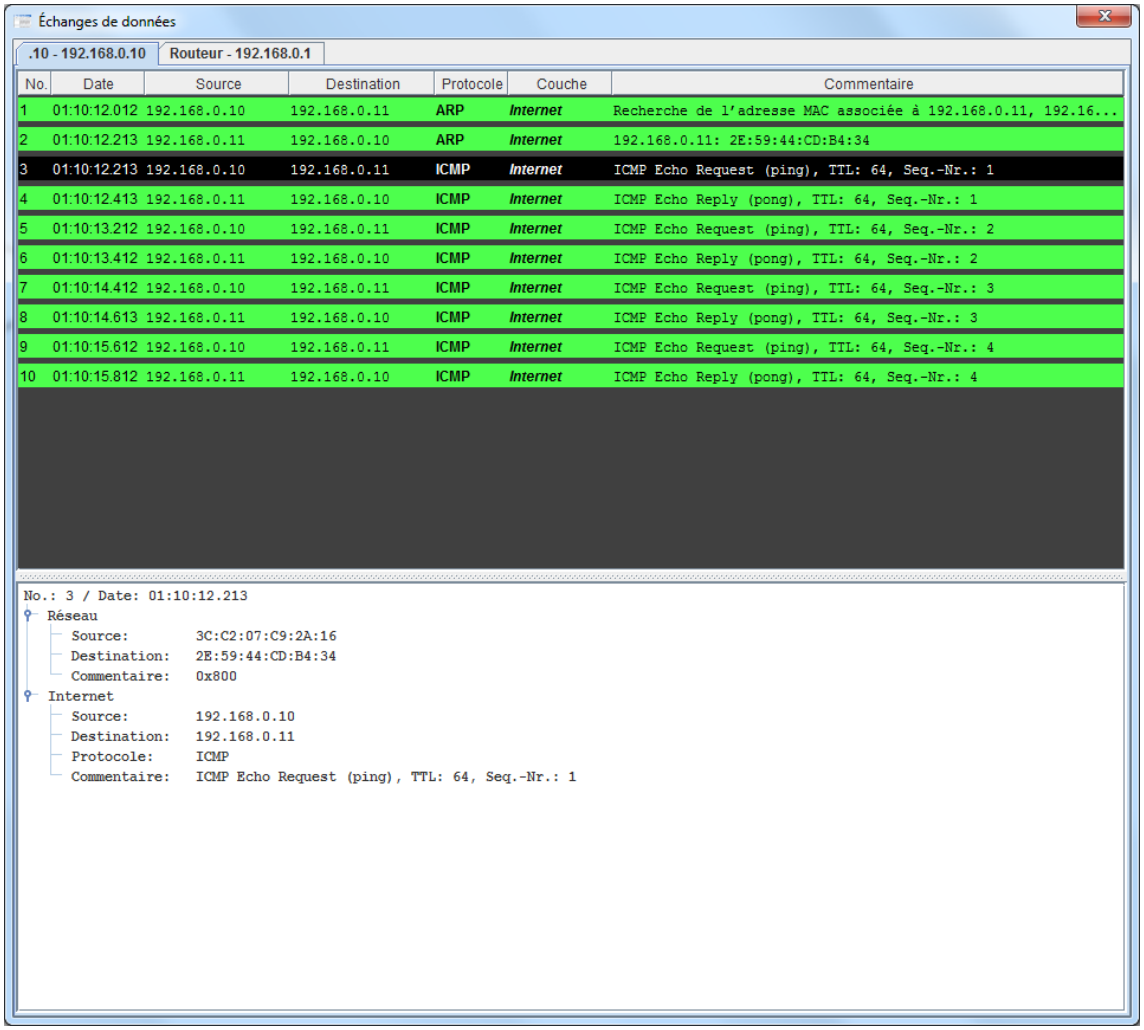
- Passer en mode *Simulation* (Ctrl-R ou flèche verte) et cliquer sur la machine 192.168.0.10.
- Cliquer sur *Installation des logiciels* et déplacer *Invite de commandes* dans la colonne de gauche.
- Cliquer sur *Appliquer les modifications* puis sélectionner *Ligne de commande*.
- Écrire la commande suivante :

```
1 ping 192.168.0.11
```

- Observer les échanges entre les deux machines.

Il est possible de voir les messages échangés entre les deux machines.

Question 21 Toujours en mode Simulation, faire un clic-droit sur 192.168.0.10 puis choisir *Afficher les échanges de données.* Cliquer sur le premier paquet ICMP pour afficher des informations complémentaires (adresses MAC, adresses IP, protocole...)



▷ ***Que représentent les deux premiers paquets, avec le protocole* ARP *?***

Le but du protocole ARP est d'associer adresse MAC et adresse IP. Avant d'envoyer le *ping*, 192.168.0.10 envoie une requête ARP à destination de 192.168.0.11 afin de connaître son adresse MAC, sans quoi il ne pourra pas former sa trame Ethernet. Pour cela, il fait un **broadcast** en utilisant l'adresse MAC $FF:FF:FF:FF:FF:FF$ comme destination, de sorte que le switch envoie le message à tout le monde (c'est pour cela que lors de l'envoi du *ping*, on voit un message transmis à 192.168.0.11 mais aussi au routeur). D'ailleurs, le routeur reçoit une requête ARP (regardez les échanges de données du routeur) mais ne répond pas : ce n'est pas son adresse IP. La machine 192.168.0.11 reçoit également la requête, et répond à l'émetteur

(192.168.0.10) en joignant son adresse MAC. Ce dernier reçoit la réponse, et connaît à présent l'adresse MAC de 192.168.0.11 : il peut alors former sa trame Ethernet et donc envoyer son *ping*.

▷ ***Difficile de faire plus compliqué...***

Effectivement, on pourrait croire que les adresses MAC sont inutiles et qu'il suffirait d'utiliser uniquement les adresses IP, mais le switch ne comprendrait rien, car il ne « parle pas le langage Internet », autrement dit le protocole IP. Il faut donc **respecter le protocole Ethernet** afin que le switch fasse correctement son travail de distribution des trames.

▷ ***J'ai une idée : on envoie toujours un broadcast réseau avec l'adresse*** $FF:FF:FF:FF:FF:FF$***. Malin non ?***

Bien au contraire : notre réseau serait envahi de requêtes en tous genres ! C'est comme si le facteur vous livrait des copies des lettres de tous vos voisins, que vous n'auriez pas le droit d'ouvrir. Non, ce n'est pas malin !

Tentons maintenant de pinguer une machine du réseau voisin.

Question 22 Effectuer un ping de 192.168.1.10 depuis la machine 192.168.0.11.

Le ping est impossible, et la réponse vous est fournie immédiatement :

```
1 Destination inaccessible
```

Et c'est normal : la machine 192.168.0.11 est située sur le réseau 192.168.0.0/24. Sur ce réseau, les adresses vont de 192.168.0.1 à 192.168.0.254 (192.168.0.0 étant l'adresse du réseau lui-même et 192.168.0.255 l'adresse de broadcast IP).
La machine 192.168.**1**.10 ne fait donc pas partie de ce réseau, et se retrouve inaccessible !

▷ ***Ce n'est pas le rôle du routeur de relier les deux réseaux ?***

Bien sûr, mais il faut signaler à 192.168.0.11 l'adresse IP du routeur, qui ne peut pas la deviner. Cette adresse est appelée l'adresse **passerelle**, adresse par laquelle il faut passer pour transmettre nos messages hors du réseau.

Question 23 Repasser en mode *Conception* (Ctrl-D ou le marteau) et modifier la passerelle de 192.168.0.11.
Faire alors un *ping* depuis 192.168.0.11 à destination de 192.168.1.10.

▷ ***Toujours pas ! Pourtant, on voit bien que 192.168.1.10 reçoit bien le message. Il est trop fainéant pour répondre ?***

Question 24 Expliquer pourquoi 192.168.1.10 ne répond pas, et corriger l'erreur.

Ouf ! Ça fonctionne enfin. Profitez-en pour régler le problème sur tous les ordinateurs de nos 2 réseaux.

Question 25 Vérifier les données échangées au niveau de 192.168.1.10, et vérifier l'adresse MAC source du premier message ICMP en provenance de 192.168.0.11. Comparer cette adresse MAC avec celle de 192.168.0.11. Sont-elles identiques ?

Récapitulatif des échanges

Pour comprendre pourquoi les adresses sont identiques, il faut analyser la situation étape par étape. Pour plus de simplicité dans les notations, appelons :

- Alice : 192.168.0.11, l'émetteur
- Bob : 192.168.1.10, le destinataire
- Roger et Robert, les deux interfaces du routeur, respectivement 192.168.0.1 et 192.168.1.1

Vous pouvez vérifier en examinant les échanges de données entre les machines, en mode Simulation, et après avoir pingué avec Succès Bob depuis Alice.

1. Alice souhaite pinguer Bob, une machine hors de son réseau. Elle va donc contacter sa passerelle, dont elle ne connaît pas l'adresse MAC.
2. Alice émet alors une requête ARP afin de récupérer l'adresse MAC de sa passerelle, Roger
3. Roger répond à la requête ARP d'Alice
4. Alice envoie alors son ping à Roger :

Ethernet			IP			
@MAC Roger	@MAC Alice	ICMP	En-tête	@IP Alice	@IP Bob	Données

5. Roger est le destinataire du message (c'est son adresse MAC) : il « ouvre » donc le message et lit la partie correspondante à la couche Internet (Roger est un routeur, il se fiche des adresses MAC)
6. Roger transmet le message à Robert, afin qu'il le transmette sur le réseau de Bob.
7. Robert doit connaître l'adresse MAC de Bob. Il envoie donc une requête ARP afin de connaître l'adresse MAC associée à l'adresse IP de Bob.
8. Robert transmet alors le message à Bob, sans modifier les adresses IP, mais sa carte réseau **modifie l'en-tête Ethernet** :

Ethernet			IP			
@MAC Bob	**@MAC Robert**	ICMP	En-tête	@IP Alice	@IP Bob	Données

9. Bob reçoit enfin le ping, auquel il s'empresse de répondre. Cependant, **Bob ne pourra jamais accéder à l'adresse MAC d'Alice**. En effet, elle a été modifié par Robert, sans quoi Bob n'aurait pas pu lui transmettre sa réponse ! Mais Bob dispose de l'adresse IP d'Alice, c'est donc suffisant. Il compose alors son message (à destination de sa passerelle, car Alice ne fait pas partie de son réseau) :

Ethernet			IP			
@MAC Robert	**@MAC Bob**	ICMP	En-tête	@IP Bob	@IP Alice	Données

10. Robert reçoit le message, le transmet gentillement à Roger, qui s'apprête à joindre la destinataire (Alice) et modifie donc l'en-tête Ethernet :

Ethernet			IP			
@MAC Alice	**@MAC Roger**	ICMP	En-tête	@IP Bob	@IP Alice	Données

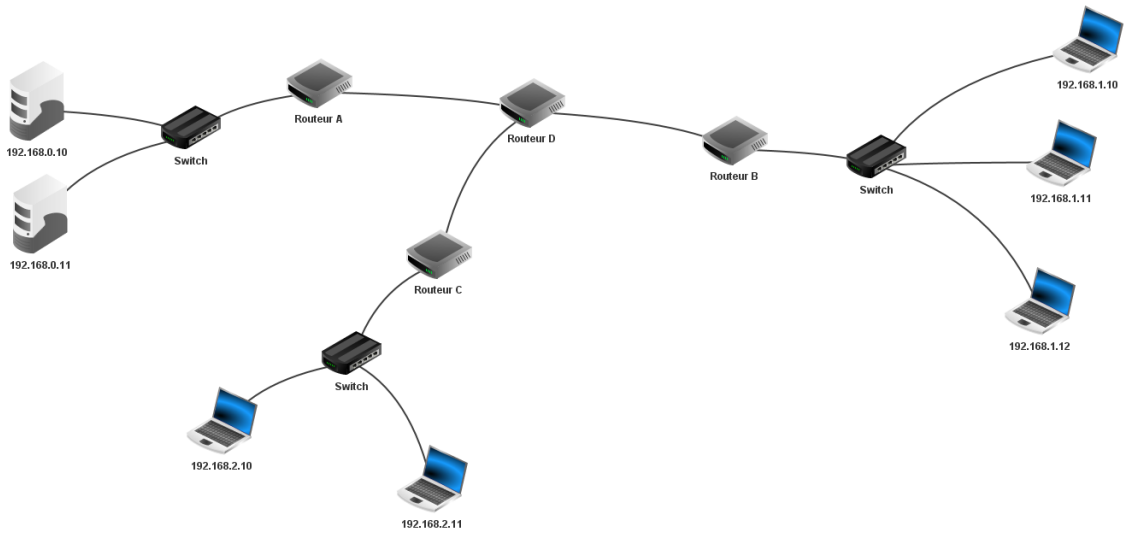
11. Alice reçoit un message de Roger (c'est son adresse MAC qui est renseignée) mais la source du message est Bob (elle le sait, car elle peut lire les informations de la couche Internet, et donc son adresse IP, adresse qu'elle vient de pinguer).

Alice vient en effet de pinguer Bob il n'y a pas si longtemps : une page de théorie (et encore, nous n'avons pas parlé des switch !), mais quelques millisecondes se sont écoulées en pratique.

Terminons par une situation un peu plus complexe, mais plus proche de « la réalité du terrain ».

Question 26 Récupérer le réseau suivant à l'adresse :

`http://nsi.dellasantina.corsica/documents/reseau/routage.fls`



On désigne toujours les machines 192.168.0.11 et 192.168.1.10 par Alice et Bob.

Question 27 Essayer de pinguer Bob depuis le poste d'Alice. Où semble s'arrêter le message ?

Le réseau de Bob n'est plus directement accessible au routeur A : il faut lui expliquer quel chemin suivre, et comme il n'y en a qu'un seul ici (vers le routeur D), on peut lui fournir une passerelle par défaut (un endroit où aller si le destinataire d'un message n'est pas dans son réseau).

Question 28 Modifier la passerelle du routeur A afin que le message puisse sortir du réseau.

Question 29 Tenter à nouveau un ping. Où s'arrête le message ?

La situation se complique une fois le message arrivé au routeur D. Quel chemin emprunter ? Il faut bien sûr passer par le routeur B, mais hors de question de fournir une passerelle par défaut : tous les messages iraient dans cette direction, y compris d'éventuels messages pour le réseau 192.168.2.0/24.
C'est là qu'intervient la notion de **table de routage** : c'est un tableau contenant d'un côté les réseaux à rejoindre, et de l'autre les passerelles à contacter pour joindre ce réseau. Il est de la forme :

IP Destination	Masque	Passerelle	Interface

Pour joindre notre réseau 192.168.1.0/24 (le masque associé est 255.255.255.0) depuis le routeur D, nous allons ajouter une ligne dans la table de routage :

IP Destination	Masque	Passerelle	Interface
192.168.1.0	**255.255.255.0**	??	??

La passerelle à joindre est l'adresse IP de l'interface du routeur B située « côté routeur D ». Ici, c'est 11.0.0.2.

IP Destination	Masque	Passerelle	Interface
192.168.1.0	255.255.255.0	**11.0.0.2**	??

L'interface est l'adresse IP de l'interface du routeur D à utiliser. C'est celle « côté routeur B », ici 11.0.0.1.

IP Destination	Masque	Passerelle	Interface
192.168.1.0	255.255.255.0	11.0.0.2	**11.0.0.1**

Question 30 Ajouter cette ligne dans la table de routage du routeur D.

Question 31 Ajouter deux lignes supplémentaires afin que les réseaux 192.168.0.0/24 et 192.168.2.0/24 soient joignables par le routeur D.

☞ *Bravo ! Vous venez d'écrire votre première table de routage !*

Question 32 Re tenter un ping, et corriger le dernier petit problème.

Question 33 Configurer les tables de routage de chacun des routeurs.

4.2.3 Couche Transport

Les protocoles des couches précédentes permettaient d'envoyer des informations d'une machine à une autre. **La couche transport permet à des applications tournant sur des machines distantes de communiquer**. Le problème consiste à identifier ces applications. En effet, suivant la machine et son système d'exploitation, l'application pourra être un programme, une tâche, un processus...
De plus, la dénomination de l'application peut varier d'un système à un autre, c'est pourquoi un système de numéro a été mis en place afin de pouvoir associer un type d'application à un type de données. Ces identifiants sont appelés **ports**.

La couche transport contient deux protocoles permettant à deux applications d'échanger des données indépendamment du type de réseau emprunté (c'est-à-dire indépendamment des couches inférieures), il s'agit des protocoles suivants :

- **TCP** (*Transmission Control Protocol*) : un protocole orienté connexion qui assure le contrôle des erreurs

  On l'utilise pour des applications qui nécessitent un **transport fiable** des données, mais qui n'ont pas de besoin particulier en ce qui concerne la vitesse de transmission

- **UDP** (*User Datagram Protocol*) : un protocole non orienté connexion dont le contrôle d'erreur est archaïque

  On l'utilise pour des applications qui nécessitent un **transport immédiat** des informations, mais qui peuvent se permettre de perdre quelques informations

Question 34 Selon vous, quel type d'application nécessite le recours de TCP ? d'UDP ?

Caractéristiques du protocole TCP

Nous l'avons vu assez rapidement dans le paragraphe précédent sur le protocole IP, mais les données échangées sont souvent **fragmentées**, c'est à dire découpées en plusieurs parties (appelées paquets ou datagrammes) et distribuées séparemment. Et ce pour plusieurs raisons :

- le protocole Ethernet impose une taille maximale de 1500 octets pour les données de la trame. Ainsi, toute information de plus de 1,5ko sera nécessairement fragmentée
- les paquets empruntent des chemins différents sur le réseau, ce qui permet d'éviter l'engorgement de celui-ci
- la machine qui envoie les données peut envoyer des paquets « simultanément » à plusieurs autres machines, sans être forcée d'envoyer la totalité d'une donnée à une seule machine avant de passer à la suivante

C'est là qu'intervient le protocole TCP, qui a pour but (entre autres) :

- de remettre en ordre les datagrammes en provenance du protocole IP
- de vérifier le flot de données afin d'éviter une saturation du réseau
- de permettre l'initialisation et la fin d'une communication de manière « courtoise »

Grâce au protocole TCP, les applications peuvent **communiquer de façon sûre** (grâce au système d'accusés de réception du protocole TCP), indépendamment des couches inférieures. Cela signifie que les routeurs (qui travaillent dans la couche Internet) ont pour seul rôle l'acheminement des données sous forme de datagrammes, sans se préoccuper du contrôle des données, car celui-ci est réalisé par le protocole TCP.

Lors d'une communication à travers le protocole TCP, les deux machines doivent **établir une connexion**. La machine émettrice (celle qui demande la connexion) est appelée **client**, tandis que la machine réceptrice est appelée **serveur**. On dit qu'on est alors dans un environnement Client-Serveur. Les machines dans un tel environnement communiquent en mode connecté, c'est-à-dire que la communication se fait dans les deux sens.

Un message de la couche transport est appelé un **segment** et posséde un en-tête, comme sur les couches inférieures. Ce segment se retrouve encapsulé dans le datagramme IP, lui même encapsulé dans la trame Ethernet qui circulera sur le réseau. Le schéma ci-dessous présente les principales informations contenues dans le segment TCP (certaines informations sont volontairement masquées dans un soucis de simplification) :

Port Source	Port Destination	???	Flags	???	Checksum	???	Données
2 octets	2 octets	-	6 bits	-	2 octets	-	-

Les **ports** sont des adresses permettant d'identifier l'application utilisée sur une machine (Navigateur Web, Client de messagerie, logiciel de visioconférence...). C'est l'équivalent en couche Transport des adresses IP de la couche Internet et des adresses MAC de la couche Réseau. Un port est codé sur 2 octets (16 bits), il existe donc $2^{16} = 65\,536$ ports différents (de 0 à 65535). Les plus couramment utilisés sont :

Application	**Port**
Web (HTTP)	80
Web (HTTPS)	443
FTP	20/21
DNS	53
Jeux Blizzard	6112

Les **flags** (ou *drapeaux* en français) permettent de préciser le type de segment transféré (paquet urgent ou non, demande de connexion, fin de connexion...). Ils sont au nombre de 6, chacun étant représenté par un bit. Parmi eux, nous retiendrons les drapeaux *SYN* (pour établir une nouvelle connexion), *ACK* (pour accuser réception d'un message) et *FIN* (pour terminer une connexion), qui nous détaillerons plus loin. Si le bit associé au drapeau est égal à 1, on dit que le drapeau est **levé**.

Enfin, le **checksum** est une somme de contrôle (comme le CRC dans la couche Réseau) permettant de vérifier l'intégrité de l'en-tête.

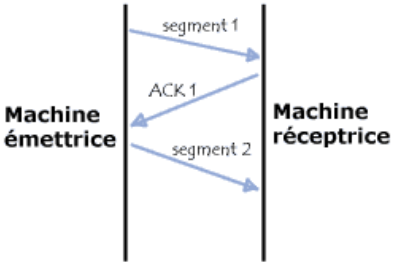
▷ ***Ce n'est pas redondant ? Pourquoi calculer deux sommes de contrôle ?***

Ceci est dû au fait que les couches Réseau et Transport ne communiquent pas entre elles (seules des couches adjacentes peuvent communiquer).

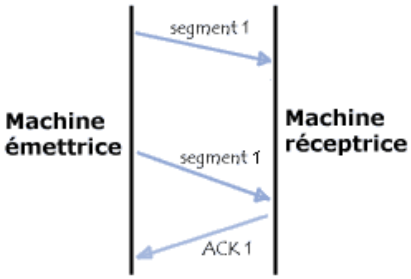
Fiabilité des transferts

Le protocole TCP permet d'assurer le transfert des données de façon fiable.

En réalité, le protocole TCP possède un système d'accusé de réception permettant au client et au serveur de s'assurer de la bonne réception mutuelle des données. Lors de l'émission d'un segment, un numéro d'ordre (appelé aussi numéro de séquence) est associé. A réception d'un segment de donnée, la machine réceptrice va retourner un segment de donnée dont le drapeau *ACK* est à 1 (afin de signaler qu'il s'agit d'un accusé de réception) accompagné d'un numéro d'accusé de réception égal au numéro d'ordre précédent.



De plus, grâce à une minuterie déclenchée dès réception d'un segment au niveau de la machine émettrice, le segment est réexpédié dès que le temps imparti est écoulé, car dans ce cas la machine émettrice considère que le segment est perdu.



Toutefois, si le segment n'est pas perdu et qu'il arrive tout de même à destination, la machine réceptrice saura grâce au numéro d'ordre qu'il s'agit d'un doublon et ne conservera que le dernier segment arrivé à destination.

Établissement d'une connexion TCP

L'établissement de la connexion se fait en 3 étapes :

1. Le client demande au serveur d'établir une connexion, en levant le drapeau *SYN*, pour *Synchronisation*

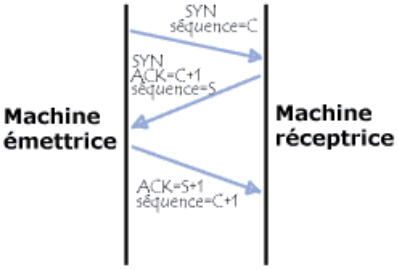
   Le client : « *Allô ?* »

2. Le serveur recevant la demande *SYN* doit normalement répondre qu'il est d'accord pour communiquer avec le client. Pour cela il va envoyer un *ACK* en réponse (pour *acknowledgement*, aquittement en français). Il demande également si le client veut bien communiquer avec lui et positionne aussi le flag *SYN* dans sa réponse. Il y aura donc les flags *SYN* et *ACK* positionnés dans sa réponse.

   Le serveur : « *Je vous entends ! Et vous ?* »

3. Le client reçoit le *ACK* du serveur, la communication client → serveur est alors validée. Il renvoie à son tour un *ACK*, afin de valider la connexion réciproque serveur → client.

   Le client : « *5 sur 5 !* »



L'établissement de la connexion TCP s'est donc fait par l'échange de trois paquets. C'est pour cela qu'on l'appelle **Three Way Handshake** ou « poignée de main tripartite » en français.

Continuation de la connexion

Maintenant que la communication est établie, les applications peuvent s'échanger des paquets autant qu'elles le veulent ! Nous ne détaillerons pas plus cette partie, qui fait intervenir d'autres flags comme le flag *PUSH*.

Fin de la connexion

Le client peut demander à mettre fin à une connexion au même titre que le serveur. La fin de la connexion se fait de la manière suivante :

- Une des machines envoie un segment avec le drapeau *FIN* à 1, et l'application se met en état d'attente de fin, c'est-à-dire qu'elle finit de recevoir le segment en cours et ignore les suivants
- Après réception de ce segment, l'autre machine envoie un accusé de réception avec le drapeau *FIN* à 1 et continue d'expédier les segments en cours. Suite à cela la machine informe l'application qu'un segment *FIN* a été reçu, puis envoie un segment *FIN* à l'autre machine, ce qui clôture la connexion

Bon, après la théorie, la pratique ! Découvrons à présent la couche Application, celle que nous, utilisateurs, manipulons.

4.2.4 Couche Application

La **couche Application** est la couche située au sommet des couches de protocoles TCP/IP. Celle-ci contient les applications réseaux permettant de communiquer grâce aux couches inférieures. Les logiciels de cette couche communiquent donc grâce à un des deux protocoles de la couche inférieure (la couche Transport) c'est-à-dire TCP ou UDP.

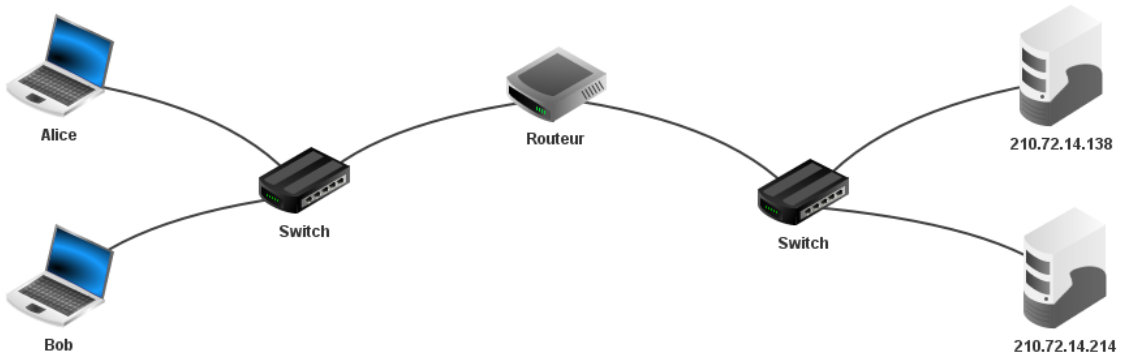
Dans cette partie, nous allons présenter quelques uns des protocoles de la couche Application, à travers un TP utilisant le logiciel que vous maîtrisez à présent à la perfection : Filius.

Naviguer sur le WEB avec le protocole HTTP

Lorsque vous naviguez sur le WEB, vous utilisez un programme conçu spécialement pour cela : un navigateur WEB (comme Firefox, Chrome ou Safari). Le rôle du navigateur est d'envoyer et de recevoir des données en utilisant le **protocole HTTP**.

Question 35 Récupérer le réseau suivant et l'ouvrir avec Filius.

`http://nsi.dellasantina.corsica/documents/reseau/couche_application.fls`



Les machines du réseau 210.72.14.0/24 sont des serveurs WEB, c'est à dire qu'elles sont capables d'envoyer des données grâce au protocole HTTP. Ces données peuvent être des pages au format HTML (que nous découvrirons dans un autre chapitre), des fichiers Javascript, des images... Le client (vous) demande ces informations au serveur (le site WEB) en renseignant l'adresse IP du serveur.

Question 36 Passer en mode Simulation, puis installer un navigateur WEB sur le poste d'Alice.
Rentrer l'adresse IP 210.72.14.138 depuis le navigateur WEB, et observer les échanges entre Alice et le serveur WEB.

Question 37 Afficher les échanges de données depuis la machine Alice (Clic-droit → Afficher les échanges...). Identifier :

1. Les 3 paquets correspondant à la connexion entre Alice et le serveur (le Three Way Handshake) (couche Transport)
2. Le nombre de requêtes effectuées par Alice (couche Application)
3. Le nombre de réponses du serveur (couche Application)

Question 38 Quel est le port utilisé par le navigateur d'Alice pour contacter le serveur 210.72.14.138 ?

☞ *Le port est inscrit juste après les deux-points, à la fin de l'adresse IP* $\Longrightarrow$ *210.72.14.138 :**port***

La première requête d'Alice envers le serveur est la suivante :

```
1 GET / HTTP/1.1 Host: 210.72.14.138
```

De cette manière, Alice demande au serveur de lui transmettre la page d'accueil du site WEB (on parle de **racine**, représentée par le caractère /). Le serveur répond par un :

```
1 HTTP/1.1 200 OK Content-type: text/html
```

signifiant qu'il transmet du **code HTML**, permettant au navigateur de mettre en forme les données reçues. En cliquant sur le paquet correspondant à la réponse du serveur, on peut voir le code HTML, à la fin du message. On remarque au passage que le protocole utilisé dans les échanges est bien le protocole HTTP, dans sa version 1.1.

Question 39 Relever les adresses MAC dans ce premier paquet envoyé par le serveur.
À qui appartient l'adresse MAC source ? l'adresse MAC de destination ?

Question 40 À quoi correspond la deuxième requête d'Alice ? Pourquoi le serveur répond-il en envoyant cette fois-ci plusieurs paquets ?

Des noms de domaines grâce au protocole DNS

Il n'est pas très pratique de contacter un serveur WEB en utilisant son adresse IP... C'est pourquoi on préfère utiliser des **noms de domaines**, composés de lettres plutôt que de chiffres, afin d'identifier de manière unique un serveur. Pour cela, il faut pouvoir faire l'association entre un nom de domaine et une adresse IP, et c'est exactement le rôle d'un **serveur DNS**.

Question 41 Passer en mode Conception :

1. Ajouter une interface au routeur, afin de pouvoir connecter notre serveur DNS :

   Clic-droit sur le routeur → Configurer → Gérer les connexions → Clic sur le `+` *dans le volet de droite.*

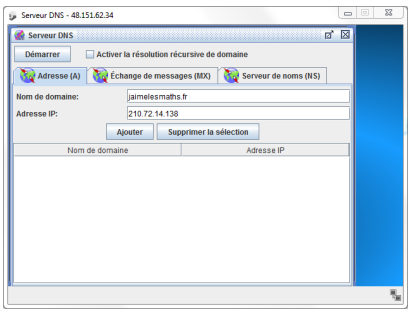
2. Ajouter une machine, nommée *Serveur DNS*, que l'on reliera au routeur. Cette machine aura pour adresse IP 48.151.62.34/24, et l'interface correspondante du routeur aura pour adresse IP 48.151.62.1/24.

   Ne pas oublier de régler la passerelle du serveur DNS !

3. Décocher puis cocher à nouveau *Routage automatique* sur le routeur (onglet *Général*).

Question 42 Passer en mode Simulation :

1. Installer un serveur DNS sur la machine nouvellement ajoutée à notre réseau.
2. Ouvrir le serveur DNS, et associer le nom de domaine *jaimelesmaths.fr* à l'adresse IP 210.72.14.138 (onglet *Adresse (A)*).
3. Démarrer le serveur DNS.



Question 43 Ouvrir maintenant le navigateur depuis le poste d'Alice et essayer de se connecter au site WEB :

http ://jaimelesmaths.fr

Question 44 Repasser en mode Conception, et configurer la machine Alice pour régler l'erreur rencontrée dans la question précédente. Il doit y avoir une case vide quelque part...

Question 45 Une fois le site WEB ouvert, décrire simplement les échanges entre Alice, le serveur DNS et le serveur Web (qui est contacté en premier, qui répond à qui...)

Question 46 Se connecter au serveur 210.72.14.214 depuis le navigateur sur la machine Bob.
Quel est le port utilisé par le navigateur de Bob pour dialoguer avec le serveur ?

Question 47 Repérer le nom de domaine sur le site WEB ouvert à la question précédente, et configurer le serveur DNS en conséquence afin de se connecter avec ce nom de domaine.

Question 48 Mettez en place votre propre serveur WEB dans le réseau 210.72.14.0/24 (pas d'adresse IP imposée) et choisissez-lui un nom du domaine, à préciser sur le serveur DNS.

☞ *Vous pouvez même installer un éditeur de texte sur votre serveur WEB et modifier le code HTML de la page d'accueil (nous découvrirons ce langage dans un autre chapitre)*

Connexion à distance avec le protocole SSH

La suite dans le prochain épisode !

4.3 Exercices

Exercice 01 Le but de cet exercice est de programmer un CRC très simple. Un CRC est un moyen de contrôle d'intégrité des données. Il permet de vérifier que des données échangées entre deux machines n'ont pas été corrompues lors du transfert. Pour calculer le CRC d'une chaîne de caractères, on additionne les valeurs numériques de chaque caractère (valeurs ASCII, obtenue via la fonction `ord` de Python), et on calcule le reste dans la division euclienne de ce nombre par 256 (afin d'obtenir une valeur entre 0 et 255). Cette valeur est ensuite codée en héxadécimal.

1. Programmer la fonction `crc(chaine)` en Python.

   ☞ *On utilisera la fonction* `hex` *pour convertir un entier en héxadécimal...*

2. Calculer le CRC des chaînes de caractères `"bonjour"` et `"bonsoir"` et vérifier que les CRC sont différents.
3. Même question avec les chaînes de caractères `"Je suis Voldemort"` et `"Tom Elvis Jedusor"`. Que peut-on en conclure ?

Exercice 02 Écrire une fonction `IPBinaire(ip)` qui convertit la chaîne de caractères `ip` (donnée en écriture décimale pointée) et retourne une chaîne de caractères correspondante à l'écriture binaire de cette adresse IP.

```
>>> IPBinaire("192.168.1.3")
'11000000101010000000000100000011'
```

☞ *On pourra utiliser la fonction* `bin` *pour transformer un entier en binaire...*

Exercice 03 Écrire une fonction `IPDecimale(ip)` qui convertit la chaîne de caractères `ip` (donnée en écriture binaire) et retourne une chaîne de caractères correspondante à l'écriture décimale pointée de cette adresse IP.

```
>>> IPDecimale("11000000101010000000000100000011")
'192.168.1.3'
```

Exercice 04 Écrire une fonction `cidr(masque)` qui retourne un entier correspondant au CIDR associé au masque réseau, donné sous forme de chaîne de caractères. Si le masque réseau n'est pas valide, la fonction doit retourner `None`.

Exercice 05 Écrire une fonction `masque(cidr)` qui retourne le masque réseau sous forme décimale pointée, en fonction du CIDR donné (sous forme d'entier). Si le CIDR n'est pas un entier compris entre 1 et 31, la fonction doit retourner `None`.

Exercice 06 Écrire une fonction `adresseReseau(ip, masque)` qui retourne l'adresse du réseau associée au couple IP/-masque donné (forme décimale pointée).

Exercice 07 Écrire une fonction `adresseBroadcast(ip, masque)` qui retourne l'adresse de broadcast associée au couple IP/masque donné (forme décimale pointée).

Exercice 08 Écrire une fonction `typeIP(ip, masque)` qui retourne `"Machine"`, `"Réseau"` ou `"Broadcast"` selon l'adresse IP et le masque donnés (sous forme de chaînes de caractères).