WIKILIVRES

Les réseaux informatiques

Une version à jour et éditable de ce livre est disponible sur Wikilivres,
une bibliothèque de livres pédagogiques, à l'URL :
https://fr.wikibooks.org/wiki/Les_r%C3%A9seaux_informatiques

Introduction aux réseaux

Un réseau informatique est un ensemble d'ordinateurs reliés entre eux qui échangent des informations. À cela près qu'outre des ordinateurs, un réseau peut aussi contenir des équipements spécialisés, comme des hub, des routeurs, et bien d'autres équipements que nous aborderons dans ce cours. Dans les grandes lignes, un réseau est intégralement composé : d'équipements informatiques (ordinateurs et matériel réseau) et de liaisons point-à-point qui relient deux équipements entre eux. Mais tous les réseaux sont très différents les uns des autres. Il y a de nombreuses manières d'organiser les liaisons et ordinateurs d'un réseau, des milliers de manières de gérer les transferts d'informations sur le réseau. Pour simplifier le propos, on peut quand même classer les réseaux suivant plusieurs critères. Dans ce qui va suivre, nous allons voir comment on classe les réseaux suivant leur taille et leur étendue géographique, mais aussi suivant ce à quoi servent les ordinateurs du réseau.

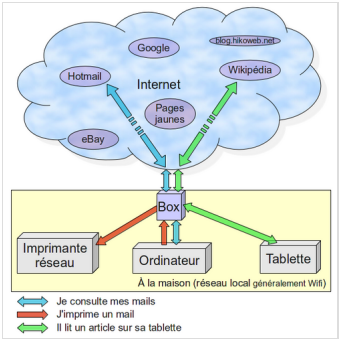
La classification des réseaux en fonction de leur taille : LAN, WAN, PAN, MAN, et Internet

Certains réseaux sont limités à une salle, voire un bâtiment. D'autres sont tellement grands qu'ils font la taille d'une ville ou d'un quartier, quand d'autres ont une étendue nationale. Internet est un réseau d'étendue mondiale : grâce au net, vous pouvez parfaitement communiquer avec quelqu'un qui est situé aux États-Unis, en Russie, au Japon, etc. Et évidemment, tous ces réseaux portent des noms différents : on n'appelle pas de la même manière un réseau qui ne contient qu'une centaine d'ordinateurs et un réseau de taille planétaire.

Les réseaux locaux

Les réseaux les plus petits contiennent entre 2 et 100 ordinateurs, qui sont reliés avec des câbles ou une connexion sans fil. Ces réseaux sont appelés des **réseaux locaux** ou encore des **réseaux LAN** (Local Area Network). Les réseaux internes aux entreprises ou aux écoles sont de ce type : par exemple, les ordinateurs d'une salle informatique peuvent généralement communiquer entre eux.

Comme autre exemple, vous avez tous chez vous un réseau local qui comprend votre box internet et tout ce qui est connecté dessus. Un tel réseau local, qui tient dans une simple chambre, est appelé un **PAN : Personnal Area Network**. Un PAN comprend généralement des équipements qui appartiennent à une même personne ou famille, qui sont regroupés dans la même maison, sur une étendue d'une dizaine de mètres.



Réseau local personnel, de type PAN.

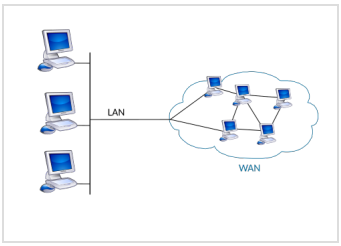
Les réseaux interconnectés

À côté de ces réseaux assez petits, on trouve les réseaux de taille moyenne, qui permettent d'interconnecter des réseaux locaux proches :

- les MAN, pour **Metropolitan Area Network** ont généralement la taille d'une ville
- les WAN, pour **Wide Area Network**, permettent de relier entre eux des réseaux locaux dans des villes différentes.

Il existe deux grandes solutions pour créer un WAN : les lignes louées et les lignes RNIS.

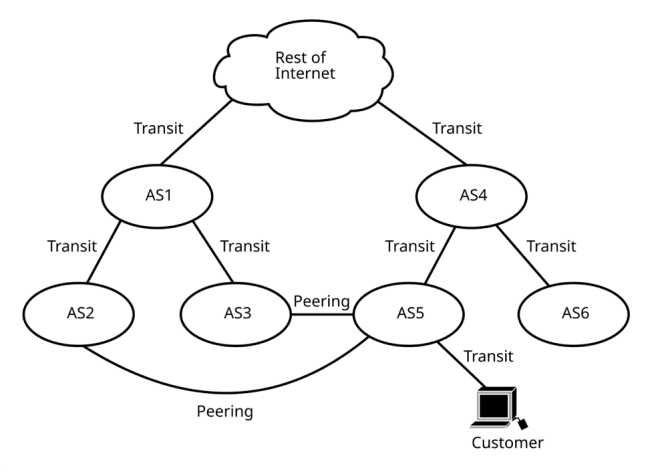
- Les **lignes louées** permettent de créer un lien permanent entre les deux réseaux locaux distants. Ce lien est une ligne téléphonique, qui est louée par l'opérateur de télécommunication. Le prix d'une ligne louée est indépendant de l'utilisation qui en est faite : peu importe que l'on échange beaucoup d'informations ou très peu sur cette ligne, le prix reste le même. Autant dire que ce genre de ligne est d'autant plus rentable que son utilisation approche des 100% en permanence.
- Les **lignes RNIS** fonctionnent sur un principe similaire, si ce n'est que que la liaison est disponible à la demande, et non permanente (24 heures sur 24). La liaison s'ouvre quand deux ordinateurs veulent l'utiliser pour échanger des informations. Quand les deux ordinateurs ont cessé d'échanger des données, la ligne se ferme après un certain délai d'inactivité. Le prix de ces lignes RNIS sont proportionnelle au temps d'utilisation. Autant dire qu'il vaut mieux limiter la durée de transmission au mieux pour réduire le coût de ce genre de ligne.



LAN et WAN

Les réseaux mondiaux

Internet est une interconnexion de réseaux à l'échelle mondiale. C'est d'ailleurs ce qui lui a valu son nom : internet est l'abréviation de *interconnection of networks*. Environ 47000 réseaux de grande envergure, des réseaux autonomes, sont interconnectés pour former internet. Ces 47000 réseaux sont souvent des réseaux appartenant à des fournisseurs d'accès. Les informations qui transitent sur internet passent par des câbles en fibre optique. Les câbles qui relient ces 47000 réseaux parcourent le monde entier : pour donner un exemple, il y a environ 6 à 7 câbles qui traversent l'Atlantique qui permettent aux réseaux américains de communiquer avec les réseaux européens.



Internet est une interconnexion de réseaux autonomes (ici nommés AS).

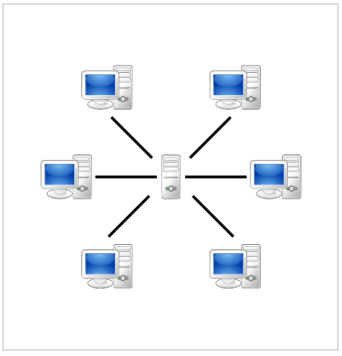
Au début, Internet n'était pas disponible pour le grand public, mais servait essentiellement aux grandes organisations. L'internet de l'époque était vraiment différent de l'internet actuel : non seulement les technologies utilisées n'étaient pas les mêmes, mais il n'y avait pas de sites web. En ce temps, internet servait surtout à échanger des données, télécharger de grands volumes de données. C'est dans les années 1990 que sont apparues les technologies qui ont permis la conception des sites web, à savoir l'HTML, les adresses URL et le protocole HTTP. De nos jours, internet est surtout utilisé par les particuliers pour consulter des sites web. L'ensemble des sites web accessibles sur internet est ce qu'on appelle le web : il faut ainsi faire la différence entre le web et internet, internet étant un réseau et le web un ensemble de sites web.

La classification en fonction des interconnexions : le client-serveur et le pair à pair

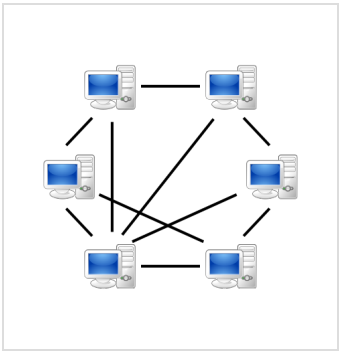
Les réseaux informatiques peuvent aussi être catégorisés selon la manière dont les ordinateurs échangent des données. On peut ainsi classer les réseaux en trois grands types :

- les architectures un tiers ;
- les architectures client-serveur ;
- les architectures pair à pair.

Un même réseau physique peut servir pour les trois. Si on prend internet, celui-ci permet à des ordinateurs de communiquer en client-serveur : c'est ce qui est fait lors de la consultation de sites web. Il permet aussi les échanges en pair à pair : les applications de téléchargement basées sur le protocole BitTorrent utilisent des échanges en pair à pair.



Un réseau de type client-serveur



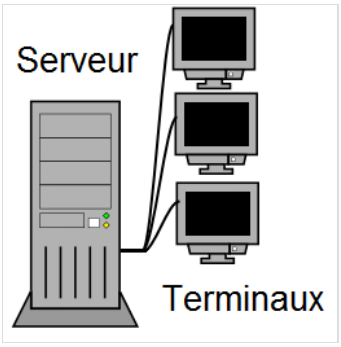
Un réseau pair-à-pair

Les réseaux organisés autour d'un terminal

Les premiers réseaux locaux datent des années 1960. À cette époque, les ordinateurs étaient très chers et prenaient énormément de place : un ordinateur pouvait facilement remplir une pièce entière. Seules les grandes entreprises et services publics d'état pouvaient se payer ces ordinateurs. Il n'y avait donc pas un poste par personne : les organisations n'avaient qu'un seul ordinateur qu'il fallait se partager. Les utilisateurs n'avaient pas accès à l'ordinateur directement. Ils devaient passer par des terminaux, des systèmes qui permettaient d'afficher des informations et de saisir des commandes et du texte : ils étaient constitués d'un clavier, d'un écran, et de quelques circuits rudimentaires. Ces terminaux se contentaient d'envoyer ou de recevoir des informations au gros ordinateur central, qui traitait les demandes des terminaux une par une.

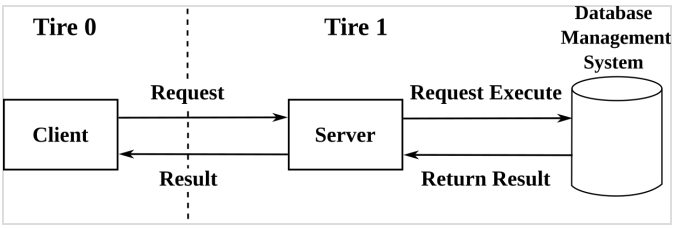
Cette organisation avec un seul ordinateur relié à des terminaux a survécu, et s'est même adaptée à l'ère internet. En effet, de nombreuses organisations utilisent encore des terminaux qui communiquent avec des serveurs via internet. Pensez à un distributeur automatique de billets : celui-ci n'est ni plus ni moins qu'un terminal assez évolué qui communique avec un serveur de la banque, via des connexions internet. L'architecture un tiers correspond au cas où plusieurs terminaux sont reliés à un ordinateur central, peut importe que cette connexion se fasse via internet ou via un réseau local.

Les réseaux de type clients-serveurs

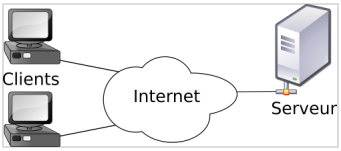


Terminaux et serveur

L'architecture client-serveur est une amélioration de l'organisation précédente, la différence étant que les terminaux sont de véritables ordinateurs. Elle fait intervenir au minimum deux ordinateurs : un **serveur** et un ou plusieurs **clients**. Le serveur est un ordinateur qui contient des données utiles et très importantes à traiter, comme un site web, des données partagées sur internet, les documents d'une entreprise, etc. Les clients veulent avoir accès aux données présentes sur le serveur : pour cela, ils doivent envoyer une demande au serveur. Le serveur traitera cette demande après réception, et renvoie le résultat de la demande. La grosse différence entre un terminal et un client est que le terminal n'est pas un ordinateur, mais un matériel sans intelligence et qui n'a aucune capacité de calcul. Cette méthode est utilisée pour les sites web et les applications web : le site web est stocké sur le serveur, tandis que le client reçoit ces fichiers et affiche le site sur votre ordinateur.



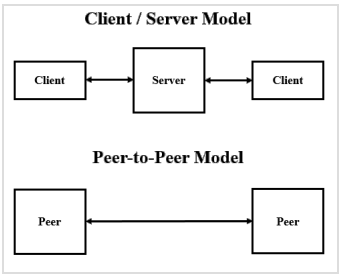
Organisation client-serveur, avec illustration de la position des données (la BDD).



Organisation client-serveur.

Cependant, ce système a une ambiguïté majeure. Outre l'affichage et le stockage des données, un programme doit effectuer un paquet de traitements. Dans le modèle avec des terminaux, ces traitements étaient pris en charge par le serveur central : un terminal est un matériel sans intelligence et qui n'a aucune capacité de calcul. Mais un client est un vrai ordinateur qui peut parfaitement prendre en charge les calculs à faire. Ainsi, avec le système client-serveur, on ne sait pas où effectuer les traitements sur les données. De nos jours, on peut déporter les calculs aussi bien du côté du serveur que du côté client. Par exemple, on peut utiliser des technologies comme Javascript pour exécuter du code dans un navigateur web client. Mais dans d'autres situations d'entreprise, on peut déporter les calculs côté serveur. Dans le cas où les calculs sont majoritairement déportés du côté du serveur, le client est qualifié de **client léger**. Dans le cas contraire, on parle de **client lourd**.

Les réseaux pair à pair

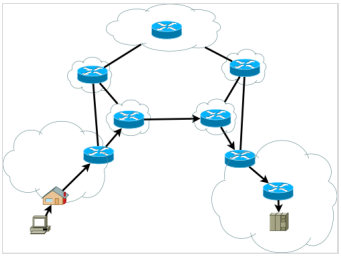


Comparaison entre architectures client-serveur et peer-to-peer.

L'**architecture pair à pair**, aussi notée P2P, peut être vu comme une sorte d'amélioration du client-serveur. Dans le client serveur, les rôles de serveur et de client sont attribués définitivement. Mais avec le pair à pair, tout ordinateur peut alternativement être serveur et client. Pour mieux comprendre ce que cela signifie, on peut prendre l'exemple d'utilisation la plus connue du P2P : le transfert de fichiers. De nombreux logiciels utilisent des protocoles P2P, comme BitTorrent, pour permettre l'échange de fichiers entre utilisateurs. Ces protocoles permettent le téléchargement de fichiers demandés à partir d'autres "serveurs", ou alors envoyer des morceaux de fichiers aux autres clients. Ainsi, chaque ordinateur peut, avec ce logiciel, servir de client ou de serveur. Attention, cependant : c'est l'échange de fichiers qui est décentralisé, mais d'autres parties du protocole peuvent faire intervenir un serveur. La découverte des sources de téléchargement en est un bon exemple : selon le protocole, il peut être effectué en pair à pair ou en client-serveur. Cela permet de distinguer deux types d'architectures pair à pair :

- Avec du **P2P centralisé**, la mise en relation est réalisée par un serveur central, qui centralise les informations sur chaque client. Pour information, ces serveurs de mise en relation sont appelés des trackers, du moins pour le protocole BitTorrent. Ce serveur central contient une base de données qui mémorise quels sont les clients qui possèdent tel ou tel fichier. Cette BDD est mise à jour à chaque connexion d'un nouveau client. Quand un client se connecte au réseau P2P, il envoie au serveur la liste des fichiers qu'il est prêt à partager. Un client qui souhaite télécharger quelque chose a juste à demander au serveur de le mettre en relation avec un client qui possède le fichier voulu. Une fois la mise en relation faite, les deux clients vont se transmettre le fichier sans passer par le serveur.
- L'organisation précédente est à comparer avec le **P2P décentralisé**, où il n'y a pas de serveur central pour la mise en relation. Ici, les transferts de données se font de poste à poste, sans qu'il n'y ait de serveur bien précis. La mise en relation des clients s'effectue par des mécanismes bien plus complexes de découverte des clients, que nous n'aborderons pas ici.

Routage, *bridging* et diffusion


Routage.

Sur la plupart des réseaux, les ordinateurs sont rarement connectés directement entre eux. Mais cela ne les empêche pas de communiquer, les données pouvant être transmises indirectement, en passant par toute une série d'intermédiaires. Les données sont propagées de proche en proche et doivent trouver leur chemin vers le récepteur, trouver par quels intermédiaires passer pour arriver à destination. Tout le problème est de trouver un chemin d'intermédiaires entre l'émetteur et le récepteur, chemin qui doit de préférence être le plus court et/ou le plus rapide possible.

Cette problématique existe sur Internet et les réseaux étendus, mais elle existe aussi sur des réseaux locaux de grande taille, voire sur certains réseaux locaux très simples (dits en étoile). Sur les réseaux locaux, on lui donne le nom de ***bridging***, alors que l'on parle de **routage** sur les réseaux étendus et sur Internet. Les intermédiaires/relais portent d'ailleurs des noms différents : routeurs pour les relais Internet, commutateurs/concentrateurs pour les relais sur les réseaux locaux.

> Dans la suite de la section, nous ferons la confusion entre routage et *bridging*, que nous désignerons tous deux sous le terme de routage. Cet abus de langage nous permettra de simplifier les formulations et l'écriture. Dans les faits, tout ce qui va être dit vaudra autant pour le routage sur les réseaux étendus que sur les réseaux locaux.

L'adressage réseau : adresses MAC et IP

Tout le problème du routage est de décider quand la donnée a atteint sa destination. Pour cela, il faut que les relais aient des moyens d'identifier les ordinateurs, par un système quelconque. Pour comprendre comment ce problème est résolu, nous allons faire une analogie entre un réseau et le système postal. Dans cette analogie, les personnes qui utilisent la poste sont équivalent aux ordinateurs d'un réseau, et les lettres sont équivalentes à des paquets de données échangés. Quand quelqu'un envoie une lettre, les relais postiers doivent identifier le destinataire, sans quoi ils ne savent où distribuer le courrier. Pour cela, on utilise une adresse, qui indique où il faut livrer la lettre. Sur les réseaux, on utilise le même mécanisme : chaque ordinateur reçoit une adresse, un identifiant qui permet de l'identifier. Mais les adresses réseaux sont bien plus simples que les adresses postales : elle n'ont pas d'organisation géographique et sont de simples numéros attribués arbitrairement.

On pourrait aussi faire une comparaison entre réseau téléphonique et réseau informatique, les deux étant très similaires et faisant usage de technologies très proches. Dans cette analogie, les ordinateurs sont équivalents aux téléphones, les données sont remplacées par des conversations téléphoniques, et les adresses par des numéros de téléphone.

Analogie entre réseau postal, réseau téléphonique et réseau informatique

Réseau informatique	Réseau postal	Réseau téléphonique
Ordinateurs	Foyers/maisons/appartements	Téléphones
Adresse réseau	Adresse postale	Numéro de téléphone
Donnée échangée	Lettre	Conversation téléphonique
Routeurs et commutateurs	Relais de distribution postaux	Relais téléphoniques

Précisons qu'il existe deux types d'adresses : les adresses physiques et les adresses logiques. Les premières sont des adresses utilisées sur les réseaux locaux, mais qui ne sont pas compatibles avec les réseaux étendu et Internet. Ce sont des adresses qui ne peuvent pas être utilisées pour identifier un ordinateur sur le net et dont la portée est limitée à un réseau local, guère plus. À l'inverse, les adresses logiques sont utilisées sur Internet et ont une portée très large, dépassant le réseau local de l'ordinateur. Les adresses physiques sont actuellement standardisées par le standard MAC, ce qui leur vaut le nom d'adresses MAC. Quant aux adresses logiques, elles sont standardisées par le protocole IP, ce qui leur vaut le nom d'adresses IP. Une telle séparation a son utilité : elle permet le remplacement d'un ordinateur sans pour autant changer son adresse internet. Par exemple, si un serveur tombe en panne et que l'on le remplace, il garde son adresse IP, alors que son adresse MAC change.

Routage par datagrammes et par circuits virtuels

Toute la problématique du routage est que les données envoyées arrivent à destination. Dans les grandes lignes, il existe deux types de routage distincts, qui fonctionnent sur des principes très différents : la méthode des **datagrammes** et la méthode des **circuits virtuels**.

> Sauf exceptions, les données envoyées sur un réseau le sont par blocs de taille fixe. Dans ce qui suit, nous allons appeler ces blocs de données des *paquets* (terme quelque peu impropre, mais passons).

Avec les **datagrammes**, chaque paquet contient toutes les informations nécessaires pour identifier le destinataire. Chaque paquet contient l'adresse du destinataire et éventuellement l'adresse de l'émetteur. Le principal avantage de cette méthode est qu'elle ne nécessite pas de maintenir une connexion entre source et destinataire. Le paquet est envoyé sans concertation préalable avec le destinataire. Pour savoir où envoyer les paquets reçus, les relais du réseau contiennent une table qui associe chaque adresse avec le prochain intermédiaire. Cette table porte des noms différents selon que l'on parle de routeurs ou de commutateurs : table de routage pour les routeurs et table CAM pour les commutateurs.

La **méthode des circuits virtuels** nécessite de maintenir une connexion active entre source et destinataire. Une fois la connexion établie, tout paquet émis par l'émetteur est envoyé automatiquement au récepteur désigné lors de la connexion. Avec elle, les paquets n'ont pas besoin de préciser leur destination : pas besoin d'insérer l'adresse de destination dans le paquet. Le destinataire est l'ordinateur avec lequel est connecté l'émetteur.

Les modes de transfert

Une transmission sur le réseau peut prendre différentes formes suivant le nombre de destinataires. Un ordinateur peut en effet vouloir communiquer avec un ordinateur bien précis, ou envoyer une donnée à plusieurs PC différents. Suivant le nombre de destinataire, on peut faire la différence entre Unicast, Anycast, Multicast et Broadcast.

Avec l'**unicast**, un ordinateur émet des données à destination d'un autre ordinateur bien identifié.

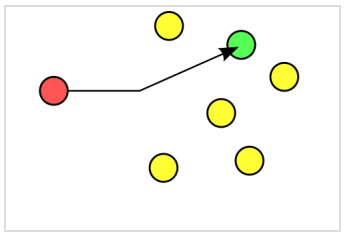
Avec l'**anycast**, un ordinateur émet des données vers un ordinateur qu'il ne connaît pas : l'émetteur ne connaît pas la destination de la donnée. L'ordinateur de destination n'est cependant pas choisi au hasard : c'est le protocole de routage qui choisit vers quel ordinateur émettre la donnée.

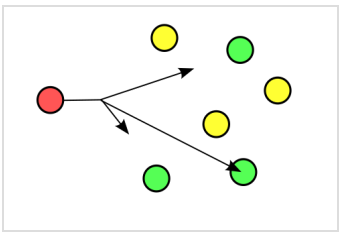
Avec le **multicast**, les données émises sont envoyées à un groupe d'ordinateur qui veulent recevoir cette donnée. Les ordinateurs qui veulent revoir la donnée se connectent à un serveur et s'inscrivent à un groupe de diffusion. Tous les ordinateur inscrits dans ce groupe recevront la donnée émise. C'est notamment utilisé lors du streaming d'évènements en live : on émet la donnée une fois, et celle-ci sera recopiée par les routeurs à toutes les personnes inscrites au groupe que le routeur connaît. Pour faire simple, le groupe possède une adresse logique (une IP) qui permet de l'identifier. Quand une donnée est envoyée à l'adresse du groupe, le serveur reçoit le paquet et en envoie des copies à tous les ordinateurs du groupe. L'adresse IP du serveur/groupe est appelée une adresse multicast.

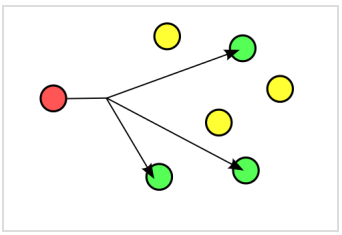
Avec le **broadcast**, le paquet émis est envoyé à tous les ordinateurs d'un réseau ou sous-réseau (un réseau local le plus souvent). Selon que le paquet se propage dans un réseau local ou sur internet, on distingue deux formes de *broadcast*.

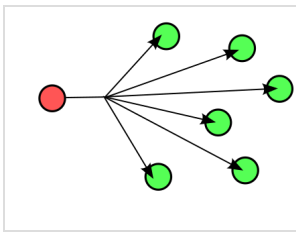
- Le *broadcast* limité a une portée limitée au réseau local de l'émetteur. Le paquet est envoyé aux voisins de l'ordinateur émetteur, mais reste confiné dans le réseau local : il ne passe pas les routeurs, mais est propagé par les commutateurs/concentrateurs.
- Le *broadcast* dirigé est similaire au précédent, sauf que le paquet n'est pas confiné dans un réseau local et peut passer les routeurs pour se déplacer

sur internet.


Unicast


Anycast


Multicast


Broadcast

Précisons une chose sur le *broadcast :* qu'il soit dirigé ou non, toute transmission en *broadcast* va se propager et atteindre un certain nombre d'ordinateurs. L'ensemble des ordinateurs d'un réseau local, pour un *broadcast* limité, un ensemble d'ordinateurs bien précis sur Internet (*broadcast* dirigé). L'ensemble des ordinateurs qui reçoit une transmission *broadcast* porte un nom : c'est le **domaine de diffusion**.

Les topologies logiques

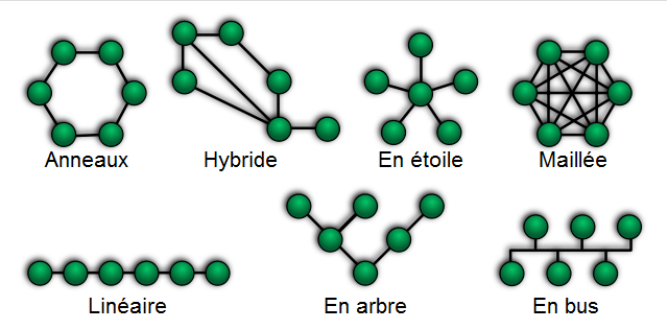
> *Dans le contenu suivant, nous parlerons de **nœuds** pour désigner tout ce qui est connecté au réseau. Ce terme regroupe un peu de tout : ordinateurs, imprimantes, serveurs, routeurs, commutateurs, et bien d'autres matériels réseau. Nous utiliserons cette dénomination pour alléger l'écriture.*

Il n'existe pas 36 façons différentes de relier les ordinateurs entre eux dans un réseau et chaque manière a ses avantages et inconvénients. Afin de s'y retrouver, les réseaux peuvent également être catégorisés par la manière dont les nœuds sont reliés dans un réseau. La connectivité des nœuds porte un nom : c'est la **topologie du réseau**. Formellement, la notion de topologie peut s'appliquer à tout type de réseaux, y compris des réseaux MAN, WAN, ou plus large encore. Mais force est de constater que l'on parle rarement de la topologie des réseaux étendus, qui est souvent très complexe. Par contre, la topologie des réseaux locaux est d'une grande importance.

Les types de topologies

Il existe différents types de topologies, mais les principales sont illustrées dans le schéma ci-dessous. On voit que les topologies possibles s'appellent : la topologie en bus, en anneau, en arbre, linéaire, maillée (totalement ou partiellement), en étoile et hybride (un mélange des précédentes).

- La **topologie en bus** correspond au cas où tous les nœuds sont connectés à un unique fil, à un unique support de transmission matériel.
- La **topologie totalement maillée** est celle où tous les ordinateurs sont reliés à tous les autres. Le nombre de liens est alors important (proportionnel au carré du nombre d'ordinateurs à relier).
- La **topologie hybride** est équivalent à une topologie maillée à laquelle on aurait retiré quelques liaisons point-à-point, pour économiser des liens sans trop diminuer la performance du réseau.
- La **topologie en étoile** est celle où les nœuds sont reliés à un équipement réseau central. Le nombre de liens est proportionnel au nombre d'ordinateurs du réseau, ce qui est peu. Le nombre d'intermédiaire est faible (seulement un), ce qui fait que le matériel réseau nécessaire pour créer un réseau en étoile est particulièrement simple et peu cher. La moindre panne de l'équipement central fait tomber tout le réseau : les ordinateurs ne peuvent plus communiquer entre eux.
- La **topologie linéaire** est celle en forme de chaine. Tous les ordinateurs sont reliés à deux voisins, sauf les deux nœuds au bout de la chaine.
- La **topologie en anneau** est celle en forme d'anneau. Tous les ordinateurs sont reliés à deux voisins, sans exception. Elle est équivalente à une topologie linéaire dont on aurait relié entre eux les deux bouts.
- En théorie, les réseaux sans-fils sont regroupés dans une catégorie à part, du fait de l'inexistence de liaisons point-à-point sur ces réseaux).



Topologies possibles des réseaux (la topologie sans-fil n'est pas représentée).

Chacune a des avantages et inconvénients. Par exemple, certaines utiliseront beaucoup de liaisons point-à-point, alors que d'autres seront relativement économes : il faudra prévoir plus ou moins de câbles, de fils, et d'équipement réseau suivant la topologie. De plus, certaines topologies ne permettent pas à deux nœuds de communiquer directement. Les données doivent alors passer par des nœuds intermédiaires et se propager dans le réseau de proche en proche. Une conséquence assez fâcheuse est que la panne de certains nœuds peut empêcher des ordinateurs de communiquer. De plus, plus le nombre d'intermédiaires entre deux ordinateurs est grand, plus les performances seront mauvaises : le temps de transmission augmente avec le nombre d'intermédiaires. Autant cela n'a pas grande conséquence sur des réseaux locaux, autant cela devient plus problématique dans le cas de grands réseaux, tel certains WAN ou pour Internet.

Topologie	Nombre de câbles (liaisons point-à-point), pour N ordinateurs	Résistance aux pannes
Topologie en bus	Un seul.	Toute coupure du câble scinde le réseau en deux sous-réseaux indépendants, voire fait tomber tout le réseau.
Topologie linéaire	N liaisons point-à-point (autant que de nœuds)	Toute coupure d'une liaison point-à-point scinde le réseau en deux sous-réseaux indépendants, voire fait tomber tout le réseau.
Topologie en anneau	N liaisons point-à-point (autant que de nœuds)	Dépend du réseau en question (voir la section sur le sujet plus bas)
Topologie en étoile	N liaisons point-à-point (autant que de nœuds)	Toute panne du nœud central fait tomber tout le réseau.
Topologie totalement maillée	$\frac{N(N+1)}{2}$ liaisons point à point.	Résistance maximale à toute coupure de liaisons point-à-point ou de panne d'ordinateur.

Généralement, les réseaux locaux utilisent des topologies en étoile, en bus ou en anneau, rarement en arbre. Les contraintes en terme de câblage font que les topologies maillées et hybrides ne sont pas très faciles à mettre en œuvre, du fait de leur grand nombre de liaisons point à point. Les réseaux étendus et Internet ne peuvent utiliser ces topologies, le nombre d'équipements et d'ordinateur étant bien trop important. Les réseaux WAN, MAN et Internet sont donc des réseaux maillés, de type hybride.

Les topologies diffusantes

Nous allons commencer par voir deux types de topologies : la topologie en bus, et les réseaux sans-fils. Si on regroupe ces deux topologies ensemble, c'est parce qu'elles ont une particularité assez intéressante : quand une trame est envoyée, toutes les machines du réseau reçoivent la trame, mais seul l'ordinateur de destination est censé la prendre en compte. Cela leur vaut d'être appelés des **réseaux de diffusion**, sous-entendu que les données sont diffusées à tous les nœuds (un peu comme une diffusion radio, qui est diffusée sur toute une région, mais n'est écoutée que par quelques personnes). C'est une propriété naturelle pour les bus et les réseaux sans fils, qui font communiquer plusieurs nœuds avec un même support partagé (le bus ou les ondes sans-fils). Les autres réseaux n'ont pas cette particularité : la donnée peut passer par des intermédiaires, ou arriver directement à destination, mais en aucun cas la donnée n'est diffusée à tous les ordinateurs du réseau.

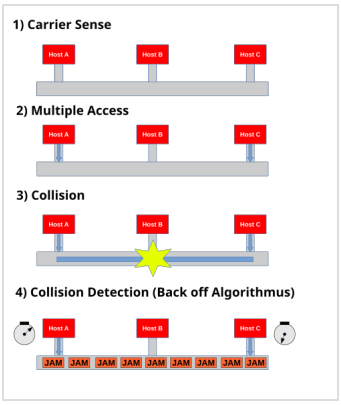


Illustration d'une collision dans un réseau diffusant (ici, un bus).

Un problème des réseaux de diffusion est que la sécurité n'est pas optimale. Tous les PC recevant les informations transmises, rien n'empêche un ordinateur (ou équipement réseau) d'espionner ce qui est transmis sur le réseau local : il lui suffit de prendre en compte la totalité des paquets qui sont envoyés sur le réseau. Un tel ordinateur est dit en mode « *promiscious* ». Il suffit d'installer un logiciel de capture de trames pour pouvoir surveiller le trafic réseau. Un tel logiciels est utile pour vérifier l'occurrence d'une intrusion ou infection, surtout quand il est couplé à un IDS (logiciel de détection d'intrusions) qui analyse les trames capturées.

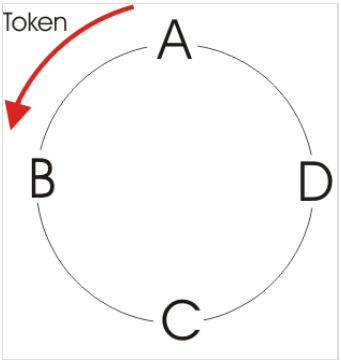
Outre le problème précédent, il arrive que plusieurs composants tentent d'envoyer ou de recevoir une donnée sur le bus en même temps. Il se produit alors un conflit d'accès au bus, aussi appelé une **collision**. Cela pose problème si un composant cherche à envoyer un 1 et l'autre un 0 : tout ce que l'on reçoit à l'autre bout du fil est une espèce de mélange incohérent des deux données envoyées sur le bus par les deux composants. Détecter l'occurrence d'une collision et la corriger est essentiel pour qu'un bus fonctionne correctement. Inutile par construction pour les liaisons point à point, la gestion des collisions est essentielle pour tout réseau de diffusion. C'est pour cela que les protocoles des réseaux de diffusion gèrent ces collisions, et s'occupent du partage de l'accès au bus, d'où leur nom de **protocoles à accès multiples**. Diverses méthodes ont été inventées et intégrées aux protocoles les plus connus et nous verrons le fameux CSMA-CD plus tard dans ce cours.

La topologie en anneau

Dans les **réseaux en anneau**, les ordinateurs sont chacun relié à deux autres : un suivant et un précédent. Les données transmises font le tour de l'anneau avant d'être détruites : elles se propagent d'un ordinateur au suivant, jusqu'à arriver sur l'ordinateur de destination. Après réception de la trame, l'ordinateur de destination envoie un accusé de réception à l'émetteur, qui se propage dans le même sens que la trame envoyée.

Les jetons de contrôle d'accès

Les réseaux en anneaux ont besoin d'un système de contrôle d'accès. Sans cela, on pourrait observer des phénomènes similaires aux collisions des réseaux diffusants. Pour éviter ceux-ci, chaque ordinateur devient émetteur a tour de rôle. Le droit d'émettre une donnée est modélisé par un jeton, une trame spéciale, pour laquelle un bit pré-déterminé est placé à 1. Il passe d'un ordinateur au suivant, à tour de rôle, et balaye tout le réseau en anneau régulièrement. Tant que l'ordinateur garde le jeton, il a le droit d'émettre une donnée et d'attendre un accusé de réception. D'ordinaire, le jeton est conservé dans un temps déterminé et fixe, le même pour tous les ordinateurs. Une exception, cependant : un ordinateur qui n'a rien à envoyer transmet immédiatement le jeton à l'ordinateur suivant.

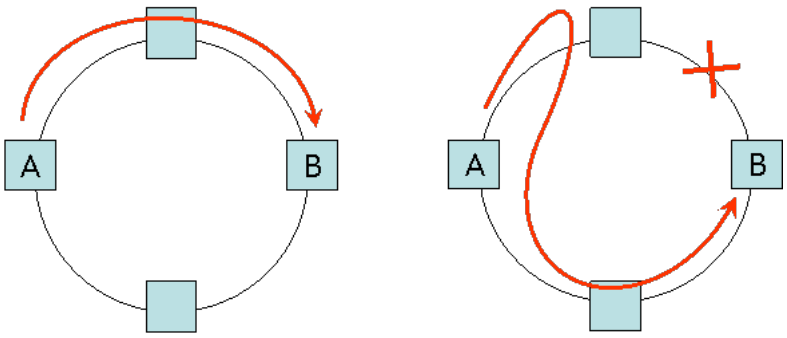


Propagation du jeton (Token) dans un réseau en anneau.

L'utilisation d'un jeton permet à tous les ordinateurs d'avoir un accès équitable au réseau. Il garantit que tous les ordinateurs aient accès au réseau à un moment ou un autre. De plus, le temps avant de récupérer le jeton est obligatoirement fini. Le temps d'attente maximal est le temps que le jeton fasse le tour de l'anneau. C'est un avantage que les bus n'ont pas : un ordinateur peut potentiellement monopoliser le bus durant un temps indéterminé si rien n'est prévu dans la conception du bus. La conséquence est que les réseaux en anneau ont un meilleur débit si le nombre d'ordinateur du réseau est assez important. L'occurrence de nombreuses collision réduit le débit utile du bus, alors que les réseaux en anneau n'ont pas ce problème.

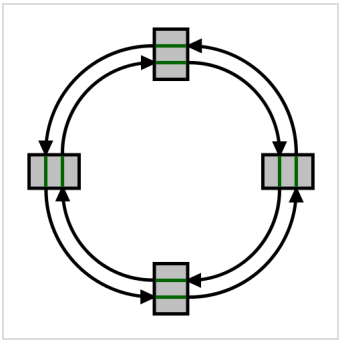
Les topologies en anneaux simples et doubles

Avec un anneau dit "simple", la panne d'un seul ordinateur met tout le réseau en panne, vu que l'ordinateur fautif ne peut plus propager les trames. Si un câble est coupé, les données ne peuvent pas atteindre l'ordinateur par le sens de propagation usuel. Mais elles le peuvent en passant dans l'autre sens. Le schéma ci-dessous illustre cette idée.

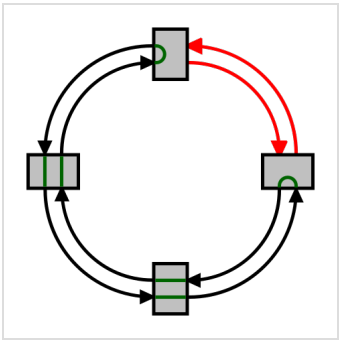


Les concepteurs de réseaux ont depuis longtemps trouvé comment propager les données dans les deux sens. Il suffit d'ajouter un second anneau qui propage les trames dans l'autre sens. Le réseau en anneau est donc composé de deux anneaux, qui propagent les jetons et trames dans des sens différents. Quand un câble est coupé, l'ordinateur ne va pas envoyer les trames sur celui-ci, mais va les renvoyer dans le sens inverse, sur l'autre anneau. Ce faisant,

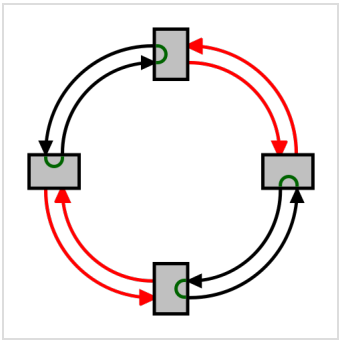
tout se passe comme si le premier anneau bouclait sur le second quand le câble est coupé. Si un seul câble est coupé, le réseau reste le même . Mais si deux câbles sont coupés, les choses deviennent plus compliquées : tout se passe comme si le réseau étant coupé en plusieurs sous-réseaux qui ne peuvent pas communiquer entre eux. Les schémas plus bas illustrent ces concepts basiques de cet anneau « auto-régénérateur » (*Self-healing Ring* en anglais).



Anneau auto-régénérateur intact.



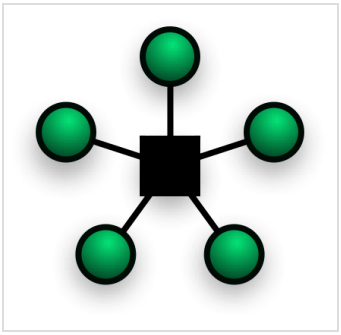
Anneau auto-régénérateur avec un lien endommagé.



Anneau auto-régénérateur avec deux liens endommagés.

Les topologies logiques

On a vu dans le chapitre précédent comment plusieurs ordinateurs peuvent échanger des données via un réseau local, ce qui est le principal problème de la couche liaison. Nous nous sommes attardés sur les réseaux de diffusion et les liaisons point à point dans le chapitre précédent et il est temps de parler plus en détail des réseaux en étoile. Pour rappel, cela signifie que les ordinateurs du réseau sont reliés à un équipement central, qui reçoit les trames et les redistribue vers les autres ordinateurs. Cette topologie en réseau est une topologie physique, ce qui veut dire qu'elle décrit comment les ordinateurs sont physiquement reliés. Mais il est possible d'émuler un réseau en bus, en anneau, ou un réseau maillé avec une topologie en étoile. Cela nous pousse à faire la distinction entre la topologie physique, qui décrit comment les ordinateurs sont câblés, et la topologie simulée, appelée topologie logique.



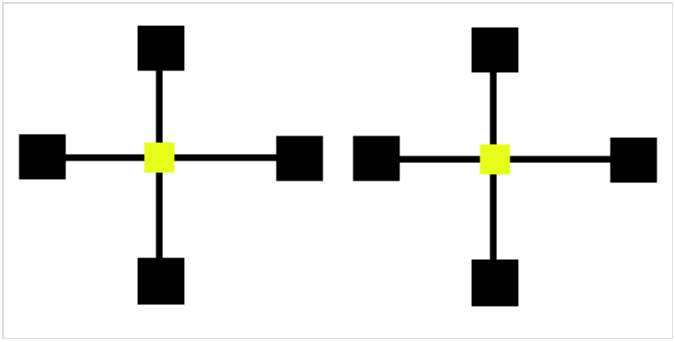
Topologie en étoile.

Les topologies logiques en bus et maillées

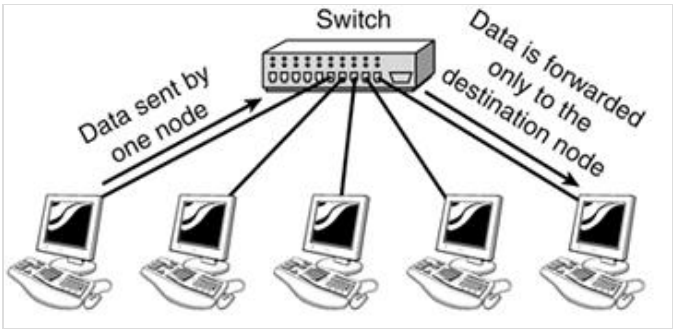
Pour simuler un bus ou un réseau maillé à partir d'une topologie en étoile, il faut que l'équipement central soit ce qu'on appelle un commutateur ou un concentrateur. Ils sont tous deux des équipements réseaux avec plusieurs ports d'entrée/sortie, sur lesquels on vient connecter des composants réseaux : carte réseau, ordinateur, récepteur/émetteur WIFI, etc. Ces ports sont soit des ports d'entrées sur lesquels on peut recevoir des paquets de données, soit des ports de sorties sur lesquels on peut envoyer des données. Dans certains cas, les ports d'entrée et de sortie sont confondus : un même port peut servir alternativement d'entrée et de sortie.

La différence entre concentrateur et commutateur est la topologie simulée : bus pour le concentrateur et réseau maillé pour un commutateur.

- Un **concentrateur** redistribue chaque paquet reçu sur tous les autres ports, sans se préoccuper de sa destination. Pour le dire autrement, il simule une topologie en bus, alors que la topologie réelle est en étoile.
- Les **commutateurs** ont un fonctionnement similaire aux concentrateurs, si ce n'est qu'ils n'envoient les données qu'au composant de destination. Un commutateur simule donc une topologie totalement maillée à partir d'une topologie en étoile : on retrouve la distinction entre topologie réelle et logique.



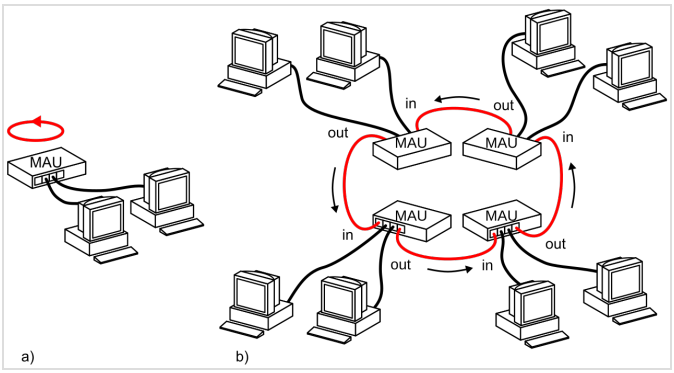
Différence entre concentrateur (à gauche) et commutateur (à droite).



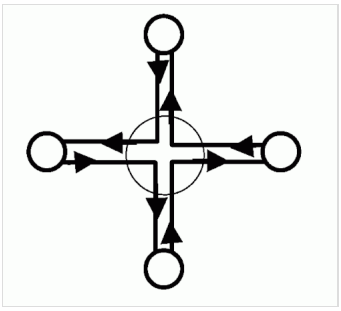
Fonctionnement d'un commutateur.

Les réseaux logiques en anneau

Il est possible de simuler un réseau en anneau avec un réseau en étoile, comme pour les réseaux en bus et les réseaux maillés. Cette fois-ci, l'équipement central n'est pas un concentrateur ou un commutateur, mais ce qu'on appelle une *Multistation Access Unit* (*MAU*). On connecte plusieurs ordinateurs sur ce boîtier, et ceux-ci seront alors reliés en anneau. On peut considérer que ce boîtier est un équivalent des concentrateurs et commutateurs, mais pour les réseaux en anneau.



Réseau Token ring.



Token ring avec un équipement central (MAU).

Les modèles OSI et TCP

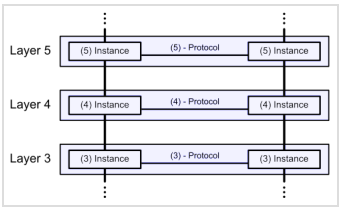
Les ordinateurs d'un réseau n'ont pas à être identiques : les différences de systèmes d'exploitation, de logiciels utilisés pour naviguer sur le net, et les autres différences du genre ne doivent pas avoir le moindre impact sur la communication entre deux machines. Par exemple, on peut parfaitement connecter des ordinateurs sous Windows avec des ordinateurs sous Linux. Pour cela, il a fallu inventer un certain nombre de standards réseau, appelés **protocoles réseaux**. Des organismes publics supranationaux s'occupent d'établir et de normaliser ces protocoles : ils consultent les grandes entreprises du web (Microsoft, Apple, et ainsi de suite), et les négociations sur ce qui doit être mis dans les protocoles sont souvent longues. Un cours sur le réseau ne peut décemment pas passer sous silence le fonctionnement des protocoles les plus connus. Peut-être en avez-vous déjà entendu parler : les protocoles IP ou TCP sont de loin les plus connus, sans compter les protocoles UDP ou MAC.

Chaque protocole gère des choses aux noms bizarres, comme le routage, la transmission des bits, l'adressage logique ou physique, et bien d'autres choses que nous allons aborder dans ce cours. Cependant, tous les protocoles ne se situent pas au même "niveau d'abstraction". Standardiser le codage des bits dans un câble réseau est plus proche de la machine que la gestion du routage. On pourrait croire que cette histoire assez vague de niveaux d'abstraction n'a pas grande importance. Il se trouve qu'il existe plusieurs classifications qui décrivent ces niveaux d'abstraction, et que celles-ci sont assez importantes. Ces classifications sont au nombre de deux : le **modèle OSI** et le **modèle TCP/IP**. Ces deux modèles forment l'ossature des chapitres qui vont suivre, aussi il est important de les voir. Une bonne présentation ne peut pas, en effet, mélanger des protocoles ou des connaissances de niveaux d'abstraction différents : nous n'allons pas passer du codage des bits sur un fil de cuivre aux protocoles de routage dans le même chapitre. Les modèles OSI et TCP/IP peuvent sembler assez compliqués, ceux qui les ont appris en cours de réseau ayant certainement eu du mal à comprendre leur principe. Mais il s'agit cependant d'un passage incontournable de la plupart des cours de réseau.

Les modèles OSI et TCP/IP

Ces modèles OSI et TCP/IP permettent de classer divers **protocoles réseaux**, à savoir des standards qui décrivent telle ou telle fonctionnalité que le réseau doit respecter. Par exemple, un protocole de la couche liaison va standardiser la façon dont deux ordinateurs vont s'échanger des données sur un câble réseau : comment les bits sont codés, comment détecter les erreurs de transmission, quelles sont les spécifications électriques des connecteurs et interfaces, etc. Ces protocoles sont classés selon leurs niveaux d'abstractions, dans ce qu'on appelle des **couches**. La définition d'une couche réseau est assez abstraite, mais on peut dire qu'il s'agit d'un ensemble de protocoles, qui sont au même niveau d'abstraction, qui ont des fonctions similaires.

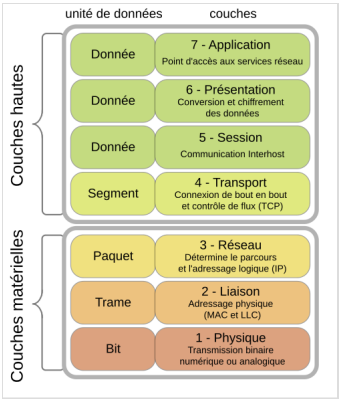
Ces couches sont censées être bien individualisées, à savoir que la communication entre couches doit suivre un certain ordre. Des données à transmettre sur le réseau doivent d'abord passer par les protocoles de la couche la plus abstraite, dans le niveau d'abstraction supérieur, avant de passer par la couche d'en dessous, et ainsi de suite. Cela ne fait que traduire une évidence : avant de traiter le codage des bits du paquet, encore faut-il que les données aient été cryptées, que l'on ait identifié l'émetteur et le récepteur et ainsi de suite. Les opérations à effectuer doivent être faites dans un certain ordre si l'on veut que tout fonctionne, ce que traduit l'ordre des couches. Pour des raisons de simplicité, un protocole d'une couche peut utiliser les fonctionnalités de la couche immédiatement inférieure, mais pas des autres couches. Cela garantit qu'un paquet de données soit traité par toutes les couches dans le bon ordre, sans passer d'étapes.



Communication entre couches.

Le modèle OSI

Le modèle OSI est de loin le plus complet. Il décrit sept couches portant les noms de couche physique, liaison, réseau, transport, session, présentation et application. Les divers protocoles qui définissent le réseau et les communications sont donc répartis dans chaque couche, selon leur utilité. Il est d'usage de diviser ces sept couches en deux : les couches basses, qui se limitent à gérer des fonctionnalités de base, et les couches hautes, qui contiennent les protocoles plus élaborés.



Représentation du modèle OSI en français

Les *couches basses*, aussi appelées *couches matérielles*, s'occupent de tout ce qui a trait au bas-niveau, au matériel. Elles permettent d'envoyer un paquet de données sur un réseau et garantir que celui-ci arrive à destination. Elle est généralement prise en charge par le matériel et le système d'exploitation, mais pas du tout par les logiciels réseaux. Les couches basses sont donc des couches assez bas-niveau, peu abstraites. Les couches basses sont au nombre de trois. Pour résumer, ces trois couches s'occupent respectivement des liaisons point à point (entre deux ordinateurs/équipements réseaux), des réseaux locaux, et des réseaux Internet.

- La **couche physique** s'occupe de la transmission physique des bits entre deux équipements réseaux. Elle s'occupe de la transmission des bits, leur encodage, la synchronisation entre deux cartes réseau, etc. Elle définit les standards des câbles réseaux, des fils de cuivre, du WIFI, de la fibre optique, ou de tout autre support électronique de transmission.
- La **couche liaison** s'occupe de la transmission d'un flux de bits entre deux ordinateurs, par l'intermédiaire d'une liaison point à point ou d'un bus. Pour simplifier, elle s'occupe de la gestion du réseau local. Elle prend notamment en charge les protocoles MAC, ARP, et quelques autres.
- La **couche réseau** s'occupe de tout ce qui a trait à internet : l'identification des différents réseaux à interconnecter, la spécification des transferts de données entre réseaux, leur synchronisation, etc. C'est notamment cette couche qui s'occupe du routage, à savoir la découverte d'un chemin de transmission entre récepteur et émetteur, chemin qui passe par une série de machines ou de routeurs qui transmettent l'information de proche en proche. Le protocole principal de cette couche est le protocole IP.

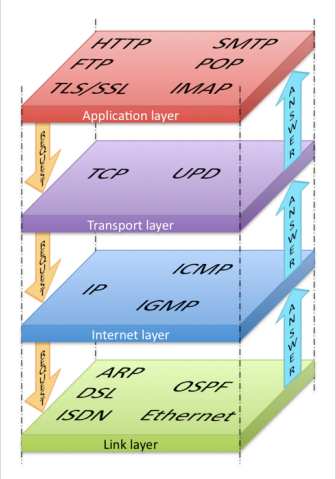
Les *couches hautes*, aussi appelées *couches logicielles*, contiennent des protocoles pour simplifier la programmation logicielle. Elles requièrent généralement que deux programmes communiquent entre eux sur le réseau. Elles sont implémentées par des bibliothèques logicielles ou directement dans divers logiciels. Le système d'exploitation ne doit pas, en général, implémenter les protocoles des couches hautes. Elles sont au nombre de quatre :

- La **couche transport** permet de gérer la communication entre deux programmes, deux processus. Les deux protocoles de cette couche sont les protocoles TCP et UDP.

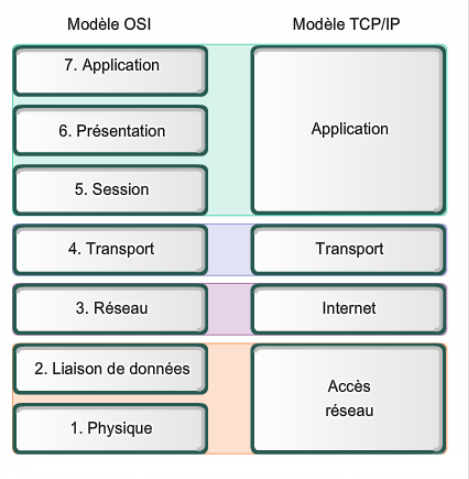
- La **couche session**, comme son nom l'indique, permet de gérer les connexions et déconnexions et la synchronisation entre deux processus.
- La **couche présentation** se charge du codage des données à transmettre. Elle s'occupe notamment des conversions de boutisme ou d'alignement, mais aussi du chiffrement ou de la compression des données transmises.
- La **couche application** prend en charge tout le reste.

Modèle TCP/IP

Le modèle TCP/IP est plus simple qu'OSI, avec seulement quatre couches : liaison, Internet, transport et application. La différence avec OSI est simplement que certaines couches ont été fusionnées. La couche liaison de TCP/IP regroupe notamment les couches physiques et liaison d'OSI. De même, la couche application de TCP/IP regroupe les couches session, application et présentation d'OSI.



InternetProtocolStack



Comparaison des modèles OSI et TCP IP

L'encapsulation

Les données ne sont pas transmises telles quelles sur le réseau, car chaque protocole a besoin d'informations bien précises pour faire son travail. Par exemple, les protocoles de couche liaison ont besoin d'informations particulières, non présentes dans la donnée transmise, pour détecter les erreurs ou indiquer le récepteur. Même chose pour les protocoles TCP et UDP de la couche transport, qui ont besoin d'informations sur le processus émetteur et récepteur, qui ne sont pas dans la donnée transmise. Et ce problème nous amène à parler de l'**encapsulation**.

Les en-têtes des paquets

Pour résoudre le problème précédent, chaque protocole ajoute les informations dont il a besoin à la donnée transmise. Ces informations sont regroupées dans un **en-tête**, placé au début des données à transmettre. Plus rarement, certains protocoles ajoutent leurs informations devant, mais aussi derrière la donnée à transmettre : l'en-tête est complété par un *pied* (le terme anglais est : *footer*). Lorsqu'un protocole prend en charge un paquet de données, il lui ajoute un en-tête à son début. Avec cette méthode, les en-têtes de chaque couche sont séparés, placés les uns à côté des autres. Lors de la réception, cet en-tête sera enlevé : les couches supérieures n'ont pas besoin des en-têtes des couches inférieures.

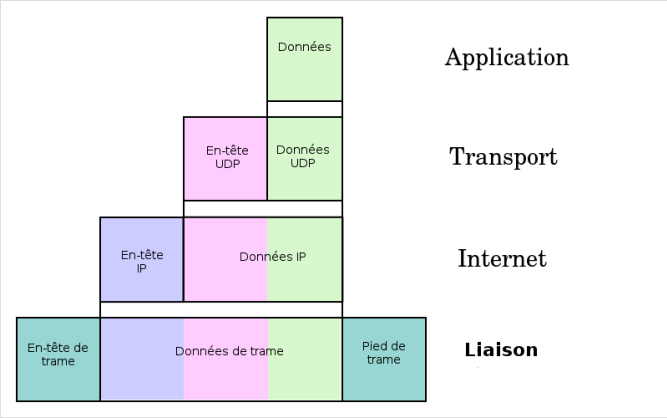


Illustration de l'encapsulation en fonction des couches TCP/IP.

Les *Protocol Data Unit*

Si la couche physique s'occupe de la transmission de bits individuels, ce n'est pas le cas des autres couches, qui traitent des paquets (données + en-tête/pied) contenant plusieurs bits. Les couches réseau, liaison et transport traitent des paquets de données de taille fixe (leur nombre de bits est fixé). Au-delà de la couche transport, les données ne sont pas structurées en paquets de taille fixe, mais en paquets de taille variable, qui dépendent du logiciel utilisé. Pour les couches liaison, réseau et transport, les paquets sont appelés des ***protocol data unit*** (PDU, unités de données d'un protocole, en français). Ils portent des noms différents selon la couche, vu qu'ils contiennent des en-têtes différents.

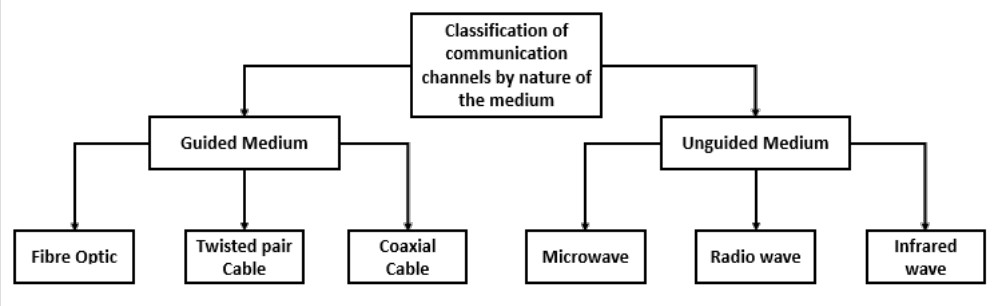
- Pour la couche liaison, l'unité est la *trame*.
- Pour la couche réseau, l'unité est le *paquet*.
- Pour la couche transport, l'unité est le *segment* pour le protocole TCP et le *datagramme* pour le protocole UDP.

<table>
<tr><th>Couche OSI</th><th colspan="2">Unité de transfert de données (PDU)</th><th>Taille du PDU</th></tr>
<tr><td>Physique</td><td colspan="2">Bit ou Symbole</td><td>Bit/symbole unique</td></tr>
<tr><td>Liaison</td><td colspan="2">Trame</td><td rowspan="3">Taille fixe, plusieurs bits, dépend du protocole utilisé.</td></tr>
<tr><td>Réseau</td><td colspan="2">Paquet</td></tr>
<tr><td>Transport</td><td>Segment (protocole TCP)</td><td>Datagramme (protocole UDP)</td></tr>
<tr><td>Session</td><td colspan="3" rowspan="3">Paquets de taille variable</td></tr>
<tr><td>Présentation</td></tr>
<tr><td>Application</td></tr>
</table>

Les supports de transmission

Pour que deux ordinateurs ou équipements réseau communiquent entre eux, il faut qu'ils soient reliés par quelque chose qui leur permet de transmettre de l'information. Ce quelque chose est ce qu'on appelle un **support de transmission**, qui est souvent un simple câble réseau, composé d'un fil de cuivre ou de fibre optique. Dans d'autres cas, la transmission se fait sans fils, avec des technologies à base d'infrarouges, d'ondes radio ou de micro-ondes. On pourrait notamment citer le WIFI, le Bluetooth, et bien d'autres. Pour résumer, il existe deux types de supports de communication : les câbles réseaux et le sans-fils.

Il faut cependant noter que les spécialistes en électricité et en électronique ont cependant un jargon un peu plus complexe que "avec fils" et "sans-fils". Ils parlent à la place de liaison guidée (avec fils) et non-guidée (sans fils). Dans les deux cas, le support de transmission va propager des ondes électromagnétiques, qui codent les informations transmises. Si le support est guidé, les ondes électromagnétiques ne pourront pas se disperser et seront contenues dans un espace restreint. Leur direction de propagation sera quelque peu contrôlée de manière à ce qu'elles aillent vers la destination et uniquement celle-ci. C'est le cas pour les fibres optiques ou les câbles réseaux : l'onde électromagnétique ne sort pas du câble et y reste confinée. Avec un support de transmission non-guidé, les ondes électromagnétiques vont se propager dans toutes les directions : elles ne seront pas guidées ou confinées dans un câble et seront émises dans toutes les directions à partir de la source. C'est le cas des ondes WIFI, Bluetooth et de toutes les technologies sans-fils en général.

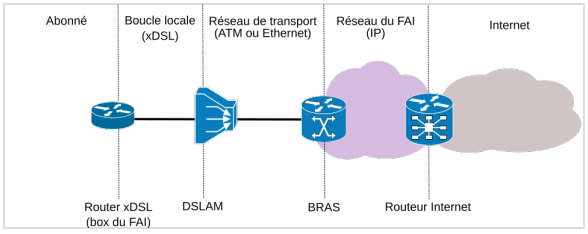


Classification des supports de transmission.

Chaque support a ses avantages et inconvénients, le plus important étant sa portée. Les supports sans-fils n'émettent pas au-delà d'une certaine distance, au-delà de laquelle le signal transmis est trop atténué pour être capté. On observe un phénomène similaire sur les câbles réseaux, qu'ils soient en cuivre ou en fibre optique, mais il est plus sensible sur les câbles en fils de cuivre, raison pour laquelle les fibres optiques sont surtout utilisées pour de grandes distances. De manière générale, les technologies sans-fils ont une mauvaise portée : les ondes électromagnétiques vont se disperser dans l'environnement, facilitant leur atténuation. Les supports guidés (les câbles en cuivre et fibres optiques) n'ont pas ce problème : ils guident l'onde électromagnétique, qui ne peut pas s'échapper du câble.

Les supports de transmission guidés

Les câbles réseaux sont de loin la technologie la plus répandue dans nos foyers. Vous le savez peut-être, mais il existe grosso-modo deux types de câbles réseaux : les câbles basés sur des fils de cuivre, et la fibre optique. Peut-être savez-vous même que la fibre optique est bien plus rapide que la paire cuivre. Dans ce qui va suivre, nous allons voir aussi bien la paire cuivre, plus répandue dans nos foyers, que la fibre optique. Gardez à l'esprit que la fibre optique est certes plus puissante, mais aussi plus chère. C'est pour cela que les câbles réseaux qui relient votre ordinateur à votre box internet sont encore des câbles réseaux standards, en cuivre. Il faut savoir que ces câbles sont utilisés aussi bien pour construire des LAN que des réseaux plus importants. Par exemple, ce sont ces câbles qui relient votre box Internet à...Internet. Votre prise électrique est en effet reliée au réseau de votre opérateur par toute une série d'intermédiaires : les DSLAM, les BRAS, etc. Le fil qui relie votre box à ces intermédiaires était autrefois un fil téléphonique en cuivre, aux performances assez mauvaises. De nos jours, de plus en plus de clients passent à la fibre optique, ce qui signifie que le câble téléphonique qui relie votre habitation au DSLAM est en fibre optique, aux performances bien meilleures.

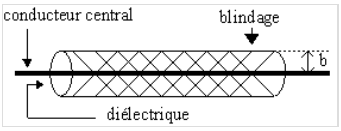


XDSL Connectivity Diagram fr

Les câbles en cuivre

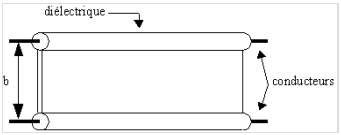
Les câbles réseaux les plus simples sont de simples fils électriques. Ils sont composés d'un fil de conducteur, souvent du cuivre, entouré d'un isolant. Ce câble permet de transmettre n'importe quel signal électrique, qui est codé sois avec une tension, soit avec un courant. De nos jours, la majorité des signaux sont codés avec une tension électrique : on place une tension à un côté du câble, et cette tension est rapidement transmise à l'autre bout. Cette transmission est très rapide, bien plus qu'avec un courant. Cependant, un tel câble est sensible aux perturbations électromagnétiques, très fréquentes dans l'environnement. Si une perturbation électrique modifie la tension dans le fil, elle peut modifier les données transmises. Et il est alors impossible de savoir quelle était la donnée transmise à la base.

Les **câbles coaxiaux** sont composés d'un fil conducteur, entouré d'un isolant, lui-même entouré d'une couche de conducteurs (le blindage), le tout étant enroulé d'une protection isolante. Ces câbles réseaux sont assez performants. Leur débit varie de 56 Kilobits à plusieurs Gigabits. Ils sont utilisés autant pour les réseaux locaux que pour des liaisons à plus grande distance. Par exemple, les câbles réseaux qui font communiquer votre ordinateur et votre box internet sont de ce type. Mais les câbles qui relient votre prise téléphonique aux équipements de votre opérateur sont aussi des câbles coaxiaux (du moins, si vous n'avez pas la fibre). Comme dit auparavant, ces câbles transmettent aussi bien les signaux analogiques que numériques.



Câble coaxial.

Pour éviter l'impact des perturbations, on peut utiliser deux fils pour la transmission, ce qui donne ce qu'on appelle une **ligne bifilaire**. Dans la plupart des cas, les deux fils sont torsadés, enroulés l'un autour de l'autre. On parle alors de **paire torsadée**. Mais certains fils ont une organisation plus complexe.

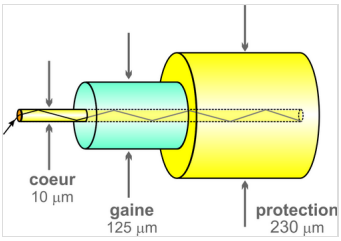


Ligne bifilaire

Schéma d'une paire torsadée.

Les fibres optiques

Les fibres optiques transmettent des signaux par le biais d'impulsions lumineuses. Ces impulsions lumineuses sont émises à un bout de la fibre optique, se propagent dans la fibre optique jusqu'à l'autre bout, où elles sont captées par un récepteur lumineux. La fibre optique est composée d'un cœur de matériau transparent, entouré par une gaine de matériau lui aussi transparent, l'ensemble étant entouré par une couche plastique de protection. Le cœur a un indice de réfraction inférieur à celui de la gaine. En conséquence, la lumière rebondit sur les parois de la gaine, et se propage dans le cœur par rebonds.



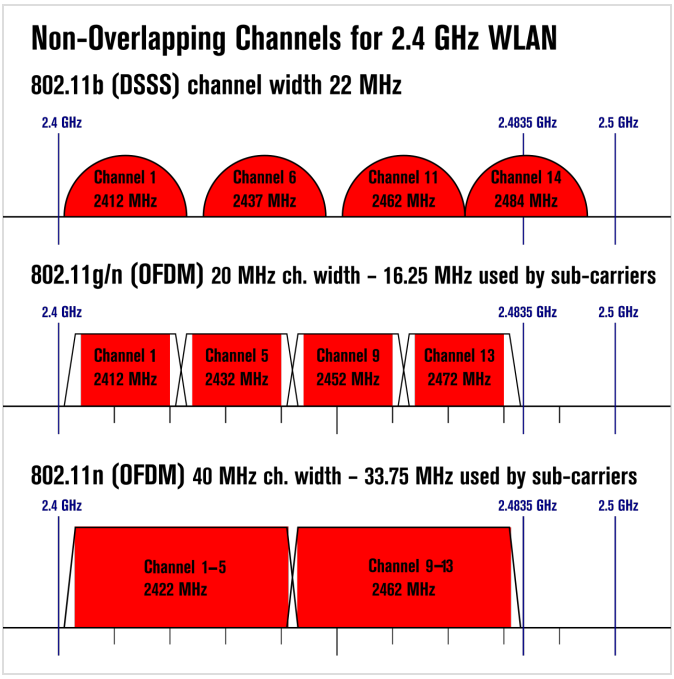
Principe d'une fibre optique.

L'atténuation des supports de transmission

Les technologies sans fils utilisent comme support des ondes électromagnétiques. La transmission des bits s'effectue en utilisant l'onde électromagnétique comme support, via des techniques de modulation, que nous aborderons plus tard. L'émission et la réception de ces ondes se fait par des antennes, intégrées dans toute carte sans fil. Ces techniques sont évidemment moins fiables que le transport par un câble, fût-il optique ou en cuivre. Déjà, les ondes s'atténuent avec la distance qu'elles parcourent : au-delà d'une certaine distance, le signal transmis est trop faible pour être capté. La portée du sans fil est donc limitée, là où les câbles sont capables d'avoir une portée bien plus longue. De plus, les murs et autres objets ont tendance à atténuer les ondes électromagnétiques qui les traversent, réduisant l'amplitude du signal. Là encore, la portée est encore diminuée.

Les longueurs d'onde utilisées sont souvent des ondes radio, des micro-ondes, ou des ondes infrarouges. Les ondes infrarouges sont rarement utilisées dans les réseaux informatiques. Il faut dire que les infrarouges ne traversent pas les murs, sans compter que beaucoup d'objets émettent un rayonnement infrarouge qui peut biaiser le signal. Autant cela ne pose pas de problèmes pour transmettre le signal d'une télécommande, autant cela ne convient pas pour des réseaux sans fils LAN ou WAN. De même, les micro-ondes ne sont pas utilisées dans les réseaux sans fils, pour des raisons différentes. La totalité des réseaux sans fils actuels utilisent des ondes radio, c'est à dire des ondes dont la fréquence est comprise entre 9 kHz et 300 GHz.

Les technologies sans fils peuvent être utilisées aussi bien pour créer des réseaux de type PAN ou LAN que des WAN. On parle alors de Wireless PAN (WPAN), Wireless LAN (WWLAN) et de Wireless WAN (WWAN). Les technologies utilisées ne sont cependant pas les mêmes, les LAN, PAN et WAN ayant des contraintes différentes. Pour les réseaux LAN et PAN, on utilise surtout la technologie WIFI, ainsi que quelques autres. Les technologies sans fils WIFI sont standardisées dans la norme IEEE 802.11. Initialement dans sa version 802.11, cette norme a reçue plusieurs révisions, qui ont donné naissance aux versions 802.11a, 802.11b, 802.11g, 802.11n et 802.11ac. Chaque norme a un débit maximal et une portée différent, les performances s'améliorant à chaque version. Chaque norme définit plusieurs intervalles de fréquences, les canaux, sur lesquels un périphérique peut émettre/recevoir. Le nombre de ces intervalles de fréquences, les canaux, ainsi que leurs limites maximales et minimales sont définie par la norme utilisée.

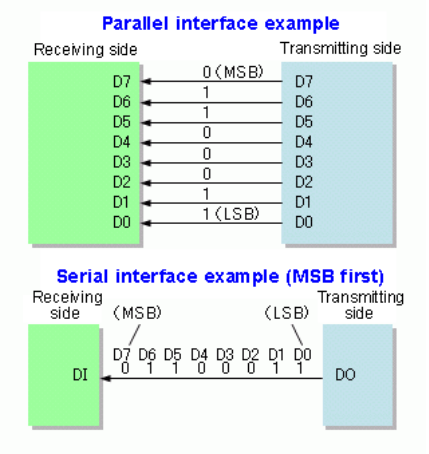


Canaux WIFI 2.4Ghz WLAN.

La couche physique

Dans ce qui va suivre, nous allons parler de la première couche du modèle OSI : la **couche physique**. Pour rappel, cette couche physique s'occupe de la transmission physique des données entre deux équipements réseaux. Elle s'occupe de tout ce qui a trait au bas-niveau, au matériel : la transmission des bits, leur encodage, la synchronisation entre deux cartes réseau, etc. Elle définit les standards des câbles réseaux, des fils de cuivre, du WIFI, de la fibre optique, ou de tout autre support électronique de transmission.

Commençons ce chapitre par préciser une chose importante : les supports de transmission réseau ne transfèrent qu'un bit à la fois. En termes techniques, on dit que ce sont des **liaisons séries**. Il existe des liaisons qui échangent plusieurs bits en même temps, les **liaisons parallèles**, mais ils ne sont pas beaucoup utilisés dans le domaine du réseau. Ils sont plus utilisés à l'intérieur des circuits électroniques, dans les ordinateurs ou pour relier un PC à un périphérique. Pour résumer, les transmissions réseaux sont des flux de bits, transmis un par un.



Les bus et les liaisons point à point

Pour commencer ce chapitre, nous allons aborder la différence entre une liaison point à point et un bus. Notons que cette distinction ne vaut que pour les réseaux qui relient entre eux leurs nœuds avec un ou plusieurs fils électriques. Les réseau sans-fils sont volontairement omis pour le moment.

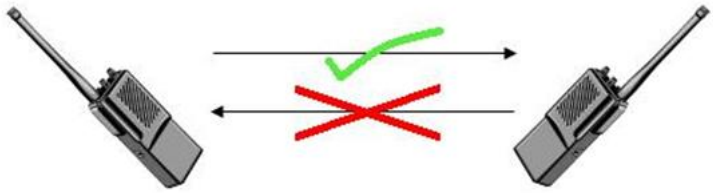
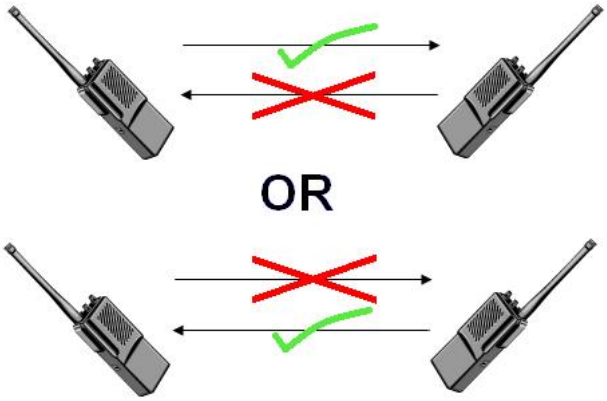
> *Petite précision de vocabulaire pour la suite : Le composant qui envoie une donnée est appelé un émetteur, alors que ceux reçoivent les données sont appelés récepteurs.*

Les liaisons point à point

Les **liaisons point-à-point** se contentent de connecter deux composants entre eux. Par exemple, le câble réseau qui relie votre ordinateur à votre box internet est une liaison point à point. Mais le terme est plus large que cela et regroupe tout ce qui connecte deux équipements informatiques/électroniques entre eux, et qui permet l'échange de données. Par exemple, le câble qui relie votre ordinateur à votre imprimante est lui aussi une liaison point à point, au même titre que les liaisons USB sur votre ordinateur.

Les liaisons point à point sont classés en trois types, suivant les récepteurs et les émetteurs.

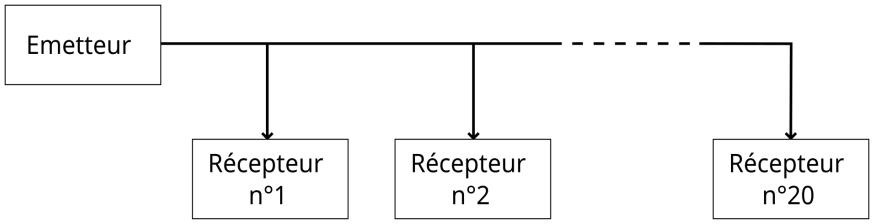
Type de bus	Description
Simplex	Les informations ne vont que dans un sens : un composant est l'émetteur et l'autre reste à tout jamais récepteur.
Half-duplex	Il est possible d'être émetteur ou récepteur, suivant la situation. Par contre, impossible d'être en même temps émetteur et récepteur.
Full-duplex	Il est possible d'être à la fois récepteur et émetteur.

Liaison *simplex*	
Liaison *half-duplex*	
Liaison *full-duplex*	

Les bus *full duplex* sont créés en regroupant deux bus simplex ensemble : un pour l'émission et un pour la réception. Mais certains bus *full-duplex*, assez rares au demeurant, n'utilisent pas cette technique et se contentent d'un seul bus bidirectionnel.

Les bus de communication

Sur un **bus**, on peut connecter un nombre assez important de composants, qui dépasse largement les deux nœuds d'une liaison point à point. Avec un bus, un émetteur va envoyer ses données à tous les autres récepteurs. Sur tous ces récepteurs, il se peut que seul l'un d'entre eux soit le destinataire du paquet : les autres vont alors ignorer le paquet, seul le destinataire traitant le paquet. Cependant, il se peut qu'il y ait plusieurs récepteurs comme destinataires : dans ce cas, les destinataires vont tous recevoir la donnée et la traiter. Ces bus permettent donc de faire des envois de données à plusieurs composants en une seule fois.



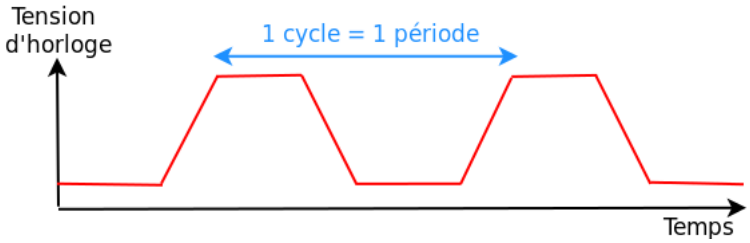
La synchronisation des nœuds

Les différents nœuds d'un réseau local ne vont pas tous à la même vitesse et ils doivent se synchroniser pour échanger des données. Cette synchronisation détermine le temps qui est mis pour transmettre un bit. Rappelons que, quel que soit le support de transmission, le transfert ne peut s'effectuer qu'un bit à la fois. Il existe deux méthodes pour synchroniser la communication, qui permettent de distinguer les liaisons synchrones des liaisons asynchrones.

Les liaisons/bus synchrones

Avec les liaisons synchrones, le support de transmission transmet un bit (ou un paquet de bits) toutes les x millisecondes ou microsecondes. Le temps de transmission d'un bit est appelé la période T. L'inverse de la période $\frac{1}{T}$ est appelé la **fréquence**. Plus celle-ci est élevée, plus le fil peut transmettre de bits en une seconde : la transmission sera donc plus rapide.

La liaison entre deux nœuds transmet, outre les bits de la donnée, une tension de même période/fréquence qui évolue de manière cyclique, appelée **signal d'horloge**. Ainsi, le récepteur sait quand un nouveau bit est transmis : il a juste à regarde le signal d'horloge pour savoir quand démarre un nouvel envoi. Sur la plupart des bus, un nouvel envoi est signalé par le passage de 0 à 1 du signal d'horloge, autrement dit un front montant. Plus rarement, certains bus considèrent au contraire qu'un nouvel envoi est signalé par un front descendant, à savoir le passage de 1 à 0 du signal d'horloge.

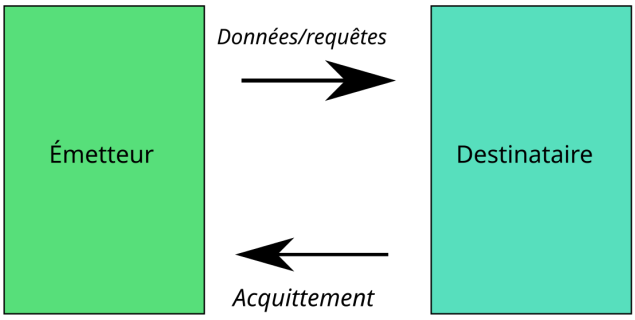


Le signal d'horloge est souvent transmis sur un fil à part, complémentaire du fil sur lequel sont transmises les données.

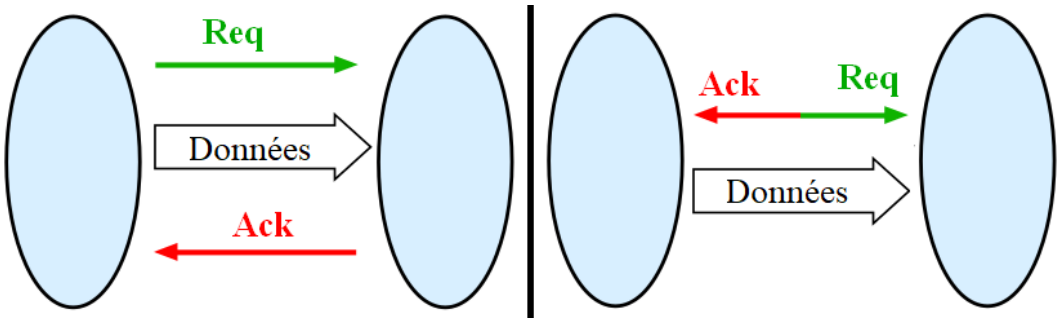
Il va de soit que plus la fréquence est élevée, plus la liaison sera rapide et pourra transmettre de bits par seconde.

Les liaisons/bus asynchrones

Les liaisons asynchrones n'utilisent pas d'horloge pour synchroniser les nœuds du réseau. Pour se synchroniser, le premier nœud indique au second qu'il lui envoie une donnée. Le récepteur réceptionne la donnée et indique qu'il l'a prise en compte. Cette synchronisation se fait grâce à des fils spécialisés du bus/de la liaison, qui transmettent des bits particuliers.



Généralement, cette synchronisation utilise un système d'acquittement, qui utilise deux fils nommés ACK et REQ. Le fil REQ indique au récepteur que l'émetteur lui a envoyé une donnée, tandis que le fil ACK indique que le récepteur a fini son travail et accepte la donnée indiquée par REQ. Lorsqu'un nœud veut envoyer une information à un autre, celui-ci place le fil REQ à 1, afin de dire au récepteur : « attention, j'ai besoin que tu me fasses quelque chose ». Le récepteur va alors réagir et va faire ce qu'on lui a demandé. Une fois qu'il a terminé, il va alors positionner le fil ACK à 1 histoire de dire : j'ai terminé ! Plus rarement, un seul fil est utilisé à la fois pour la requête et l'acquittement : un 1 sur ce fil signifie qu'une requête est en attente (le second composant est occupé), tandis qu'un 0 indique que le second composant est libre. Ce fil est manipulé aussi bien par l'émetteur que par le récepteur. L'émetteur met ce fil à 1 pour envoyer une donnée, le récepteur le remet à 0 une fois qu'il est libre.

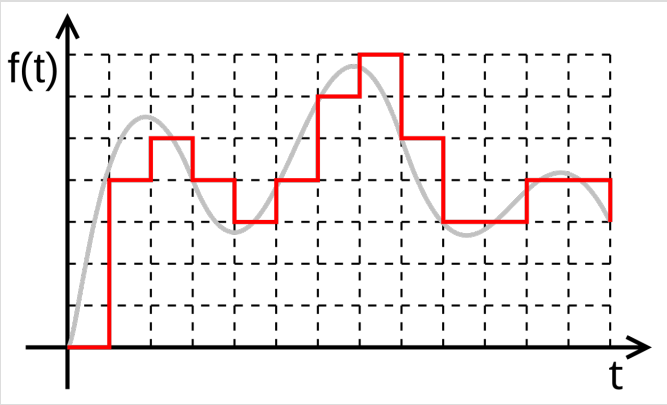


Le codage des bits

Avec les supports actuels, l'information à transmettre va être codée sous la forme d'une tension électrique ou de tout autre mesure électrique. Le transfert de cette information prend donc la forme d'ondes électromagnétiques qui se propagent de l'émetteur vers le récepteur, soit dans l'atmosphère (technologies sans-fils) soit sur un support matériel (câbles réseaux). Cette onde va représenter un flux de bits, la conversion entre flux de bits et onde électromagnétique étant appelé un **codage en ligne**. Nous allons commencer par voir comment un flux de bits peut être représenté/envoyé sur le support de communication avec des tensions/ondes électromagnétiques.

Généralement, un bit est codé avec une tension électrique, plus rarement avec un courant. Les méthodes pour faire la correspondance tension <-> bit envoyé sont assez nombreuses. On peut les classer grosso-modo en deux classes : les codes numériques, aussi appelés codes en ligne, et les codes analogiques. Les codes numériques transmettent un signal discontinu sur le support de transmission. Dit autrement, ils se bornent à représenter un bit par

un certain niveau de tension, comme un +5 Volts ou -5 Volts. Le flux de bits peut donc être transmis tel quel sur le support de transmission. Les codes numériques sont à opposer aux codes analogiques, qui transmettent un signal continu. Généralement, ce signal est une tension sinusoïdale, qui est déformée selon le bit à transmettre.



Comparaison entre signal digital (carré, en rouge) et signal analogique (sinusoïde, en gris).

Le codage en ligne

Il existe des méthodes relativement nombreuses pour coder un bit de données. Toutes codent celui-ci avec une tension, qui peut prendre un état haut (tension forte) ou un état bas (tension faible, le plus souvent proche de 0 volts).

Les codages non-différentiels

Pour commencer, nous allons voir les codages qui permettent de transférer un bit sur un seul fil (nous verrons que d'autres font autrement, mais laissons cela à plus tard). Il en existe de toutes sortes, qui se distinguent par des caractéristiques électriques qui sont à l'avantage ou au désavantage de l'un ou l'autre suivant la situation : meilleur spectre de bande passante, composante continue nulle/non-nulle, etc. Les plus courants sont les suivants :

- Le **codage NRZ-L** utilise l'état haut pour coder un 1 et l'état bas pour le zéro (ou l'inverse).
- Le **codage RZ** est similaire au codage NRZ, si ce n'est que la tension retourne systématiquement à l'état bas après la moitié d'un cycle d'horloge. Celui-ci permet une meilleure synchronisation avec le signal d'horloge, notamment dans les environnements bruités.
- Le codage **NRZ-M** fonctionne différemment : un état haut signifie que le bit envoyé est l'inverse du précédent, tandis que l'état bas indique que le bit envoyé est identique au précédent.
- Le codage **NRZ-S** est identique au codage NRZ-M si ce n'est que l'état haut et bas sont inversés.
- Avec le **codage Manchester**, aussi appelé codage biphasé, un 1 est codé par un front descendant, alors qu'un 0 est codé par un front montant (ou l'inverse, dans certaines variantes).

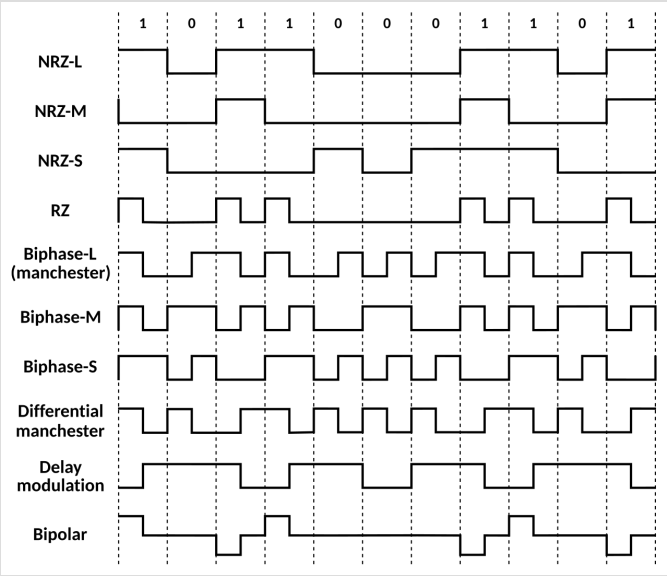
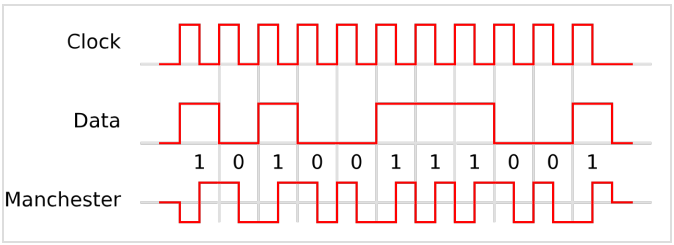


Illustration des différents codes en ligne.

Faisons une petite remarque sur le codage de Manchester : il s'obtient en faisant un OU logique entre l'horloge et le flux de bits à envoyer (codé en NRZ-L). Les bits y sont donc codés par des fronts montants ou descendants, et l'absence de front est censé être une valeur invalide. Si je dis censé, c'est que de telles valeurs invalides peuvent avoir leur utilité, comme nous le verrons dans le chapitre sur la couche liaison. Elles peuvent en effet servir pour coder autre chose que des bits, comme des bits de synchronisation entre émetteur et récepteur, qui ne doivent pas être confondus avec des bits de données. Mais laissons cela à plus tard.



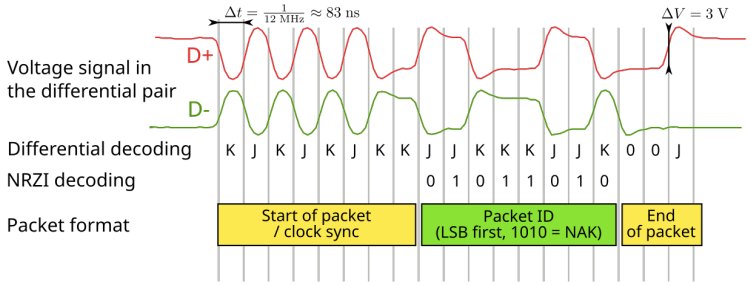
Codage de Manchester.

Les codages différentiels

Pour plus de fiabilité, il est possible d'utiliser deux fils pour envoyer un bit (sur un bus série). Ces deux fils ont un contenu qui est inversé électriquement : le premier utilise une tension positive pour l'état haut et le second une tension négative. Ce faisant, on utilise la différence de tension pour coder le bit, ce qui est plus fiable que d'utiliser des tensions deux fois plus élevées. Ce codage permet une meilleure résistance aux perturbations électromagnétiques, aux parasite et autres formes de bruits qui peuvent modifier les bits transmis. L'intérêt d'un tel montage est que les perturbations électromagnétiques vont modifier la tension dans les deux fils, la variation induite étant identique dans chaque fil. La différence de tension entre les deux fils ne sera donc pas influencée par la perturbation. Un tel codage est appelé un **codage différentiel**.

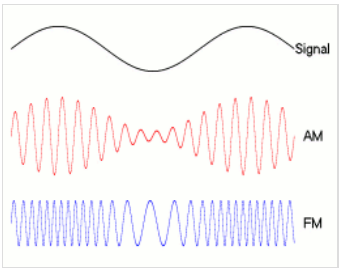
> *Évidemment, chaque codage a son propre version différentielle, à savoir avec deux fils de transmission.*

Ce type de codage est, par exemple, utilisé sur le protocole USB. Sur ce protocole, deux fils sont utilisés pour transmettre un bit, via codage différentiel. Dans chaque fil, le bit est codé par un codage NRZ-L.



Le codage par modulation

Généralement, les signaux analogiques les plus simples sont formés de tensions sinusoïdales ou cosinusoïdales. IL est possible de transmettre un signal numérique à travers un signal analogique comme une sinusoïde. Celle-ci doit simplement subir quelques transformations via des techniques dites de modulation. Cela peut consister à modifier l'amplitude de la sinusoïde selon le bit à transmettre, ou à modifier sa fréquence ou sa phase. La sinusoïde non-déformée est appelée la **porteuse**. Cette porteuse étant un signal cyclique, elle se reproduit à l'identique toutes les x microsecondes, la durée d'un cycle étant appelée la période. Et qui dit période dit aussi fréquence, comme pour les signaux numériques. Il va de soit que plus la fréquence de cette porteuse est importante, plus celle-ci pourra transmettre un grand nombre de bits par secondes. La différence avec les codes numériques est qu'un cycle de porteuse peut parfaitement transmettre plusieurs bits à la fois. D'où des performances parfois plus importantes avec la modulation qu'avec les codes en ligne, dans une certaine mesure.

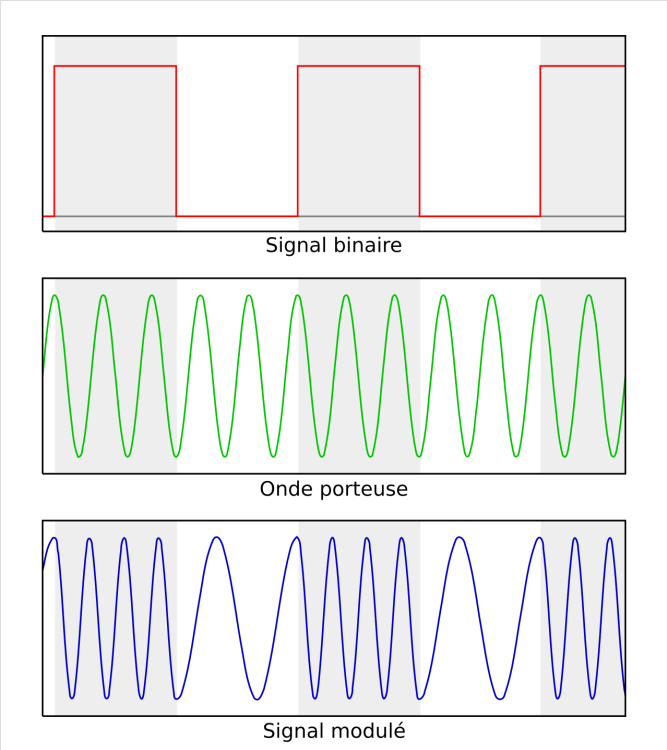


Un signal modulant une porteuse en amplitude ou en fréquence.

Techniques de modulation les plus simples

La méthode de modulation la plus simple est la **modulation d'amplitude**. Elle consiste à modifier l'amplitude de la sinusoïde en fonction du bit à transmettre. L'amplitude est alors maximale pour un 1 et minimale pour un 0. Il est aussi possible de transmettre un paquet de bits en une seule période : l'amplitude dépend alors de la valeur de ce paquet de bits. Par exemple, prenons un paquet de 2 bits. L'amplitude pourra prendre quatre valeurs, chacune correspondant à un paquet bien précis. Elle sera maximale pour le paquet 11, minimale pour le paquet 00 et intermédiaire pour les deux autres. On peut aussi augmenter le nombre de paliers : de 4 paliers pour deux bits, on peut passer à 8 paliers pour 3 bits, 16 pour 4 bits, et ainsi de suite. Mais cela demande des circuits capables de déceler des différences de tension assez fines, vu que la différence de tension entre deux paliers diminue avec le nombre de paliers, à tension maximale fixe.

La **modulation de fréquence** consiste à modifier la fréquence de la porteuse en fonction du bit à transmettre. La porteuse peut donc prendre plusieurs valeurs en fréquences, chaque valeur correspondant à un bit ou paquet de bit précis. Cette fois-ci, on trouve plusieurs paliers de fréquence, au lieu de paliers d'amplitudes. Cette forme de modulation est moins sensible aux interférences électromagnétiques, qui perturbent surtout l'amplitude du signal. Par contre, les circuits de détection de la fréquence, qui extraient le signal numérique transmis, sont plus complexes.

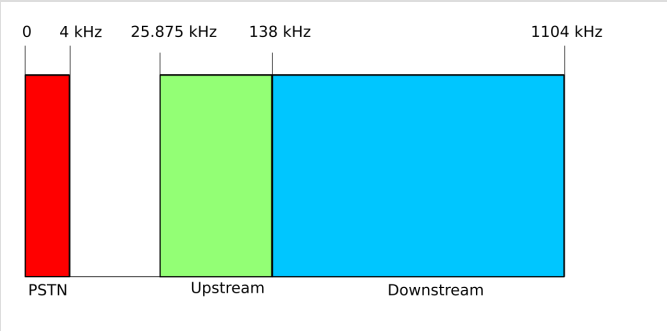


Modulation de fréquence

Utilisation dans les réseaux

Généralement, ces signaux analogiques sont bien adaptés aux supports de transmission sans fils, qui fonctionnement à base d'ondes électromagnétiques. Il est facile de créer des ondes électromagnétiques sinusoïdales, ce qui facilite fortement leur usage. Mais il faut signaler que ces câbles téléphoniques en cuivre peuvent transmettre aussi bien un signal numérique (un flux de bits) qu'un signal analogique (codé avec des signaux sinusoïdaux). Que ce soit pour transmettre le signal téléphonique ou le signal Internet, le câble de cuivre est utilisé pour transmettre des tensions sinusoïdales, dans un intervalle de fréquence fixe. Avant l'invention de l'ADSL, les fréquences utilisées étaient celles qui transmettent le signal téléphonique. On ne pouvait donc pas utiliser le téléphone en même temps qu'Internet. Les fréquences du signal téléphoniques étant assez faibles, la vitesse de transmission ne pouvait pas être très bonne. L'Internet de l'époque était donc limité à 56 Kbits par secondes. Les technologies DSL utilisent des fréquences différentes de celles utilisées pour transmettre le signal téléphonique. Les fréquences utilisées par l'ADSL sont illustrées dans le schéma à droite.

Les technologies (A)DSL utilisent une forme de codage par modulation assez impressionnante techniquement. L'ADSL utilise précisément plusieurs porteuses, dont la fréquence est multiple de 4,3125 kHz. L'ADSL utilise un intervalle de fréquences la bande de fréquences entre 0 Hz et 1,1 MHz, découpé en 255 intervalles plus petits. Chaque intervalle est associé à une porteuse, sauf le tout premier : on a donc 254 porteuses en tout. Certaines porteuses sont réservées à l'upload (transfert client -> internet) et d'autres au download (téléchargement d'internet vers le client). Le nombre de porteuses étant plus grand pour le téléchargement que pour l'upload, on en déduit que la vitesse d'upload est naturellement plus limitée que celle du download. Chaque porteuse peut convoyer environ 2000 bits par secondes. La méthode de modulation varie selon le modem ou la technologie ADSL utilisée, aussi nous ne pouvons pas détailler plus. Parler plus en détail de ces techniques de modulation demanderait cependant de faire un véritable cours de traitement du signal analogique, chose qui pourrait rebuter beaucoup de nos lecteurs. Aussi, nous n'irons pas plus loin dans les explications.



Fréquences utilisées par l'ADSL.

La couche liaison

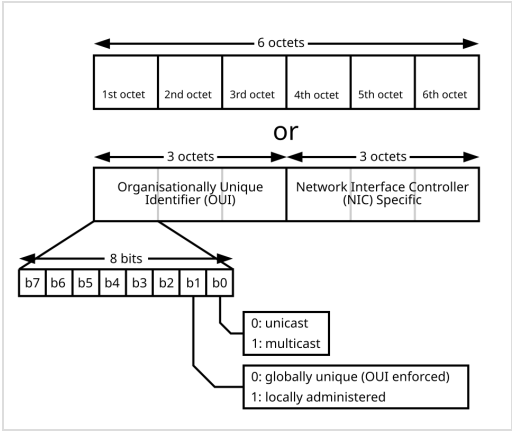
Comme dit dans le chapitre précédent, un réseau local est un réseau contenant peu d'ordinateurs, qui sont reliés entre eux par des câbles réseaux, des routeurs ou des switchs. La gestion des réseaux locaux est le fait de la **couche liaison**, qui prend en charge les fonctionnalités suivantes : la mise en forme des données envoyés sur le réseau, l'identification des ordinateurs et la détection/correction des erreurs de transmission.

L'adressage physique

Pour simplifier, chaque ordinateur d'un réseau local reçoit un numéro qui permet de l'identifier : l'**adresse physique**. Pour être plus précis, l'adresse MAC est attribué aux cartes réseaux, mais aussi aux concentrateurs, aux routeurs, et à d'autres types de matériels. L'utilisation principale est celle des cartes réseaux : chaque carte réseau d'un ordinateur dispose de sa propre adresse physique. Par abus de langage, on dit que c'est l'ordinateur qui a cette adresse MAC, ce qui est rarement problématique. Mais il existe quelques rares cas où un ordinateur a plusieurs cartes réseaux, et donc plusieurs adresses physiques ! D'autres équipements, comme les routeurs ou les commutateurs ont une adresse physique, qui permet de s'y connecter si besoin. Cela peut, par exemple, servir à configurer un routeur/commutateur via un ordinateur connecté sur le réseau, sans avoir à se brancher directement sur le routeur/commutateur. Les administrateurs réseaux utilisent souvent cette possibilité pour configurer les routeurs d'un réseau depuis leur bureau, sans avoir à se déplacer dans plusieurs bâtiments pour chaque configurer chaque routeur à la main.

L'adressage MAC (*Media Access Control*)

De nos jours, les adresses physiques sont définies par le standard MAC (*Media Access Control*). Avec ce standard, chaque adresse physique fait 6 octets (48 bits), et est appelée une **adresse MAC**.



MAC-48 Address

Chaque équipement réseau dispose d'une adresse MAC unique au monde, attribuée par son fabricant. Mais l'utilisateur peut changer l'adresse MAC d'un réseau ou d'un ordinateur. Pour indiquer si l'adresse a été changée par l'utilisateur, l'adresse MAC contient un bit U/L. Il vaut 1 si l'adresse a été changée par l'utilisateur et 0 si l'adresse est l'adresse MAC originelle.

Les réseaux locaux eux-mêmes ont une adresse MAC, ce qui sert pour connecter des réseaux locaux entre eux. Si on envoie un paquet sur l'adresse d'un réseau, celle-ci sera envoyée à tous les ordinateurs du réseau : on parle alors de *broadcast*, ou encore de diffusion. Pour savoir si l'adresse MAC est celle d'un réseau ou d'un équipement réseau, chaque adresse MAC contient un bit I/G, qui vaut 1 si l'adresse est celle d'un réseau et 0 sinon.

Bit I/G	Bit U/L	Reste de l'adresse MAC
0	0	L'adresse MAC n'a pas été modifiée par l'utilisateur. L'adresse MAC se décompose en une adresse de 24 bits qui identifie un ordinateur dans le réseau, et un Organizationally Unique Identifier (OUI) de 24 bits qui identifie le constructeur du matériel.
0	1	L'adresse MAC a été modifiée par l'utilisateur.
1	0	Le reste de l'adresse pointe vers un réseau local.
1	1	Le reste de l'adresse est intégralement remplie avec des bits à 1. Elle est interprétée comme étant une adresse de broadcast pour le réseau local.

L'utilité de l'adressage MAC

L'adressage MAC permet de résoudre un problème se pose sur tous les réseaux (diffusants ou non). Grâce à lui, on peut déterminer le destinataire d'une trame pour qu'elle arrive à destination. Pour cela, chaque trame contient toutes les informations nécessaires pour identifier le destinataire. La trame précise son destinataire dans son en-tête de couche liaison, en donnant son adresse MAC.

Sur les réseaux diffusants, les trames sont envoyées à tous les ordinateurs en même temps, même à ceux qui ne sont pas destinataires de la donnée. Les récepteurs espionnent le support de transmission en permanence pour détecter les trames qui leur sont destinées, mais ils ne doivent pas prendre en compte les trames dont ils ne sont pas destinataires. Pour cela, toute trame indique quelle est son adresse MAC de destination. L'adresse en question est intégrée à la trame et est placée à un endroit précis, le plus souvent à son début. Les ordinateurs ont juste à extraire l'adresse MAC de la trame, et la comparer avec la leur pour savoir s'ils sont destinataires. Pour que la transmission soit la plus rapide, il vaut mieux que les périphériques sachent le plus rapidement si la trame leur est destinée, ce qui demande de mettre l'adresse au début de la trame.

Le codage du début et de la fin de transmission

Sur un réseau local, les données sont échangées sous la forme de paquets de données de taille fixe, qui sont émis par un ordinateur et se propagent dans le réseau local jusqu'à l'ordinateur de destination. Les paquets de données en question s'appellent des **trames** réseau. Le terme trame réfère aux données transmises : toutes les informations nécessaires pour une transmission sont regroupées dans ce qu'on appelle une **trame** qui est envoyée telle quelle.

Le codage des trames indique comment l'émetteur doit envoyer les données sur le support de communication, ce qui permet aux récepteurs de détecter les données transmises et les interpréter. Une trame doit fournir plusieurs informations. Premièrement, le codage des trames doit permettre une synchronisation des équipements réseaux, qui ne fonctionnent pas forcément à la même vitesse. Cela se fait par la transmission d'un signal de synchronisation ou signal d'horloge dans les trames. Enfin, la trame doit contenir toutes les données de la transmission, et de quoi contrôler celle-ci :

indiquer les récepteurs, dire quand commence la transmission et quand elle se termine, etc. Mais surtout, le codage des trames doit indiquer quand commence une transmission et où elle se termine. Le récepteur ne reçoit en effet qu'un flux de bits et doit en extraire les trames. Cette extraction n'est pas simple et l'émetteur doit fatalement ajouter des bits pour coder le début et la fin des trames.

Inactiver la liaison à la fin de l'envoi d'une trame

Une première solution est de laisser la liaison complètement inactive durant un certain temps, entre l'envoi de deux trames. La liaison reste à 0 Volts durant un temps fixe à la fin de l'émission d'une trame. Les composants détectent alors ce temps mort et en déduisent que l'envoi de la trame est terminée. Malheureusement, cette méthode pose quelques problèmes.

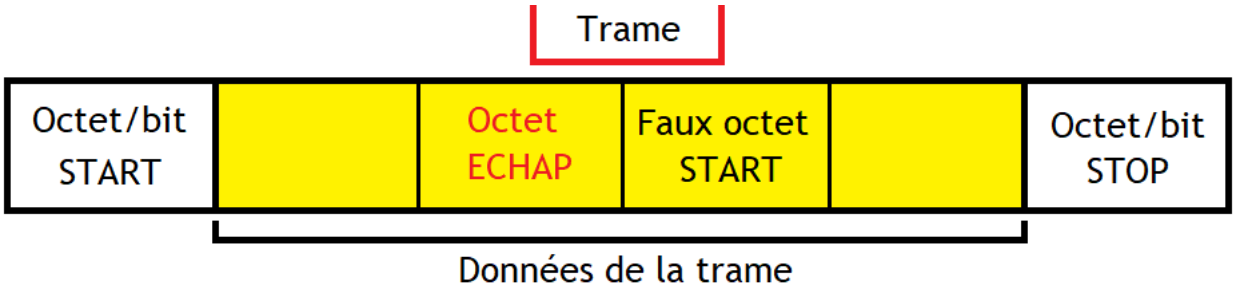
- Premièrement, elle réduit les performances. Une bonne partie du débit binaire de la liaison passe dans les temps morts de fin de trame, lorsque la liaison est inactivée.
- Deuxièmement, certaines trames contiennent de longues suites de 0, qui peuvent être confondues avec une liaison inactive.

Dans ce cas, le protocole de couche liaison peut résoudre le problème en ajoutant des bits à 1, dans les données de la trame, pour couper le flux de 0. Ces bits sont identifiés comme tel par l'émetteur, qui reconnait les séquences de bits problématiques.

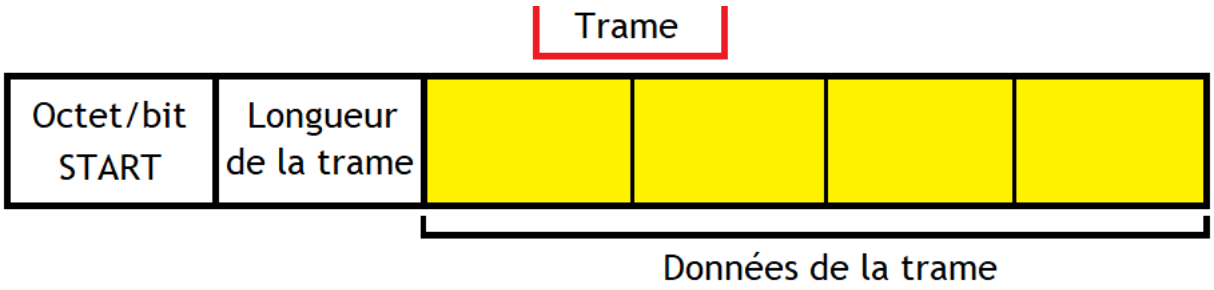
Les octets START et STOP

De nos jours, la quasi-totalité des protocoles utilisent la même technique : ils placent un octet spécial (ou une suite d'octet) au début de la trame, et un autre octet spécial pour la fin de la trame. Ces octets de synchronisation, respectivement nommés START et STOP, sont standardisés par le protocole.

Problème : il se peut qu'un octet de la trame soit identique à un octet START ou STOP. Pour éviter tout problème, ces pseudo-octets START/STOP sont précédés par un octet d'échappement, lui aussi standardisé, qui indique qu'ils ne sont pas à prendre en compte. Les vrais octets START et STOP ne sont pas précédés de cet octet d'échappement et sont pris en compte, là où les pseudo-START/STOP sont ignorés car précédés de l'octet d'échappement. Cette méthode impose au récepteur d'analyser les trames, pour détecter les octets d'échappements et interpréter correctement le flux de bits reçu. Mais cette méthode a l'avantage de gérer des trames de longueur arbitrairement grandes, sans vraiment de limites.



Une autre solution consiste à remplacer l'octet/bit STOP par la longueur de la trame. Immédiatement à la suite de l'octet/bit START, l'émetteur va envoyer la longueur de la trame en octet ou en bits. Cette information permettra au récepteur de savoir quand la trame se termine. Cette technique permet de se passer totalement des octets d'échappement : on sait que les octets START dans une trame sont des données et il n'y a pas d'octet STOP à échapper. Le récepteur a juste à compter les octets qu'il reçoit et 'a pas à détecter d'octets d'échappements. Avec cette approche, la longueur des trames est bornée par le nombre de bits utilisés pour coder la longueur. Dit autrement, elle ne permet pas de trames aussi grandes que possibles.



Dans le cas où les trames ont une taille fixe, à savoir que leur nombre d'octet ne varie pas selon la trame, les deux techniques précédentes sont inutiles. Il suffit d'utiliser un octet/bit de START, les récepteurs ayant juste à compter les octets envoyés à sa suite. Pas besoin de STOP ou de coder la longueur de la trame.

Les bits de START/STOP

Il arrive plus rarement que les octets de START/STOP soient remplacés par des bits spéciaux ou une séquence particulière de fronts montants/descendants.

Une possibilité est d'utiliser les propriétés certains codages, comme le codage de Manchester. Dans celui-ci, un bit valide est représenté par un front montant ou descendant, qui survient au beau milieu d'une période. L'absence de fronts durant une période est censé être une valeur invalide, mais les concepteurs de certains bus ont décidé de l'utiliser comme bit de START ou STOP. Cela donne du sens aux deux possibilités suivantes : la tension reste constante durant une période complète, soit à l'état haut, soit à l'état bas. Cela permet de coder deux valeurs supplémentaires : une où la tension reste à l'état haut, et une autre où la tension reste à l'état bas. La première valeur sert de bit de START, alors que l'autre sert de bit de STOP. Cette méthode est presque identique aux octets de START et de STOP, sauf qu'elle a un énorme avantage en comparaison : elle n'a pas besoin d'octet d'échappement dans la trame, pas plus que d'indiquer la longueur de la trame.

Un autre exemple est celui des bus RS-232, RS-485 et I²C, où les bits de START et STOP sont codés par des fronts sur les bus de données et de commande.

La fiabilité des transmissions

Lorsqu'une trame est envoyée, il se peut qu'elle n'arrive pas à destination correctement. Des parasites peuvent déformer la trame et/ou en modifier des bits au point de la rendre inexploitable. Dans ces conditions, il faut systématiquement que l'émetteur et le récepteur détectent l'erreur : ils doivent savoir que la trame n'a pas été transmise ou qu'elle est erronée. Pour cela, il existe diverses méthodes de détection et de correction d'erreur. On en distingue deux classes : celles qui ne font que détecter l'erreur, et celles qui permettent de la corriger.

Détection et correction d'erreur

Tous les codes correcteurs et détecteurs d'erreur ajoutent tous des bits aux données de base, ces bits étant appelés des bits de correction/détection d'erreur. Ces bits servent à détecter et éventuellement corriger toute erreur de transmission/stockage. Plus le nombre de bits ajoutés est important, plus la fiabilité des données sera importante. Dans le cas le plus simple, on se contente d'un simple bit de parité. Dans d'autres cas, on peut ajouter une somme de contrôle ou un code de Hamming à la trame. Cela permet de détecter les erreurs de transmission, et de la corriger dans certains cas.

Bit de parité

Nous allons commercer par aborder le **bit de parité/imparité**, une technique utilisée dans un grand nombre de circuits électroniques, comme certaines mémoires RAM, ou certains bus (RS232, notamment). Il permet de détecter des corruptions qui touchent un nombre impair de bits. Si un nombre pair de bit est modifié, il sera impossible de détecter l'erreur avec un bit de parité. Il faut noter que ce bit de parité, utilisé seul, n permet pas de localiser le bit corrompu. Ce bit de parité est ajouté aux bits à stocker. Avec ce bit de parité, le nombre stocké (bit de parité inclus) contient toujours un nombre pair de bits à 1. Ainsi, le bit de parité vaut 0 si le nombre contient déjà un nombre pair de 1, et 1 si le nombre de 1 est impair. Si un bit s'inverse, quelle qu'en soit la raison, la parité du nombre total de 1 est modifié : ce nombre deviendra impair si un bit est modifié. Et ce qui est valable avec un bit l'est aussi pour 3, 5, 7, et pour tout nombre impair de bits modifiés. Mais tout change si un nombre pair de bit est modifié : la parité ne changera pas. Ainsi, on peut vérifier si un bit (ou un nombre impair) a été modifié : il suffit de vérifier si le nombre de 1 est impair.

Le **bit d'imparité** est similaire au bit de parité, si ce n'est que le nombre total de bits doit être impair, et non pair comme avec un bit de parité. Sa valeur est l'inverse du bit de parité du nombre : quand le premier vaut 1, le second vaut 0, et réciproquement. Mais celui-ci n'est pas meilleur que le bit de parité : on retrouve l'impossibilité de détecter une erreur qui corrompt un nombre pair de bits.

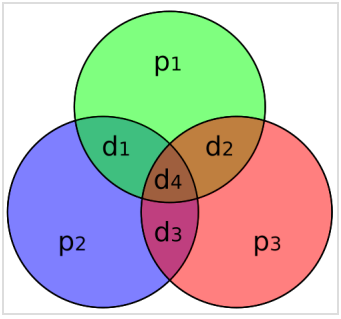
Pour obtenir plus d'informations sur le calcul logiciel de ce bit, je vous suggère de lire le chapitre de mon cours sur les opérations bits à bits : Bit de parité.

Code de Hamming

Le **code de Hamming** se base sur l'usage de plusieurs bits de parité pour un seul nombre. Chaque bit de parité n'est cependant pas calculé en prenant en compte la totalité des bits du nombre : seul un sous-ensemble de ces bits est utilisé pour calculer chaque bit de parité. Chaque bit de parité a son propre sous-ensemble, tous étant différents, mais pouvant avoir des bits en commun. Le but étant que deux sous-ensembles partagent un bit : si ce bit est modifié, cela modifiera les deux bits de parité associés. Et la modification de ce bit est la seule possibilité pour que ces deux bits soient modifiés en même temps : si ces deux bits de parité sont modifiés en même temps, on sait que le bit partagé a été modifié. Pour résumer, un code de Hamming utilise plusieurs bits de parité, calculés chacun à partir de bits différents, souvent partagés entre bits de parité.

Le code de Hamming le plus connu est certainement le **code 7-4-3**, un code de Hamming parmi les plus simples à comprendre. Celui-ci prend des données sur 4 bits, et leur ajoute 3 bits de parité, ce qui fait en tout 7 bits : c'est de là que vient le nom de 7-4-3 du code. Chaque bit de parité se calcule à partir de 3 bits du nombre. Pour poursuivre, nous allons noter les bits de parité p1, p2 et p3, tandis que les bits de données seront notés d1, d2, d3 et d4. Voici à partir de quels bits de données sont calculés chaque bit de parité :

Bits de parité incorrects	Bit modifié
Les trois bits de parité : p1, p2 et p3	Bit d4
p1 et p2	d1
p2 et p3	d3
p1 et p3	d2



Hamming(7,4)

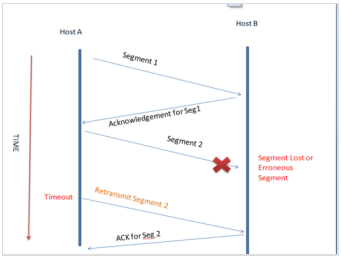
Il faut préciser que toute modification d'un bit de donnée entraine la modification de plusieurs bits de parité. Si un seul bit de parité est incorrect, il est possible que ce bit de parité a été corrompu et que les données sont correctes. Ou alors, il se peut que deux bits de données ont été modifiés, sans qu'on sache lesquels.

Sommes de contrôle

Les sommes de contrôle sont des techniques de correction d'erreur, où les bits de correction d'erreur sont ajoutés à la suite des données. Les bits de correction d'erreur, ajoutés à la fin du nombre à coder, sont appelés la **somme de contrôle**. Techniquement, les techniques précédentes font partie des sommes de contrôle au sens large. Mais il existe un sens plus restreint pour le terme de somme de contrôle. Ce terme est souvent utilisé pour des techniques telle l'addition modulaire, le CRC, et quelques autres. Toutes ont en commun de traiter les données à coder comme un gros nombre entier, sur lequel on effectue des opérations arithmétiques pour calculer les bits de correction d'erreur. La seule différence est que l'arithmétique utilisée est quelque peu différente de l'arithmétique binaire usuelle. Dans les calculs de CRC, on utilise une arithmétique où les retenues ne sont pas propagées. Le calcul des additions et soustractions est alors extrêmement simple : elles se résument à de vulgaires XOR.

Une méthode très utilisée dans le cadre du réseau consiste à prendre les données à envoyer, à les diviser par un nombre entier arbitraire, et à utiliser le reste de la division euclidienne comme somme de contrôle. Cette méthode, qui n'a pas de nom, est similaire à celle utilisée dans les **Codes de Redondance Cyclique**. Effectuer une division dans l'arithmétique utilisée est alors très simple : il suffit d'effectuer la division comme en décimal, en remplaçant les soustractions par des XOR. C'est ainsi que sont calculés les CRC : on divise les données par un diviseur standardisé pour chaque CRC, dans l'arithmétique mentionnée précédemment, et on utilise le reste de la division comme somme de contrôle. L'avantage de ces CRC est qu'ils sont faciles à calculer en matériel. Le remplacement des soustractions entières par des XOR facilite fortement les calculs et leur implémentation. Les circuits de calcul de CRC sont ainsi très simples à concevoir : ce sont souvent de simples registres à décalage à rétroaction linéaire améliorés.

L'Automatic repeat request



ACK TCP

Si l'erreur peut être corrigée par le récepteur, tout va bien. Mais il arrive souvent que ce ne soit pas le cas : l'émetteur doit alors être prévenu et agir en conséquence. Pour cela, le récepteur peut envoyer une trame à l'émetteur qui signifie : la trame précédente envoyée est invalide. Cette trame est appelée un **accusé de non-réception**. La trame fautive est alors renvoyée au récepteur, en espérant que ce nouvel essai soit le bon. Mais cette méthode ne fonctionne pas si la trame est tellement endommagée que le récepteur ne la détecte pas. Pour éviter ce problème, on utilise une autre solution, beaucoup plus utilisée dans le domaine du réseau. Celle-ci utilise des **accusés de réception**, à savoir l'inverse des accusés de non-réception. Ces accusés de réception sont envoyés à l'émetteur pour signifier que la trame est valide et a bien été reçue. Nous les noterons ACK dans ce qui suivra.

Après avoir envoyé une trame, l'émetteur va attendra un certain temps que l'ACK correspondant lui soit envoyé. Si l'émetteur ne reçoit pas d'ACK pour la trame envoyée, il considère que celle-ci n'a pas été reçue correctement et la renvoie. Pour résumer, on peut corriger et détecter les erreurs avec une technique qui mélange ACK et durée d'attente : après l'envoi d'une trame, on attend durant un temps nommé *time-out* que l'ACK arrive, et on renvoie la trame au bout de ce temps si non-réception. Cette technique porte un nom : on parle d'***Automatic repeat request***.

Le protocole *Stop-and-Wait*

Dans le cas le plus simple, les trames sont envoyées unes par unes au rythme d'une trame après chaque ACK. En clair, l'émetteur attend d'avoir reçu l'ACK de la trame précédente avant d'en envoyer une nouvelle. Parmi les méthodes de ce genre, la plus connue est le **protocole *Stop-and-Wait***.

Cette méthode a cependant un problème pour une raison simple : les trames mettent du temps avant d'atteindre le récepteur, de même que les ACK mettent du temps à faire le chemin inverse. Une autre conséquence des temps de transmission est que l'ACK peut arriver après que le time-out (temps d'attente avant retransmission de la trame) soit écoulé. La trame est alors renvoyée une seconde fois avant que son ACK arrive. Le récepteur va alors croire que ce second envoi est en fait l'envoi d'une nouvelle trame !

Pour éviter cela, la trame contient un bit qui est inversé à chaque nouvelle trame. Si ce bit est le même dans deux trames consécutives, c'est que l'émetteur l'a renvoyée car l'ACK était en retard. Mais les temps de transmission ont un autre défaut avec cette technique : durant le temps d'aller-retour, l'émetteur ne peut pas envoyer de nouvelle trame et doit juste attendre. Le support de transmission n'est donc pas utilisé de manière optimale et de la bande passante est gâchée lors de ces temps d'attente.

Les protocoles à fenêtre glissante

Les deux problèmes précédents peuvent être résolus en utilisant ce qu'on appelle une **fenêtre glissante**. Avec cette méthode, les trames sont envoyées les unes après les autres, sans attendre la réception des ACKs. Chaque trame est numérotée de manière à ce que l'émetteur et le récepteur puisse l'identifier. Lorsque le récepteur envoie les ACK, il précise le numéro de la trame dont il accuse la réception. Ce faisant, l'émetteur sait quelles sont les trames qui ont été reçues et celles à renvoyer (modulo les time-out de chaque trame).

On peut remarquer qu'avec cette méthode, les trames sont parfois reçues dans le désordre, alors qu'elles ont été envoyées dans l'ordre. Ce mécanisme permet donc de conserver l'ordre des données envoyées, tout en garantissant le fait que les données sont effectivement transmises sans problèmes. Avec cette méthode, l'émetteur va accumuler les trames à envoyer/déjà envoyées dans une mémoire. L'émetteur devra gérer deux choses : où se situe la première trame pour laquelle il n'a pas d'ACK, et la dernière trame envoyée. La raison est simple : la prochaine trame à envoyer est l'une de ces deux trames. Tout dépend si la première trame pour laquelle il n'a pas d'ACK est validée ou non. Si son ACK n'est pas envoyé, elle doit être renvoyée, ce qui demande de savoir quelle est cette trame. Si elle est validée, l'émetteur pourra envoyer une nouvelle trame, ce qui demande de savoir quelle est la dernière trame envoyée (mais pas encore confirmée). Le récepteur doit juste mémoriser quelle est la dernière trame qu'il a reçue. Lui aussi va devoir accumuler les trames reçues dans une mémoire, pour les remettre dans l'ordre.

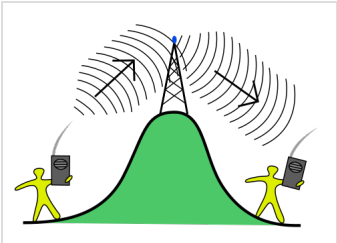
Le matériel des couches 1 et 2 (physique et liaison)

Dans ce chapitre, nous allons parler plus en détail du fonctionnement du matériel réseau. Plus précisément, nous allons parler des concentrateurs, des commutateurs, des cartes réseau et des répéteurs. Ces équipements ont la particularité d'être liés de près ou de loin avec les chapitres précédents. Ce sont des matériels qui prennent en charge tout ce qui a trait aux couches physique et liaison du modèle OSI, mais qui n'ont rien à voir avec les couches plus hautes. En clair, ils ne s'occupent pas de routage, privilège dédié aux routeurs.

Le répéteur

Quand un signal électrique est transmis sur un support de communication, il a tendance à s'atténuer rapidement avec la distance. Cela vaut aussi bien pour les transmissions sans-fils, que pour les signaux guidés par un câble en cuivre ou une fibre optique. L'atténuation est surtout un problème sur les connexions sans-fils, dont la portée ne dépasse pas quelques mètres pour les technologies domestiques. Mais elle pose problème pour les câbles réseaux si la distance parcourue devient assez grande. Par exemple, les grands câbles téléphoniques qui parcourent les villes et les campagnes subissent une atténuation non-négligeable.

Pour éviter les problèmes liés à l'atténuation des signaux, les ingénieurs ont inventé le **répéteur**. Il s'agit ni plus ni moins que d'un matériel qui régénère le signal perçu en entrée. Il reçoit sur son entrée le signal transmis, et produit en sortie le même signal amplifié, similaire au signal non-atténué. En plus de recopier le signal transmis, l'atténuation en moins, il peut aussi connecter deux câbles qui ont des caractéristiques électriques différentes. En somme, il peut servir d'interface entre deux supports de transmission différents, de convertisseur. Précisons cependant que tous les répéteurs ne sont pas dans ce cas : la plupart ne peut pas faire cette conversion et se limite à un rôle d'amplificateur.



Exemple d'un répéteur pour les technologies sans-fils.

Les répéteurs sont surtout utilisés pour les câbles réseaux, les supports de transmission guidés, mais pas pour les technologies sans-fils. Il faut dire que les technologies sans-fils sont quasi-exclusivement utilisées pour les réseaux domestiques, éventuellement les réseaux LAN. Leur faible portée ne les rend pas assez intéressants pour les réseaux étendus, et l'existence de répéteurs pour technologies sans-fils n'y change pas grand chose. La seule exception est celle des télécommunications radio, la télévision et pour les téléphones portables. Outre les antennes d'émission, le pays est maillé par un système de répéteurs radio et téléphoniques qui amplifient les signaux transmis sur les diverses fréquences radio ou télé.

Le concentrateur

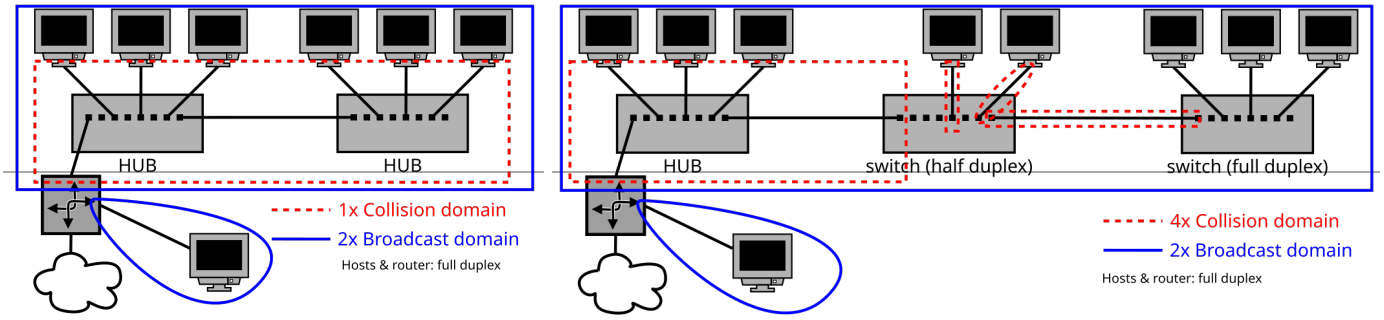
Le **concentrateur** est une version multi-port du répéteur : quand il reçoit un flux de bits sur un port, il recopie celui-ci sur tous les autres ports. Ce faisant, chaque donnée envoyée par un ordinateur est redistribuée à tous les autres. Quand il est placé au centre d'un réseau en étoile, il permet de simuler un réseau en bus.

Pour mieux comprendre ce que fait un concentrateur, il est intéressant de parler du domaine de collision. Le **domaine de collision** comprend l'ensemble des ordinateurs connectés à un même bus logique et/ou physique. Deux ordinateurs dans un même domaine de collision ne peuvent pas envoyer des données sur le bus en même temps : s'ils le font, cela entraîne une collision. Tous les ordinateurs/équipements connectés à un même concentrateurs font partie du même domaine de collision. Si on connecte des concentrateurs entre eux, tous les ordinateurs reliés aux différents concentrateurs (eux-même connectés) forment le domaine de collisions.

Petite précision : il ne faut pas confondre le domaine de collision avec le domaine de diffusion vu il y a quelques chapitres. Pour rappel, le domaine de diffusion est l'ensemble d'ordinateurs qui peut être atteint par une transmission de type *broadcast* (envoyée à tous les ordinateurs d'un réseau). La différence entre les deux est subtile, surtout que les deux sont confondus dans les réseaux en bus. Mais il y a une différence entre domaine de diffusion et de collision dans les réseaux qui utilisent des commutateurs et/ou des routeurs. Par exemple, si on relie des ordinateurs avec un commutateur, ceux-ci ne font pas partie du même domaine de collision, mais partagent le même domaine de diffusion (le commutateur peut parfaitement envoyer une trame à tous les ordinateurs d'un réseau). Avec un commutateur, il y a un domaine de collision par port, contre un domaine de collision unique pour le concentrateur.

On peut résumer cela avec les deux affirmations suivantes :

- Les transmissions en *broadcast* traversent à la fois les concentrateurs et les commutateurs.
- Les collisions traversent les concentrateurs, mais ne passent pas à travers les commutateurs.



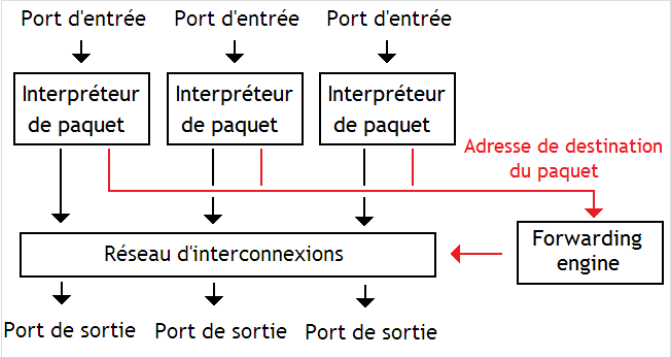
Les commutateurs

Pour rappel, le commutateur est un équipement de couche liaison, qui est utilisé dans les réseaux locaux en étoile, au même titre que les concentrateurs. La différence avec le concentrateur est qu'il redirige les trames reçues vers l'ordinateur de destination uniquement, il ne diffuse pas la trame à tous les ordinateurs du réseau local comme le ferait un concentrateur. Lorsqu'il reçoit une trame, il la renvoie sur le port qui est associée à l'ordinateur de destination.

L'intérieur d'un commutateur

Pour faire ce travail, le commutateur dispose de plusieurs sous-circuits :

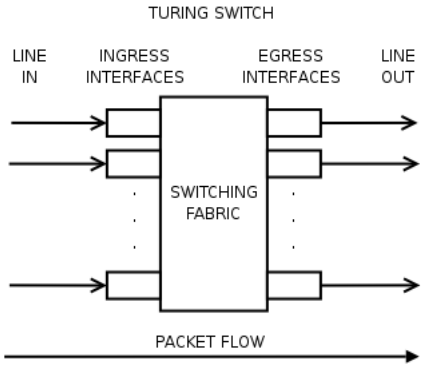
- Un circuit extrait l'adresse MAC de destination des trames reçues (l'analyseur de trames).
- Un circuit qui décide, en fonction de l'adresse MAC de destination, sur quel port l'envoyer (le *forwarding engine*).
- Un circuit d'interconnexion, équivalent à un ensemble de liaisons point-à-point qui relie chaque port à tous les autres.



Micro-architecture d'un switch

Pour faire le lien entre adresse MAC de destination et le port qui correspond, le commutateur a juste besoin de maintenir une table de correspondance entre adresse MAC et numéro de port, appelée la **table CAM**.

Une fois que le port de destination est connu, le commutateur en déduit quels ports connecter entre eux. Pour cela, il a juste à configurer un circuit d'interconnexion, qui relie les ports entre eux et qui est équivalent à un ensemble de connexions point à point configurables. Ce circuit d'interconnexion porte un nom : c'est la ***switch fabric***. Il suffit de lui envoyer le numéro du port d'entrée et du port de sortie et le circuit d'interconnexion connecte ces deux ports (les autres ports restent déconnectés, en principe et sauf optimisation).



Pour ceux qui veulent en savoir plus, je conseille la lecture de mon cours sur le fonctionnement d'un ordinateur, et plus précisément du chapitre sur le matériel réseau.

La détection de la topologie par le commutateur

Un commutateur doit découvrir par lui-même les adresses MAC des composants qu'on branche sur ses ports : il ne peut pas les connaitre à l'avance. Pour cela, le commutateur utilise plusieurs méthodes assez simples. Premièrement, si un ordinateur lui envoie une trame sur un port, il met à jour la table CAM avec l'adresse de l'émetteur de la trame : cela fait un port de connu. Même chose avec les accusés de réception des trames, qui contient l'adresse MAC du destinataire : cela fait une autre adresse de connue. Une fois que tous les ordinateurs ont envoyé quelque chose sur le réseau, il connait tous les ports. Si aucun port n'est associé à une adresse de destination, le commutateur envoie le paquet à tous les ports, à tous les ordinateurs du réseau. Le destinataire répondra alors avec un accusé de réception, qui permettra de déterminer son adresse MAC. Ces trois méthodes permettent de remplir progressivement la table CAM. Au tout début, celle-ci est vide. Le commutateur recevra alors des trames qu'il ne saura pas envoyer à destination et devra les envoyer à tous les ordinateurs du réseau. Progressivement, les accusés de réception permettront de remplir la table CAM, de même que les trames envoyées.

Il faut noter que le contenu de la table CAM a une durée de péremption, appelée le **Time To Live**, ou TTL, qui vaut entre 0 et 255 secondes[1].. Cela permet de mettre à jour un réseau local sans avoir à redémarrer le commutateur. On peut changer le composant branché sur un commutateur, celui-ci ne restera pas bloqué sur l'ancien composant et finira par repérer le nouveau au bout d'un certain temps.

Les modes de transfert d'une trame

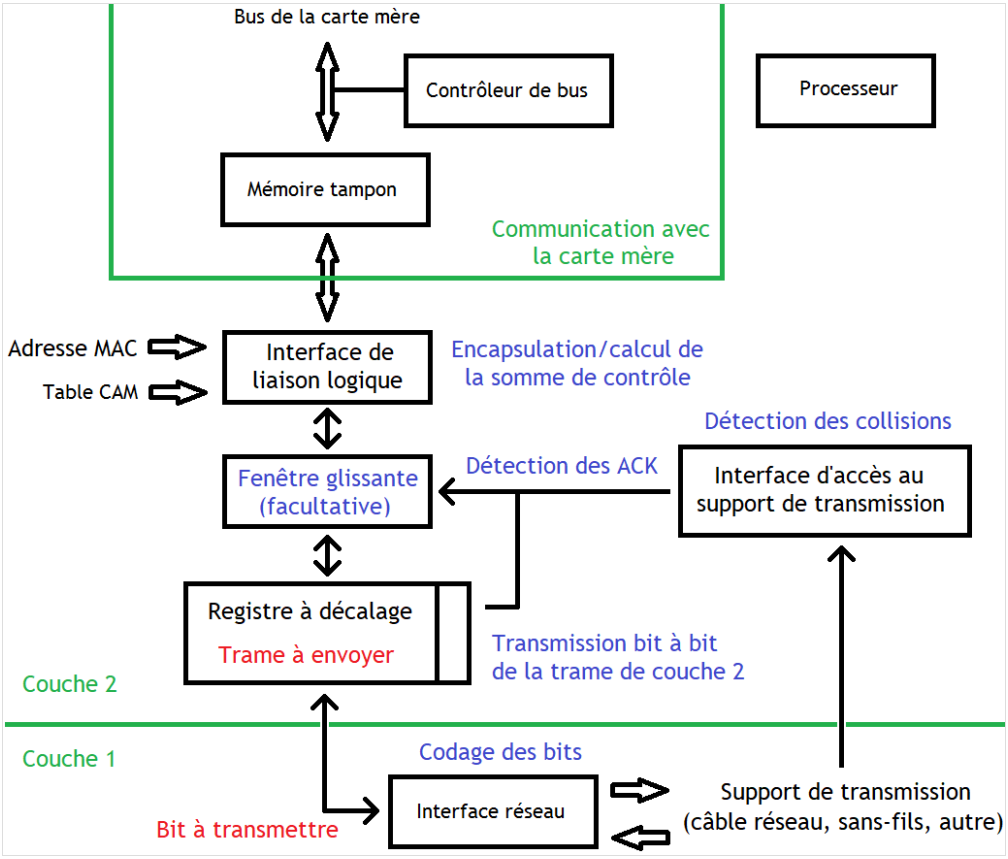
Un commutateur peut transmettre les trames de plusieurs manières différentes, les deux principales étant appelées le ***Store-and-Forward*** et le ***Cut-through***. Le premier mode (*Store-and-Forward*) correspond au cas où le commutateur met en mémoire la trame reçue, l'analyse et la renvoie sur le port adéquat. Ce faisant, le commutateur attend d'avoir reçu l'intégralité de la trame avant de la renvoyer vers sa destination. À l'opposé, le mode *Cut-through*

retransmet la trame dès que possible, sans attendre qu'elle soit reçue dans son entièreté. Le commutateur attend juste de recevoir le début de l'en-tête, juste de quoi recevoir l'adresse MAC de destination, ce qui suffit pour déterminer le port de destination.

En somme, les deux modes ne sont pas vraiment différents, mais ils ont chacun des avantages et désavantages par rapport à l'autre. Dans le mode *Store-and-Forward*, la mise en attente de la trame ajoute un petit temps de latence, ce qui ralentit la communication. Par contre, le commutateur a accès à l'intégralité de la trame, octets de détection/correction d'erreur compris. Il peut alors vérifier l'intégrité de la trame et envoyer un message d'erreur à l'émetteur. De la même manière, des trames incomplètes, liées à des collisions, peuvent être relayés. Dans le mode *Cut-through*, c'est l'inverse. Le temps de latence avant retransmission est minimal, mais le commutateur retransmet des trames incorrectes (fragments de collisions ou trames avec des erreurs).

La carte réseau

Sur un ordinateur tous les traitements des couches liaison et physique sont pris en charge par la **carte réseau**. Celle-ci est le composant qui permet à un ordinateur de communiquer sur un réseau (local ou internet). D'ordinaire, elle permet d'envoyer ou de recevoir des informations sur un câble réseau ou une connexion WIFI. Elle communique avec le reste de l'ordinateur via le bus de la carte mère. Les données échangées sont mémorisées temporairement dans une mémoire tampon. Celle-ci permet de mettre en attente les données à envoyer tant que le réseau n'est pas disponible, ou d'accumuler les données reçues en attendant de les recevoir complètement. Ces données sont ensuite gérées par un circuit qui s'occupe de gérer l'encapsulation (ajout/retrait des adresses MAC, calcul de la somme de contrôle). La gestion de la fenêtre glissante, si elle existe, est prise en charge par un circuit spécialisé juste après. La carte réseau contient ensuite un circuit qui transforme les données à transmettre en ondes WIFI ou en signaux électriques (pour les câbles réseau). Dans tous les cas, les transferts d'informations se font en série (le câble est l'équivalent d'un bus série). L'interface de transfert contient donc deux registres à décalage : un pour faire la conversion parallèle -> série, et un autre pour la conversion série -> parallèle.



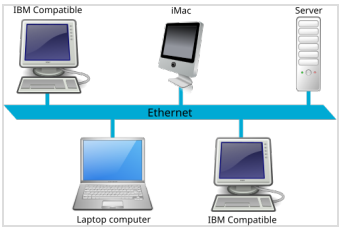
Architecture matérielle d'une carte réseau.

Les protocoles de couche 1 et 2 (physique et liaison)

Dans ce chapitre, nous allons détailler les protocoles des couches liaison et physique qui sont les plus couramment utilisés. Nous allons notamment parler du célébrissime protocole Ethernet.

Un exemple de protocole : Ethernet

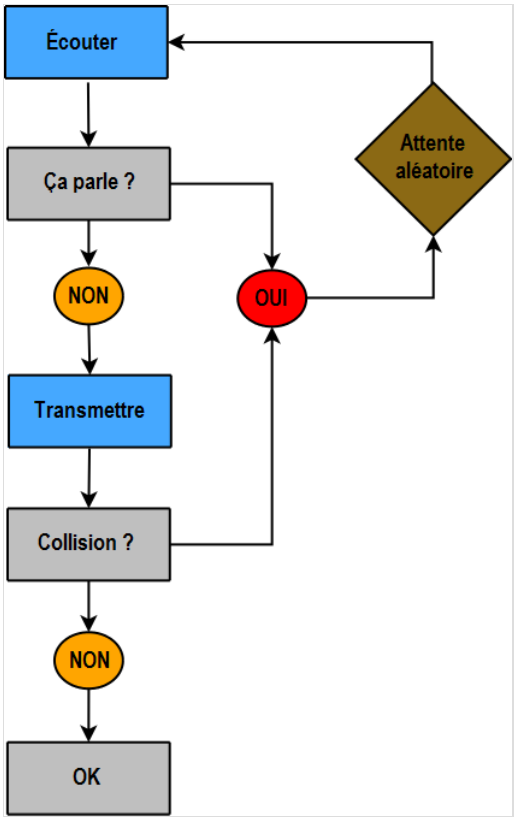
Le standard courant de transmission des données sur un réseau local est l'**Ethernet**. Sa première version date des années 1970 (1973-1976), son inventeur étant la fameuse entreprise Xerox. Elle standardisait la communication sur un bus, aujourd'hui émulé par des concentrateurs (hubs). C'est en 1982 que la seconde version d'Ethernet vit le jour. La troisième version, standardisée par l'IEEE, porte le nom de protocole 802.3. Ce protocole est depuis devenu le protocole le plus populaire pour les réseaux locaux. Techniquement, Ethernet est à la fois un protocole de couche 1 et de couche 2. Sa spécification standardise l'encodage des bits sur un câble Ethernet, par exemple. Cependant, nous allons nous concentrer sur ses fonctionnalités de couche 2 dans ce qui va suivre.



Ethernet LAN

L'arbitrage du bus : le CSMA-CD

Comme tous les autres bus, le bus Ethernet doit contenir des mécanismes d'arbitrage pour résoudre la situation. Le protocole Ethernet utilise une méthode d'arbitrage spéciale : le **CSMA-CD**. De base, les machines attendent que le bus soit libre (personne n'émet) pour émettre une trame. Mais il arrive que deux machines voient que le bus est libre simultanément et démarrent une transmission chacune de leur côté. Il est ainsi parfaitement possible que deux machines émettent sur le bus en même temps et une collision a lieu.



CMSA-CD

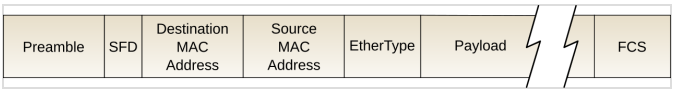
Avec ce protocole, la détection de collision est relativement simple. Quand une machine envoie un 1 sur le bus, elle s'attend à ce que le bus contienne une tension positive, correspondant à un 1. Même chose pour un 0, qui doit donner une tension nulle. Si deux machines émettent sur le bus, les 1 l'emportent sur les 0 : si une machine émet un 1 et une autre un 0, on observera un 1 sur le bus. Pour détecter une collision, chaque machine compare ce qu'elle envoie sur le bus et ce qu'il y a sur le bus. Si elle observe un 1 sur le bus alors qu'elle a envoyé un 0, une collision a eu lieu. Quand une collision a lieu, a machine qui a détecté la collision stoppe sa transmission et envoie une trame spéciale sur le bus, qui indique l'occurrence d'une collision. Lorsqu'une collision a lieu, ou que le bus n'est pas libre, la machine va attendre son tour. Chaque machine attend durant un temps aléatoire, histoire de limiter l'occurrence des collisions.

Les trames Ethernet

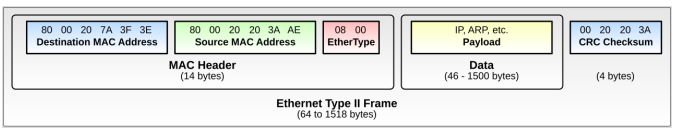
Toute trame Ethernet contient diverses informations, dont :

- l'adresse MAC du destinataire : sans cela, on ne sait pas à qui la donnée est destinée ;
- l'adresse de l'émetteur, information utilisable par le destinataire pour savoir à qui envoyer une éventuelle réponse ;
- la donnée envoyée : un simple bloc de données de taille fixe, le plus souvent ;
- éventuellement des octets de synchronisation ou de contrôle d'erreur.

Dans la première version d'Ethernet, la trame indique elle-même sa longueur en nombre d'octets. Celle-ci est comprise entre 46 et 1500 octets. Si le nombre total d'octets est inférieur à 42, l'équipement réseau ajoute des octets inutiles dans la trame, histoire d'avoir au moins 46 octets. Dans les versions suivantes d'Ethernet, la longueur est remplacée par un numéro qui indique quels sont les protocoles utilisés pour la transmission des données sur le réseau : l'Ethertype. Pour garantir la compatibilité, il a été convenu que les trames Ethernet version 2 ont un champ dont la valeur est supérieure à la longueur de la trame, à savoir 1500.



Trame Ethernet.



Trame Ethernet, version 2.

L'adressage IP

On a vu que les machines d'un réseau local ont une adresse MAC, pour que l'on sache qui est qui sur le réseau. Le même problème se pose sur internet : comment identifier un PC bien précis sur un réseau global ? Par exemple, si vous voulez accéder à un site web sur un serveur, comment votre ordinateur fait-il pour dire : je veux communiquer avec ce serveur bien précis, et pas un autre ? Pour cela, chaque ordinateur possède un numéro qui permet de l'identifier sur le net, chaque numéro étant appelé une adresse logique. De nos jours, les adresses IP sont standardisées par le **protocole IP**, raison pour laquelle les adresses logiques sont aussi appelées des **adresses IP**. Il existe actuellement deux versions du protocole IP : IPv4 et IPv6. Ne cherchez pas la version 5 d'IP, elle n'existe pas. Elle a été envisagée, prototypée, mais abandonnée pour ensuite renaître, sous une forme particulièrement remaniée, en IPv6.

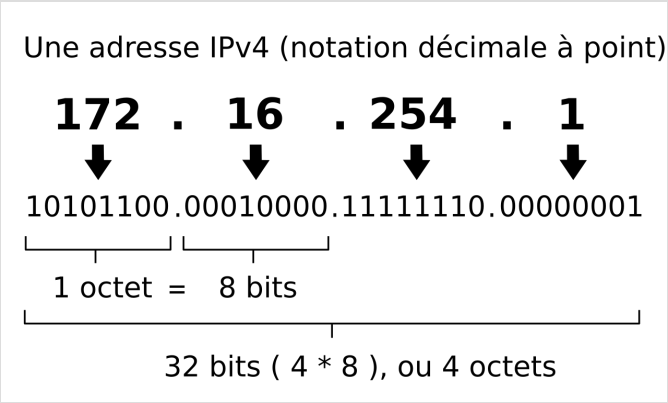
L'adresse IP doit non seulement indiquer l'ordinateur dans le réseau local, mais aussi quel est le réseau voulu : cela fait une information en plus comparé à l'adresse physique. Les adresses IP sont donc décomposées en deux portions : un **préfixe réseau** qui indique le réseau local adressé et un **suffixe hôte** qui adresse l'ordinateur dans ce réseau. Il est évident que toutes les machines d'un réseau ont des adresses avec le même préfixe réseau : la seule différence entre ces adresses sera le suffixe hôte. Par convention, les bits de poids forts forment le préfixe réseau, alors que les bits de poids faible forment le suffixe réseau. Parmi les adresses d'un réseau, deux sont spéciales :

- l'**adresse de diffusion** (*broadcast*) est la plus grande adresse possible qui appartient au réseau : quand on envoie une donnée à cette adresse, elle est envoyée à tous les ordinateurs du réseau ;
- l'**adresse de réseau** est l'adresse qui sert à identifier le réseau : c'est la plus petite adresse du réseau (le suffixe hôte est à 0) ;
- les autres adresses servent à identifier des machines connectées au réseau.

Ce que je viens de dire pourrait faire croire qu'on obtient l'adresse logique en concaténant l'adresse physique à un identifiant de réseau. Ce n'est pas du tout ce qui est fait avec les adresses IP : l'adresse logique n'a aucun lien avec l'adresse physique. Il faut donc garder une table de correspondance entre IP et adresse MAC pour chaque réseau local (la table en question est la table ARP). La subdivision entre préfixe réseau et suffixe hôte est réalisée par des méthodes assez spéciales, qu'il est important de bien comprendre. Le reste du chapitre sera consacré à ces méthodes.

Les adresses IPv4

En IPv4, les adresses font exactement 32 bits (4 octets) : de quoi coder 4 294 967 296 adresses IP différentes. La notation des adresses IP est normalisée : on doit noter chaque octet en décimal et les séparer par des points.



Adresse Ipv4

Les adresses IPv4 réservées

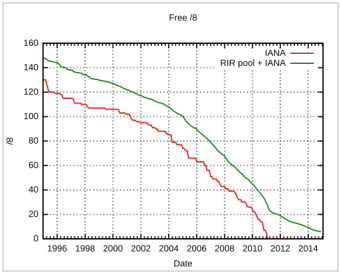
Si une adresse IPv4 identifie un ordinateur dans le cas général, certaines adresses IPv4 sont spéciales et réservées pour des utilisations particulières. Si l'on omet les adresses de réseaux privés (nous reviendrons dessus tout à l'heure), la liste des adresses spéciales est la suivante :

Intervalle d'adresses	Description
0.0.0.0/8	Réseau actuel.
100.64.0.0/10	Espace d'adresses partagé
127.0.0.0/8	Adresses de loopback
169.254.0.0/16	Adresses Link-local
192.0.0.0/24	Adresses réservées au protocole IETF
192.0.2.0/24	Adresses TEST-NET-1
192.88.99.0/24	Adresses utilisées pour la compatibilité entre IPv4 et IPv6
198.18.0.0/15	Adresses pour benchmarks réseaux
198.51.100.0/24	Adresses TEST-NET-2
203.0.113.0/24	Adresses TEST-NET-3
224.0.0.0/4	Adresses routées en multicast
240.0.0.0/4	Adresses de classe E, réservées
255.255.255.255	Adresse de diffusion

Les adresses comprises entre 127.0.0.1 (inclue) et 128.255.255.255 (inclue) sont assez intéressantes à étudier. Ces adresses ne sont pas des adresses IP d'Internet, mais correspondent à l'ordinateur local (celui sur lequel vous travaillez, par exemple). L'adresse la plus importante est clairement l'adresse **127.0.0.1**. Si vous envoyez un paquet sur cette adresse, le paquet ne quittera pas l'ordinateur émetteur et ne sera pas envoyé sur le réseau. À la place, il passera directement de la mémoire d'émission (où les paquets sont mis en attente avant envoi) vers la mémoire de réception (là où sont stocké les paquets réceptionnés en attente de traitement). On pourrait croire qu'elle est inutile, mais elle est extrêmement importante pour diagnostiquer certaines pannes réseau. Elle permet de vérifier que l'ordinateur fonctionne correctement en cas de panne. Si l'envoi d'un paquet sur cette adresse réussit, alors l'ordinateur n'est pas responsable de la panne et le coupable est ailleurs. Mais si l'envoi échoue (on ne récupère pas le paquet envoyé), alors la panne est à chercher sur l'ordinateur testé (logiciel défectueux, mauvaise installation de Windows, problème de pilote de carte réseau, ou autre).

La pénurie d'IPv4 et ses solutions

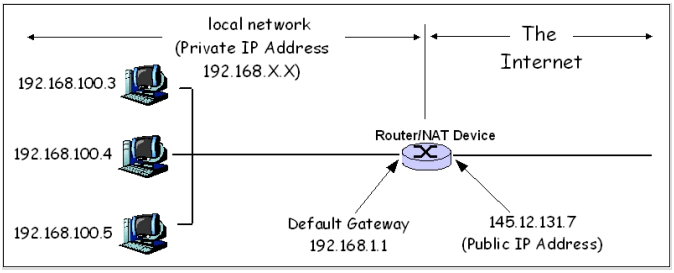
Depuis les années 2000, on n'a plus assez d'adresses IPv4 pour combler les besoins du monde entier. Diverses mesures ont donc étés prises pour faire perdurer l'IPv4 durant quelques décennies. Le NAT (*Network Address Translation*) en est une. Celle-ci se base sur des adresses qui ne peuvent pas être routées sur internet et sont des adresses IP internes à un réseau local. Plusieurs équipements peuvent utiliser ces adresses, à condition qu'ils soient dans des réseaux locaux différents. Ces adresses sont appelées des **adresses privées**, les adresses IP normales étant appelées des adresses IP publiques. Voici les intervalles d'adresses privées :



Pourcentage d'adresses IPv4 libres restantes, en fonction du temps.

Adresse de base et masque	Intervalle d'adresses
10.0.0.0/8	10.0.0.0 – 10.255.255.255
172.16.0.0/12	172.16.0.0 – 172.31.255.255
192.168.0.0/16	192.168.0.0 – 192.168.255.255

Le NAT permet d'attribuer des adresses privées à des équipements qui doivent communiquer sur internet. Le réseau local étant obligatoirement connecté sur internet via un routeur, celui-ci a une adresse IP publique. Le NAT permet de router des paquets émis par des ordinateurs d'un réseau local en se faisant passer pour le routeur : l'adresse privée de l'émetteur du paquet sera remplacée par l'adresse publique du routeur. Il permet aussi de rediriger les paquets entrants, qui sont routés vers l'adresse IP publique du routeur, vers l'ordinateur destinataire : l'adresse de destination est remplacée par l'adresse privée de destination par le routeur avant d'être routée sur le réseau local.

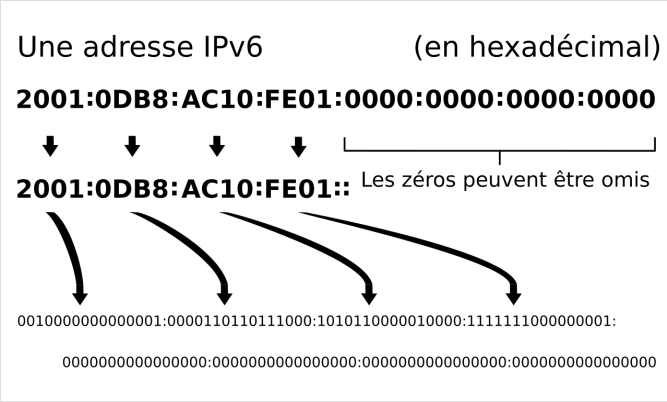


Exemple de réseau qui utilise le NAT (*Network Address Translation*).

Les adresses IPv6

En IPv6, les adresses font quatre fois plus que les adresses IPv4 : exactement 128 bits, à savoir 16 octets. Plus de 256 milliards de milliards de milliards de milliards d'adresses IP différentes. Autant dire qu'on a le temps de voir venir la prochaine pénurie !

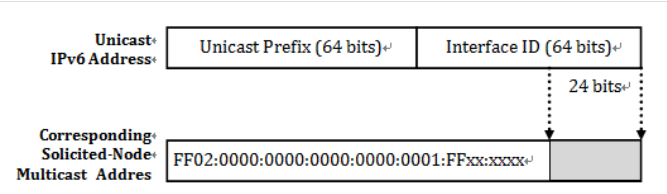
Pour noter une adresse IPv6, on n'utilise pas des nombres écrits en décimal : les octets sont notés en hexadécimal, pour économiser de la place. De plus, la séparation utilise le symbole deux-points (:) pour séparer des groupes de deux octets.



Adresse Ipv6.

Le préfixe réseau

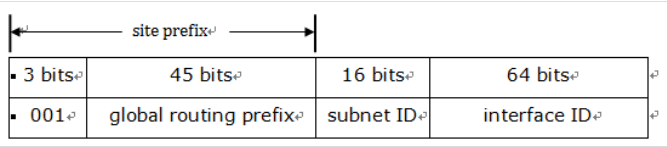
Il faut signaler que le suffixe hôte d'une adresse IPv6 prend au moins la moitié de l'adresse IP, à savoir 64 bits. Évidemment, le reste est utilisé pour le préfixe réseau. Elle peut faire plus dans quelques cas, mais la quasi-totalité des adresses de machines ont systématiquement des suffixes et préfixes de 64 bits.



Structure d'une adresse IPv6 unicast.

Sur la majorité des adresses IPv6 unicast, le préfixe réseau est subdivisé en deux sections :

- une section de 48 bits, qui commence systématiquement par 001 : le préfixe de routage global ;
- et une section de 16 bits qui identifie un sous-réseau précis parmi l'espace de routage global.

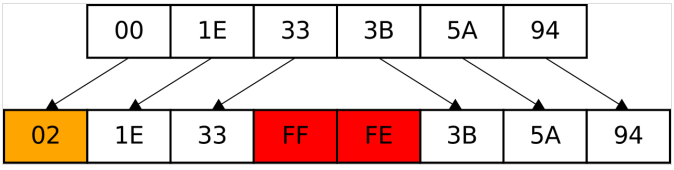


Structure d'une adresse IPv6 unicast.

Le suffixe hôte

Le suffixe hôte est une portion d'adresse IP assez importante. À l'heure actuelle, ce suffixe doit être généré pour chaque ordinateur/équipement réseau. En IPv4, la génération du suffixe hôte passait par un protocole spécialisé, appelé DHCP. Ce protocole, bien que très bien fait, entraînait cependant une certaine complexité dans la gestion des adresses IP. En IPv6, le protocole DHCP est toujours utilisable, même s'il existe des méthodes beaucoup plus simples pour générer le suffixe hôte. Dans les grandes lignes, la génération de l'adresse IPv6 (de son suffixe hôte) est réalisée soit en utilisant l'EUI-64, soit en tirant un nombre aléatoire, soit en utilisant DHCPv6. Il est aussi possible de configurer manuellement l'adresse IP, ce que tout administrateur réseau sait faire avec les outils adaptés.

La première technique utilise l'adresse MAC de l'ordinateur pour dériver une adresse IPv6. Évidemment, l'adresse MAC subit quelques transformations. En effet, l'adresse MAC de 48 bits ne permet pas d'obtenir un suffixe hôte IPV6 de 64 bits : l'adresse MAC ne fournit que 6 sur les 8 nécessaires. Les deux octets manquant sont remplis par une valeur par défaut : FFFE (en hexadécimal), soit 255. 254 en décimal. Ces deux octets sont insérés au milieu de l'adresse MAC, entre les trois premiers octets et les trois derniers. Enfin, en guise de dernière modification, le 7ème bit du premier octet est inversé. Ce bit est le bit U du couple U/L. Dans la majorité des cas, le premier octet de l'adresse MAC, dont la valeur est souvent 00, passe alors à 02. L'utilisation de l'EUI-64 pose des problèmes de confidentialité, vu que l'adresse MAC peut être déduite de l'adresse IPv6 d'un ordinateur. D'où le fait que d'autres techniques existent pour dériver le suffixe hôte. La méthode idéale est de fabriquer celle-ci de manière aléatoire. Une méthode serait par exemple d'utiliser un algorithme comme MD5 ou SHA-1 sur l'adresse EUI-64, afin de camoufler celle-ci.



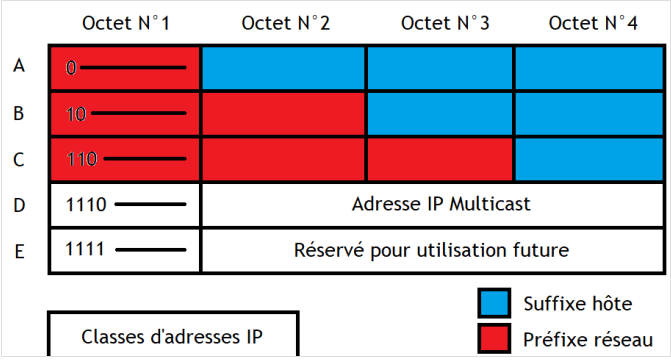
EUI 64

Les masques de sous-réseau

Dans le chapitre précédent, nous avons vu que toute adresse IP est subdivisée en deux portions : un préfixe réseau qui indique dans quel réseau/sous-réseau se situe l'ordinateur, et un suffixe hôte qui indique de quelle est la place de l'ordinateur dans le dit réseau. On peut se demander comment est fait ce découpage, comment on sait que telle IP se décompose ainsi. C'est le but de ce chapitre que de vous expliquer comment on sait quelle portion d'une adresse est le préfixe réseau et l'autre le suffixe hôte. De plus, nous allons voir comment les administrateurs réseaux font pour configurer l'adresse IP ainsi, comment ils configurent la séparation entre préfixe et suffixe lors de l'installation d'un réseau. Tout cela est réalisé par ce qu'on appelle un masque de sous-réseau.

Les classes d'adresse IP

Historiquement, la première méthode utilisée pour séparer suffixe et préfixe se basait sur un cadre assez rigide. On ne pouvait placer la démarcation qu'à des endroits bien précis : les seuls préfixes réseaux autorisés avaient une taille de 1 octet, 2 octets et 3 octets. Les premiers bits de l'adresse IP indiquaient la taille du préfixe réseau, la position de la démarcation. Ce mécanisme, inventé par les concepteurs du protocole IP, est appelé l'adressage par **classes d'adresses IP**. Une classe est un ensemble d'adresses IP défini par un même préfixe réseau. Il en existe cinq types : A (préfixe de 1 octet), B (préfixe de 2 octets), C (préfixe de trois octets), D (adresse *multicast*) et E (usage réservé, inutilisé).



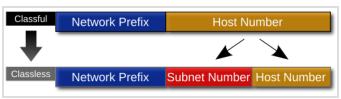
Classes d'adresses IPv4

La taille de chaque classe, les intervalles et nombres d'IP sont indiqués ci-dessous. Les intervalles de chaque classe n'ont pas été choisis au hasard, pas plus que leur nombre d'adresses. Vous remarquerez que chaque classe contient un nombre d'adresse qui est une puissance de deux, sans compter que les intervalles en sont des multiples. Tout cela fait que les premiers bits d'une adresse permettent d'identifier sa classe facilement : il y a une correspondance directe entre les premiers bits et la classe d'une IP.

Classe	Intervalle d'adresses IP	Nombre d'adresses
Classe A	0.0.0.0 - 127.255.255.255	16 777 214 adresses.
Classe B	128.0.0.0 - 191.255.255.255	65 534 adresses.
Classe C	192.0.0.0 - 223.255.255.255	254 adresses.
Classe D	224.0.0.0 - 239.255.255.255	
Classe E	240.0.0.0 - 255.255.255.255	

Cette organisation n'était pas très souple et gâchait pas mal d'adresses IPv4. Imaginez le cas d'une entreprise qui a besoin de 1024 adresses, pour 1024 machines : on lui donnait une adresse de classe B pour son réseau, à savoir 65536 adresses. Il est évident que la quasi-totalité des adresses de ce réseau sont alors inutilisées.

La taille du réseau doit être choisie dans quelques classes de taille fixe et pré-déterminée : on ne peut pas choisir un intervalle d'adresses le plus proche possible du nombre de machines utilisées. Ce fait ne posait pas de problèmes au début des réseaux IP, mais il a commencé à prendre de plus en plus d'ampleur avec le temps. En août 1990, le problème a été soulevé lors de la réunion de l'IETF, un organisme de normalisation qui gère notamment IP. Divers groupes de travail se sont rassemblés et ont commencé à réfléchir sur le sujet. La première solution, le *sub-netting*, est apparue dès 1985. La seconde solution technique, apparue en novembre 1992 dans une réunion de l'IESG, est encore en vigueur aujourd'hui (2021).



Subnetting.

Avec le ***sub-netting***, il est devenu possible d'utiliser une classe d'adresse complète pour plusieurs réseaux distincts. Prenons par exemple une entreprise qui dispose de quatre réseaux distincts, contenant chacun respectivement 5000, 4000, 2000 et 6000 machines. Avec l'adressage par classe strict, il faudrait donner à chaque réseau un préfixe réseau de classe B, soit 65535 adresses différentes. Or, ces 65535 adresses suffisent à elles seules pour les quatre réseaux, mais l'adressage par classe strict ne le permet pas. C'est ce genre de problème que permet de résoudre le *sub-netting* : il permet de découper une classe d'adresse en sous-classes indépendantes. La classe d'adresse A/B/C correspond à un réseau complet, tandis que chaque sous-classe correspondra à ce qu'on appelle un sous-réseau. Pour cela, le suffixe hôte de l'adresse est découpé en deux morceaux : un qui identifie le sous-réseau et l'autre qui identifie la machine dans le sous-réseau. Le sub-netting permet de définir le partage en spécifiant le nombre de bits pour le sous-réseau, les bits restant étant utilisés pour la machine. C'est à l'administrateur réseau de l'entreprise ou de l'administration que revient la définition de cette configuration.

L'adressage CIDR

De nos jours, ces classes de réseaux ont laissé la place à un système plus souple : l'adressage **Classless Inter-Domain Routing** (CIDR). Avec cette méthode, on peut découper une adresse en identifiant hôte et préfixe réseau où l'on veut dans l'adresse. Cela résout le problème de l'adressage par classe : on peut mettre juste ce qu'il faut d'adresses dans le réseau (à peu près, on va dire). La subdivision de l'adresse en préfixe réseau et suffixe hôte est indiquée par un masque de sous-réseau, un nombre entier de la même taille que l'IP. Si on superpose le masque et l'IP, on peut deviner le préfixe réseau et le suffixe hôte : les bits à 1 dans le masque indiquent les bits du préfixe réseau, alors que les bits à 0 indiquent le suffixe hôte. Dit autrement, on obtient l'adresse réseau en faisant un ET logique entre l'IP et le masque.

Ce masque n'a pas besoin d'être envoyé dans les paquets de données, en même temps que l'IP. En effet, ce masque sert à savoir si une adresse IP appartient à un réseau local. Quand la donnée est transférer de proche en proche sur le net, chaque réseau local va tester si l'IP qu'il vient de recevoir correspond à une de ses machines. Pour cela, il a besoin du masque, pour vérifier si L'IP et celle du réseau local ont le même préfixe réseau : si ce n'est pas le cas, la machine n'appartient pas au réseau, et elle est routée ailleurs.

La validité d'un masque de sous-réseau

Un masque valide doit respecter deux contraintes : les bits à 1 sont tous contiguës et regroupés dans les bits de gauche. Ce qui fait qu'un masque doit ressembler à quelque chose dans le genre :

- 11111111 11111111 11111111 00000000 ;
- 11111111 11111000 00000000 00000000 ;
- 11111111 11111111 11111111 11110000.

On peut facilement déduire des contraintes sur le masque les 3 règles suivantes :

- seuls les octets suivants sont valides pour un masque de sous-réseau :
  - 255 (1111 1111) ,
  - 254 (1111 1110) ,
  - 252 (1111 1100) ,
  - 248 (1111 1000) ,
  - 240 (1111 0000) ,
  - 224 (1110 0000) ,
  - 192 (1100 0000) ,
  - 128 (1000 0000) ,
  - et 0 (0000 0000) ;
- Un nombre autre que 0 ne peut être précédé que de nombres valant 255 ;
- Un nombre autre que 255 ne peut être suivi que de nombres valant 0.

Les notations des masques de sous-réseau

On peut noter ces masques de plusieurs manières équivalentes :

- soit on écrit le masque en binaire ;
- soit on écrit chaque octet en décimal et on les sépare par des points (cela permet de gagner de la place) ;
- soit on indique le nombre de bits à 1 dans le masque, précédé d'un "/" (les contraintes vues juste avant permettent alors de retrouver le masque facilement).

Par exemple, les notations suivantes sont équivalentes (c'est le même masque) :

- 11111111 11111111 11111111 00000000 ;
- 255 . 255 . 255 . 0 ;
- /24.

De même pour :

- 11111111 11111111 00000000 00000000 ;
- 255 . 255 . 0 . 0 ;
- /16.

De même pour :

- 11111111 11100000 00000000 00000000 ;
- 255 . 224. 0 . 0 ;
- /11.

Les datagrammes IP

Au niveau de la couche réseau, les données sont envoyées par **paquets**, composés d'une trame (de couche liaison) à laquelle on ajoute un en-tête de la couche réseau. Le protocole de la couche réseau est le protocole IP, qui a donné son nom aux adresses IP mentionnées dans le chapitre précédent. Dans ce chapitre, nous allons voir comment est structuré l'en-tête IP et nous verrons comment les paquets sont formés et structurés. Nous parlerons aussi du phénomène de fragmentation des paquets IP.

La taille maximale des paquets

En théorie, rien ne limite la taille d'un paquet IP : on peut mettre autant de données qu'on veut derrière l'en-tête. Mais dans la pratique, la taille d'un paquet n'est pas infinie. Les divers équipements réseau n'ont pas une mémoire infinie et ont une limite fixe, qu'il faut prendre en compte.

Le *Maximum Transmission Unit*

Chaque réseau/routeur/ordinateur/... a une taille de paquet maximale qu'il accepte et/ou peut gérer : le ***Maximum Transmission Unit*** (unité de transmission maximale), ou MTU. Précisons que le MTU comprend les données utiles et l'en-tête (qui est inclut dans le MTU).

Exemples de MTU pour divers protocoles réseau.

Type de réseau	MTU (en octets)
Arpanet (ancètre d'Internet)	1 000
Ethernet	1 500
FDDI	4 470

Plus le MTU est grand, plus la transmission est efficace. Pour être plus précis, il faut introduire le rapport suivant :

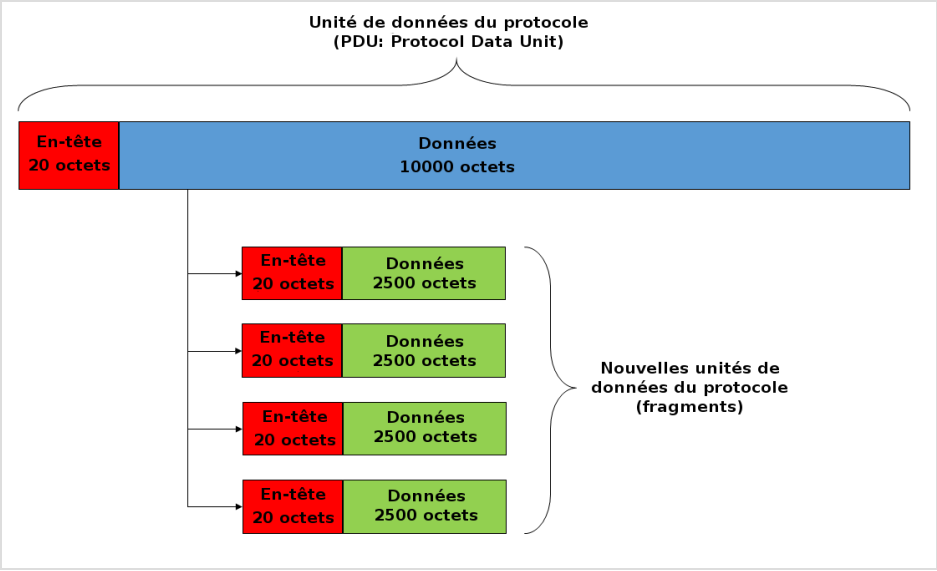
$$\frac{taille\ des\ donnees\ utiles}{taille\ totale\ des\ donnees} = \frac{donnees}{en-tete\ IP + donnees} = \frac{donnees}{MTU}$$

Plus le MTU est grand, plus ce rapport augmente. On peut le comprendre assez intuitivement : plus le MTU est grand, moins l'en-tête prendra une proportion importante du MTU (rappelons que l'en-tête a une taille fixe).

La fragmentation des paquets IP

Il est possible qu'un paquet IP doive passer par un réseau dont le MTU est trop petit pour lui. Cela arrive parfois au niveau des routeurs, lors du passage d'un réseau à un autre, à condition que les deux réseaux connectés aient des MTU différents. Si les deux réseaux connectés par un routeur ont un MTU identique, il n'y a pas de problème et le paquet IP est routé sans encombres. Si un paquet dépasse le MTU, il ne peut pas être transmis tel quel. Mais la transmission n'est pas interrompue ou annulée pour autant. À la place, le routeur utilise une procédure de **fragmentation** du paquet.

Le nom trahit ce que fait le routeur : il découpe le paquet en plusieurs mini-paquets compatibles avec le MTU de l'équipement, qui sont envoyés un par un. Les mini-paquets créés par le routeur sont appelés des **fragments**. Notons que l'en-tête du paquet IP originel est recopié au début de chaque fragment, sans quoi ceux-ci ne pourraient pas être routés correctement. De plus, les fragments sont envoyés uns par uns par le routeur, si possible dans l'ordre. Mais il se peut que les paquets arrivent dans le désordre au récepteur. Le routeur doit donc, en plus de fragmenter le paquet, ajouter des informations d'ordre à chaque fragment. Ces informations permettent au récepteur de remettre les fragments dans l'ordre, pour reconstruire le paquet initial.



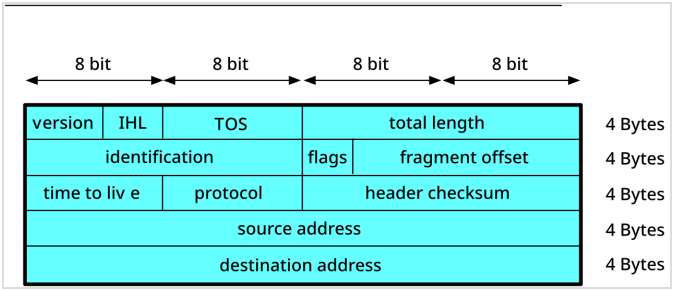
Fragmentation d'un paquet réseau (illustration en anglais).

L'en-tête du protocole IP

Commençons par aborder l'en-tête du protocole IP. Précisons en premier lieu que celui-ci n'est pas le même selon que l'on parle d'en-tête Ipv4 ou IPv6. Il existe quelques différences, certes mineures, mais qui rendent l'exposé un peu difficile. On peut cependant noter qu'il existe quelques régularités qui sont valables aussi bien en IPv4 qu'en IPv6. Par exemple, l'en-tête contient systématiquement l'adresse IP de l'émetteur et celle du récepteur.

L'en-tête du protocole IPv4

L'en-tête d'IPv4 est illustré à droite et chaque champ est expliqué ci-dessous. Dans les grandes lignes, on peut décomposer l'en-tête IPv4 en deux sections : une section de commande qui contient diverses options, et un champ d'adresse dans lequel on trouve les adresses IP de l'émetteur et du récepteur. L'ensemble des champs est codée sur minimum 20 octets, soit 160 bits, mais peut aussi prendre plus de place sur certains paquets.



En-tête IPv4.

La **version du protocole** est codée sur 4 bits. Ce champ sert à préciser si le segment est un segment IPv4 ou IPv6.

Les quatre bits suivants, notés IHL, indiquent la **longueur de l'en-tête**. Vu que l'IHL fait 4 bits, on en déduit que la longueur de l'en-tête est comprise entre 0 et 15 unités. Vous serez peut-être étonné, sachant que l'en-tête fait 20 octets, mais c'est parce que vous ne savez pas que cette longueur n'est pas exprimée en bits ou en octets, mais en paquets de 32 bits (4 octets). Ce qui fait donc une longueur totale théorique comprise entre 0 et 60 octets, pour l'en-tête. Dans les faits, l'en-tête a toujours une taille comprise entre 20 et 40 octets, par construction, ce qui fait que le champ IHL est toujours compris entre 5 et 15.

Le champ **type de service** est décrit par plusieurs RFC et ce champ est passé par trois versions différentes. Il sert pour la qualité de service, à savoir la gestion de la saturation du débit d'un nœud du réseau. Il est utilisé quand un routeur est saturé de paquets IP et qu'il ne peut plus en mettre en attente. Ce phénomène n'est pas rare et a reçu le nom de *congestion réseau*. Gérer une telle situation peut se faire de plusieurs manières. La plus simple est de supprimer certains paquets en attente, quitte à prévenir l'émetteur. En théorie, les paquets sont choisis de manière aléatoire, mais il est possible de privilégier certains types de paquets sur les autres. Pour cela, il faut créer plusieurs classes de paquets, de priorités différentes : les paquets de faible priorité sont supprimés en premier, alors que ceux de forte priorité sont les derniers à être supprimés. La priorité du paquet est indiquée par un numéro de six bits, le DSCP, qui est placé dans le champ *type de service*. Les deux bits restants sont utilisés pour autre chose, toujours pour gérer la congestion réseau.

Le champ **longueur totale** donne la longueur totale du segment, en-tête compris. Il est codé sur 16 bits, ce qui permet d'utiliser des segments de 64 kibioctets (65536 octets). Dans la réalité, les protocoles de couche 2 ne supportent pas des paquets aussi gros. Par exemple, le protocole Ethernet a une limite à 1500 octets par paquet, appelée MTU. Pour être compatibles avec une telle taille, les segments doivent donc être découpés en sous-segments compatibles avec le MTU. Ces sous-segments sont appelés des fragments. On appelle une telle opération la fragmentation.

Quand un segment est fragmenté en sous-segment, un problème se pose à la réception : comment remettre en ordre les morceaux et comment identifier les fragments ? C'est justement à cela que sert le champ suivant, nommé **identification**. Tous les fragments d'un même segment se voient attribuer le même identifiant, un numéro qui indique qu'ils sont du même segment.

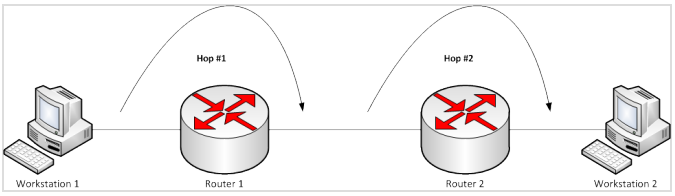
Le champ suivant, les **indicateurs**, indique si le paquet reçu a été fragmenté, si des fragments vont suivre, et ainsi de suite. Il est codé sur trois bits :

Champ indicateurs (*flags*) de l'en-tête IPv4.

- Le premier bit est réservé et reste à 0 ;
- Le second bit est appelé le bit ***Do Not Fragment***. Comme son nom l'indique, il précise si le paquet peut être fragmenté ou si cette opération est interdite. S'il est à 0, la fragmentation est autorisée si nécessaire, alors qu'un 1 dit que la fragmentation est interdite.
- Le troisième bit précise si le paquet a déjà été fragmenté et si ce fragment est le dernier d'entre eux. Son nom est : bit ***More Fragments***.

La position de chaque fragment dans le segment est quant à elle indiquée par un numéro, le **fragment offset**, présent à la suite des indicateurs.

Le champ suivant, le **TTL** est décrémenté quand il passe dans un routeur ou une machine. Quand il tombe à zéro, le segment est tout simplement abandonné, éliminé. Cela permet d'éviter qu'un segment soit routé indéfiniment. Le TTL étant codé sur un octet, sa valeur maximale est de 255 intermédiaires de routage.



Nombre de routeurs intermédiaires entre émetteur et récepteurs. Du fait de l'existence du TTL, il ne peut pas dépasser 255.

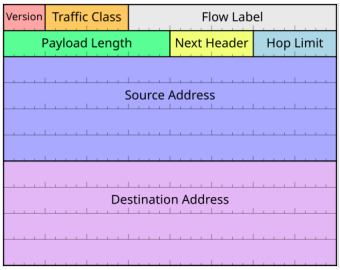
Le champ **protocole** sert à indiquer s'il s'agit d'un segment TCP ou UDP. Il précise quel est le protocole de couche transport qui a généré le paquet.

Le reste de la section de commande est composé de bits de détection/correction d'erreurs.

Enfin, on trouve les adresses IP source et de destination.

L'en-tête du protocole IPv6

L'en-tête utilisé par le protocole IPv6 est illustré ci-contre. On voit rapidement que l'en-tête IPv6 est plus simple que l'en-tête IPv4 : le nombre de champ est plus faible, et leur signification est plus facile à appréhender.



En-tête IPv6.

Comme pour IPv4, le tout premier champ donne la **version du protocole** utilisé, ici IPv6. Vu qu'il est codé sur 4 bits, il peut prendre 16 valeurs, mais la plupart n'a pas de signification. Seules les valeurs suivantes sont autorisées :

- 00 – Réservé ;
- 01 – Non assigné ;
- 04 – IP V4 ;
- 05 – ST Datagram Mode ;
- 06 – IP V6, 15 – Réservé.

Le champ **Traffic Class**, sur 8 bits, donne la priorité du paquet. Il sert dans les scénario qui requièrent une certaine qualité de service.

Le champ **Flow Label**, codé sur 20 bits est utilisé pour coder des séquences de paquets qui doivent subir un traitement spécial. Ce champ stocke le numéro du paquet dans la séquence.

Le champ **Payload Length** encode la longueur du paquet, en-tête IPv6 exclu.

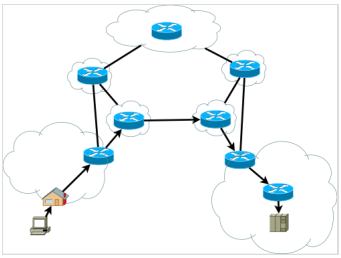
Le champ **Next Header** code le type de données transmises dans le paquet : est-ce un paquet TCP, UDP, ICMP, ou autre ? Pour faire simple, il remplace le champ protocole d'IPv4.

Le champ **Hop Limit** est l'exact copie du champ TTL d'IPv4.

Enfin, les adresses source et destination sont placées en fin de paquet.

Le matériel de couche réseau : les routeurs

Il est maintenant temps de laisser les réseaux locaux derrière nous et de passer aux réseaux composés d'une interconnexion de réseaux locaux plus simples. Internet est l'un d'entre eux, mais il ne faut pas oublier l'ensemble des réseaux étendus. On a vu au début du cours que, sur Internet, les paquets sont propagés de proche en proche, d'intermédiaire en intermédiaire, jusqu'à la destination. Cette propagation doit cependant être gérée, histoire que la donnée arrive bien à destination. Déterminer quel est le chemin que doit parcourir la donnée pour arriver la destination est ce qu'on appelle le **routage**.


Routage.

Sur Internet, les intermédiaires en question, qui propagent les paquets et s'occupent du routage, s'appellent des **routeurs**. À la réception d'un paquet, le routeur prend une décision et décide vers quel routeur ou ordinateur il doit propager la donnée. Il n'y a pas de serveur central qui déciderait comment router la donnée. En conséquence, cette opération demande des ressources matérielles pour décider vers quel voisin il faut envoyer la donnée. Et cela demande du temps de calcul, de la mémoire, et potentiellement d'autres ressources.

La table de routage

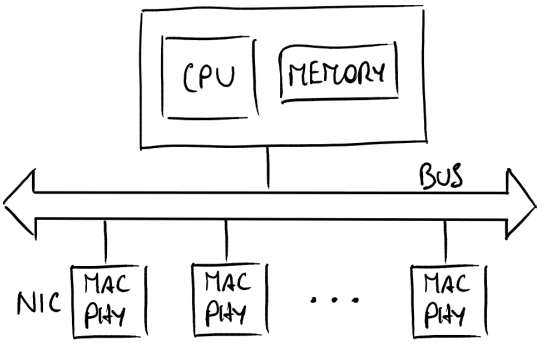
Un **routeur** peut être vu comme l'équivalent d'un commutateur, mais pour une interconnexion de réseaux : là où le commutateur connecte des machines dans un même réseau local, le routeur connecte et sert d'interface à deux réseaux différents. Il reçoit des paquets sur certains ports, et les renvoie sur d'autres : il doit juste envoyer les paquets reçus vers le meilleur port de sortie, celui qui rapprochera le paquet de sa destination. Pour cela, le routeur doit savoir quelle est la meilleure sortie pour chaque adresse IP de destination possible. Du moins, c'est la théorie, vu que le routeur peut compresser ces informations de différentes manières.

Les routeurs sont similaires aux commutateurs, si ce n'est qu'ils gèrent des adresses IP au lieu d'adresses MAC. Ils reçoivent des trames sur un port d'entrée, trames qui destinées à une adresse IP de destination. Cette trame doit être envoyée à l'ordinateur de destination, et donc envoyée sur un des ports de sortie du routeur, celui qui mènera ultimement la trame à destination. Son fonctionnement est similaire à celui d'un commutateur amélioré. Dans les grandes lignes, la table CAM est remplacée par une **table de routage**, qui associe une adresse IP de destination au port de sortie qui correspond. Le reste de l'architecture interne du routeur est basée soit sur un bus, soit sur une *switch fabric*.

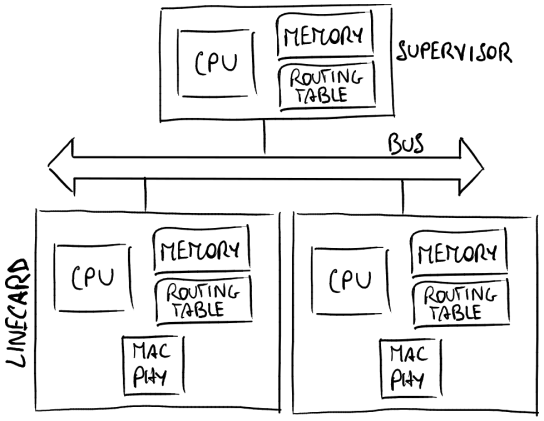
Quoiqu'il en soit, le routeur doit bel et bien garder des correspondance entre une adresse IP de destination et le numéro du port sur lequel il doit envoyer le paquet. Tout cela est mémorisé dans une sorte de mémoire RAM : la table de routage. Cependant, ces tables de routage sont rarement complètes : elles ont une taille limitée, et elles ne peuvent pas mémoriser toutes les correspondances possibles et imaginables. Et cela peut poser quelques problèmes. Mettons-nous dans le cas où un routeur doit router un paquet vers une IP de destination, et où le routeur n'a pas de correspondance pour cette IP dans sa table de routage. Celui-ci ne sait pas sur quel port il doit router le paquet. Mais le routeur a une solution : router le paquet vers une route par défaut, vers un port choisi par défaut en cas d'absence de correspondance.

Les générations de routeurs

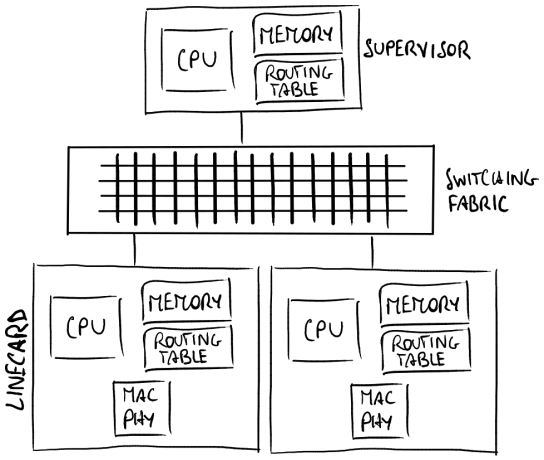
Les tout premiers routeurs, dits de première génération, relient leurs ports d'entrée et de sortie avec un bus. ils contiennent aussi un processeur tout ce qu'il y a de plus normal pour traiter les trames IP, ainsi qu'une mémoire RAM pour stocker les trames et la table de routage. Chaque port est relié à de circuits chargés de gérer le port. Ces circuits reçoivent des trames, les envoient et effectuent quelques traitements basiques. Ils gèrent notamment tout ce qui a trait aux adresses MAC. Une fois que ces circuits ont fait leur office, ils envoient la trame traitée sur le bus interne au routeur. La trame est alors réceptionnée par le processeur, éventuellement stockée en mémoire RAM. Celui-ci accède alors à la table de routage, pour identifier le port de sortie. Enfin, le processeur envoie la trame vers le port de sortie qu'il a déduit de ses traitements. La trame est alors envoyée sur le réseau. Le défaut principal de ce type de routeur est que les transferts en direction du processeur principal saturent le bus dans certaines situations critiques.



Les routeurs de seconde génération sont plus complexes. Ceux-ci multiplient le processeur, la RAM et la table de routage en plusieurs exemplaires : un exemplaire par port. Ainsi, les trames reçues sur un port sont directement traités dans les circuits de gestion de ce port. Une fois traitée, elles sont envoyées directement sur le port de sortie, et envoyée immédiatement sur le réseau. Cependant, cela ne vaut que pour des trames simples. Les trames plus complexes doivent être traitées par un processeur plus complexe, non-attaché à un port. Ce processeur, le superviseur, est unique dans le routeur.



Cependant, les deux types de routeurs précédents utilisent un bus pour afin de faire communiquer les différents composants. Or, il se peut que les conflits d'accès au bus minent les performances. Pour éviter cela, certains routeurs remplacent le bus par une *switch fabric*, pour gagner en performance. Ainsi, les transfert n'entrent pas en conflit pour l'accès à un unique bus, chaque transfert pouvant se faire en parallèle des autres.



Les protocoles de routage

Le routage sur internet n'est pas effectué par une instance centralisée, vu la taille du réseau et son organisation. Difficile d'organiser autour d'un seul protocole ce qui est avant tout une interconnexion de réseaux très différents. Il est proprement impossible de décider de la table de routage de chaque routeur tant il y en a, sans même parler de propager les mises à jour des tables de routage. En réalité, Internet est organisé autour de réseaux qui sont chacun "indépendants" en terme de routage : les ***Autonomous systems***, abréviés AS. Un AS est un ensemble de réseaux et de routeurs reliés entre eux, qui sont soumis à un même protocole de routage. Ceux-ci sont souvent soumis à une même entité commerciale ou administrative : par exemple, chaque fournisseur d'accès possède son propre AS. Chaque AS est identifié par un numéro, l'*Autonomous System Number* (ASN), attribué par diverses organisations internationales. Tous identifient un AS, à l'exception de quelques numéros dédiés à des usages un peu particuliers, dans les protocoles de routage par exemple.

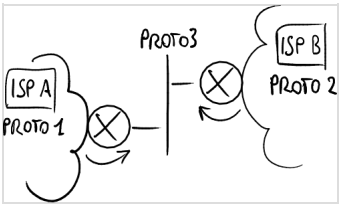


Illustration de la communication entre deux *Autonomous Systems*, chacun appartenant à un fournisseur d'accès (ISP).

Le routage est cohérent à l'intérieur d'un AS, alors qu'il ne l'est pas entre les AS, et cela se ressent dans les algorithmes de routage utilisés. Les AS utilisent des protocoles de type IGP (*Interior Gateway Protocols*) pour mettre à jour les tables de routage, alors que la communication entre AS est réalisée par des protocoles de type EGP (*Exterior Gateway Protocols*)/BGP (*Border Gateway Protocols*).

Les différentes formes de routage

Évidemment, il existe plusieurs manières de router les paquets à destination, qui sont implémentées par divers standards. Et on peut classer les différentes méthodes de routage en plusieurs catégories. Dans ce qui va suivre, nous allons donner quelques critères qui permettent de classer les protocoles de routage.

Routage statique et dynamique

Pour savoir où envoyer les paquets reçus, les relais du réseau contiennent une table qui associe chaque adresse avec le prochain intermédiaire. Cette table porte des noms différents selon que l'on parle de routeurs ou de commutateurs : table de routage pour les routeurs et table CAM pour les commutateurs. Nous allons ici parler de la table de routage et de sa mise à jour. Nous en reparlerons plus en détail dans le prochain chapitre. Pour le moment, nous allons simplement dire qu'elle permet de savoir où envoyer chaque paquet reçu. Nous n'avons pas besoin de plus.

Le contenu de la table de routage peut être déterminé à l'avance par les concepteurs du réseau, ou mis à jour régulièrement (pour s'adapter à des ajouts ou retraits de machines). Dans le premier cas, les tables de routage sont remplies lors de l'allumage du routeur, et ne sont jamais mises à jour. On parle alors de **routage statique**, dans le sens où il ne peut pas évoluer sans que le gestionnaire du réseau ne fasse les modifications adéquates. Bien que très simple, cette approche a cependant de nombreux défauts. Déjà, saisir à la main chaque ligne de la table de routage est parfois long, compliqué, chronophage. Le faire de manière automatisée, chaque routeur construisant la table de routage de lui-même, étant de loin une meilleure solution. De plus, la table de routage ne peut pas s'adapter à une panne de réseau ou au changement de celui-ci. Toute modification des connexions intra-réseau demande de modifier la table de routage, manuellement.

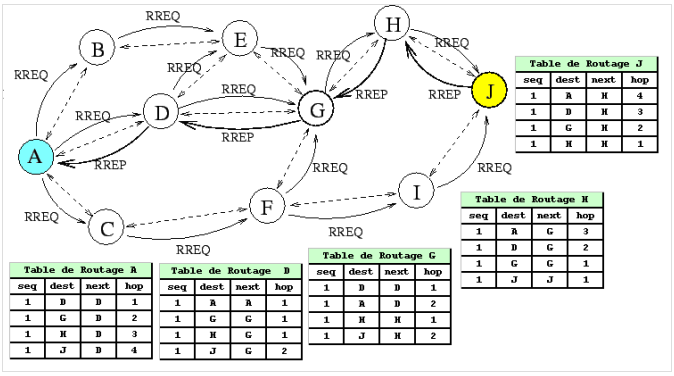
Un **routage dynamique** permet de mettre à jour les tables de routage à la volée, régulièrement, sans intervention humaine. Cette mise à jour des tables de routage est alors prise en charge par un algorithme de routage, une sorte de programme intégré aux routeurs qui leur dit quoi faire pour se mettre à jour. Une sorte d'équivalent des mises à jour Windows, mais pour la table de routage. Ces algorithmes de routage sont pris en charge par des protocoles divers comme IGP ou BGP, que nous n'aborderons pas dans le détail tellement ils sont complexes. Les algorithmes de routage dynamique peuvent repérer les chemins endommagés, qui ne fonctionnent plus (un routeur débranché ou en panne, par exemple), et trouver des routes alternatives. Le réseau se reconfigure à chaque instant pour que le service soit maintenu, et que les performances soient conservées.

Routage centralisé ou distribué

On peut aussi distinguer les méthodes de routage selon la méthode utilisée pour mettre à jour les tables de routage. La mise à jour des tables de routage peut être gouvernée par un routeur central, qui communique aux autres routeurs les informations de mise à jour : on parle alors de **routage centralisé**. Dans l'autre cas, le routage est un **routage décentralisé** : chaque routeur met à jour sa table de routage individuellement, sans intervention d'un routeur central.

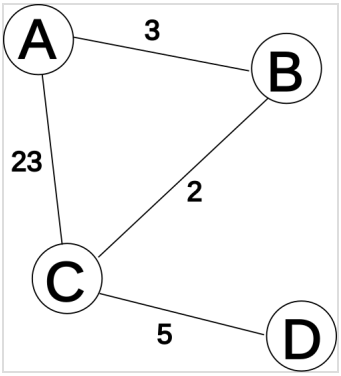
Les algorithmes de routage

Tous les algorithmes de routage se basent sur les mêmes principes mathématiques, à savoir la théorie des graphes. Pour faire simple, ces algorithmes modélisent un réseau sous la forme d'un ensemble de points reliés par des flèches : les points représentent les routeurs et ordinateurs, alors que les flèches indiquent les liens entre ces routeurs. Un exemple de graphe est donné dans le dessin à votre droite. Le but de l'algorithme est de trouver un chemin dans ce graphe qui relie l'émetteur au destinataire. Il existe de nombreux algorithmes pour trouver le chemin le plus court entre deux points d'un graphe, et il n'est pas question d'en faire la liste ici. Cependant, sachez que ceux-ci ne sont pas utilisés tels quels par les algorithmes de routage.



Exemple de réseau représenté sous la forme de graphe, avec les tables de routage qui correspondent.

Les algorithmes de routage peuvent aussi tenir compte des performances des différents chemins entre deux routeurs. La mise à jour des tables routage permet alors de trouver des chemins plus courts ou plus rapides pour acheminer une donnée à une IP précise. Il suffit pour cela de tenir compte des temps de transferts entre routeurs. Pour cela il suffit d'associer à chaque flèche, chaque chemin entre deux routeurs, un poids qui indique sa rapidité. Plus la vitesse de transfert est faible entre ces deux routeurs, plus ce nombre sera fort. Pour chaque chemin identifié, l'algorithme additionne le temps de transfert de chaque flèche. Le but de l'algorithme est de trouver le chemin qui minimise le temps de transfert, qui minimise la somme finale.



Graphe numéroté.

Les deux types principaux de protocoles de routage : *vecteur de distance* et *état de liens*

Dans les grandes lignes, on peut classer les protocoles de routage en deux types principaux, eux-même subdivisés en pleins de sous-types : les protocoles de type vecteur de distance, et les protocoles à état de liens.

Avec les **protocoles à vecteur de distance**, les routeurs envoient régulièrement l'ensemble de leur table de routage aux routeurs voisins, auxquels il est directement connecté. Ceux-ci mettent alors à jour leur propre table de routage en tenant compte des informations envoyées. Évidemment, cela consomme beaucoup de débit réseau, ce qui est un défaut majeur. C'est la raison principale pour laquelle ils ne peuvent pas être utilisés sur des réseaux trop importants : les envois prendraient beaucoup trop de temps avec des grosses tables de routage. De plus, les tables de routage mettent beaucoup de temps avant de se stabiliser, ce qui rend le routage très inefficace au début de l'utilisation du réseau. Encore un autre défaut rédhibitoire.

Les **protocoles à état de lien** sont plus efficaces en terme de débit et de stabilisation des tables de routage. Au lieu d'envoyer la table de routage complète, ils envoient des messages courts, qui donnent des indications sur la connectivité du réseau. De manière générale, ces messages sont de simples tests qui vérifient que le voisin est toujours accessible, connecté. Au bout de plusieurs tentatives échouées, le routeur contacté est considéré comme inaccessible et le routeur envoie une indication aux autres voisins comme quoi il ne peut plus communiquer avec lui. En clair, chaque routeur indique quel est l'état des liens qu'il entretient avec les autres routeurs : il dit qu'il est connecté à tel ou tel routeur, que telle connexion a été déconnectée, etc. Avec ces protocoles, le volume de données transmises est plus faible. Pas besoin d'envoyer une table de routage complète, juste quelques messages courts. Outre la forte réduction de débit utilisé, la vitesse de convergence est plus rapide, pour diverses raisons techniques que omettons volontairement.

Les algorithmes de routage par inondation (aléatoires)

Les algorithmes de **routage par inondation** font un routage complètement aléatoire, où chaque routeur émet les paquets reçus sur toutes ses sorties. Ce qui explique le nom de cet algorithme : le routeur inonde le réseau du paquet reçu. Et aussi bizarre que cela puisse paraitre, cet algorithme garantit que le récepteur recevra le paquet, tant qu'il existe un paquet entre l'émetteur et le récepteur. De plus, cet algorithme réagit parfaitement aux changement du réseau : on peut changer les connexions du réseau sans que cela empêche le paquet d'arriver à destination. Mais les défauts sont assez évidents : beaucoup de bande passante est gâchée pour envoyer un paquet en plusieurs exemplaires.

Pour éviter cela, on peut modifier l'algorithme précédent et n'envoyer le paquet reçu que sur une seule sortie, qui est choisie aléatoirement. Cela évite d'envoyer plusieurs copies d'un même paquet, mais celui-ci prendra parfois un chemin assez tordu et mal commode pour arriver à destination. Le paquet peut se perdre en chemin et mettre beaucoup de temps avant d'arriver. Les performances du réseau sont améliorées, du terme de bande passante, mais pas en temps de latence, qui augmente. L'algorithme en question est appelé **algorithme par inondation sélective**.

La traduction des adresses IP en adresses MAC : ARP et NDP

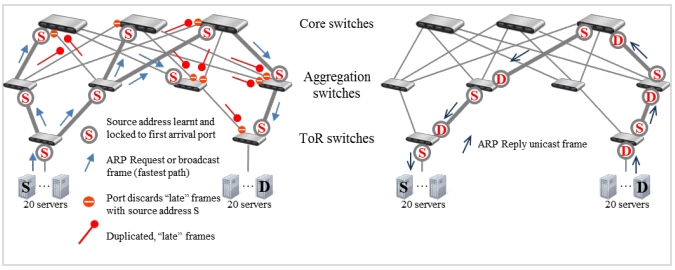
Comment savoir à quelle adresse MAC correspond telle adresse IP ? La solution à ce problème est très simple : il suffit de mémoriser les correspondances entre une IP et une adresse MAC dans une mémoire intégrée au routeur ou au matériel réseau. Cette table de correspondance est appelée le **cache ARP** ou la table ARP suivant le protocole utilisé.

Remplir et mettre à jour le cache ARP est très important : sans cela, certains ordinateurs deviendraient inaccessibles au moindre changement de carte réseau ou d'adresse mac. Il est possible de configurer manuellement le cache ARP des différentes machines d'un réseau, mais cela n'est pas très pratique. Le moindre changement de carte réseau demande de modifier les caches ARP de plusieurs machines, ce qui peut rapidement devenir une gêne notable. Une autre solution est de faire en sorte que le matériel du réseau gère tout cela de lui-même. La gestion du cache ARP est de la responsabilité des ordinateurs et routeurs du réseau, pas besoin de la moindre intervention humaine. Pour cela, les équipements réseaux peuvent utiliser deux protocoles relativement simples : le **protocole ARP** pour les adresses IPV4, ou le **Neighbor Discovery Protocol** pour les adresses IPV6. Ce chapitre va aborder ces deux protocoles, en commençant par le protocole ARP, conceptuellement plus simple.

Le protocole ARP

Imaginons qu'un routeur reçoive un paquet vers une IP pour laquelle il ne connait pas l'adresse MAC (il n'a pas de correspondance IP-MAC dans son cache ARP). Il va alors utiliser une transmission ARP pour découvrir l'adresse MAC de destination. Le processus se déroule en deux étapes : d'abord le routeur envoie une demande aux autres ordinateurs, puis l'ordinateur de destination répond en envoyant son adresse MAC.

1. **Envoi de la demande ARP** : Le routeur va envoyer un message aux ordinateurs du réseau local, sur l'adresse de *broadcast*. Ce message aura la signification suivante : "Quel est l'ordinateur qui a telle adresse IP ?". Ce message contient évidemment l'adresse IP en question, l'adresse MAC de l'émetteur, et quelques autres informations.
2. **Réponse ARP** : L'ordinateur de destination (celui qui a l'IP indiquée dans la demande ARP) répond avec un message de réponse qui signifie : je suis l'ordinateur qui a cette adresse IP. Ce message contient évidemment, l'IP en question, l'adresse MAC du routeur, mais aussi et surtout l'adresse MAC de la machine émettrice. Cette adresse MAC n'est autre que l'adresse MAC qui correspond à l'IP : le routeur a juste à l'utiliser pour mettre à jour sa table ARP.



Ce schéma résume ce qui a été dit plus haut. Quand un routeur ne connait pas l'adresse MAC liée à une IP, il *broadcast* un paquet ARP avec l'IP de destination. Vu que ce paquet contient les adresses MAC et IP source, les autres ordinateurs apprennent la correspondance (IP source)-(MAC source). L'ordinateur de destination répond avec un paquet ARP en *unicast*, à destination du routeur émetteur, qui dit : "mon adresse MAC est .....".

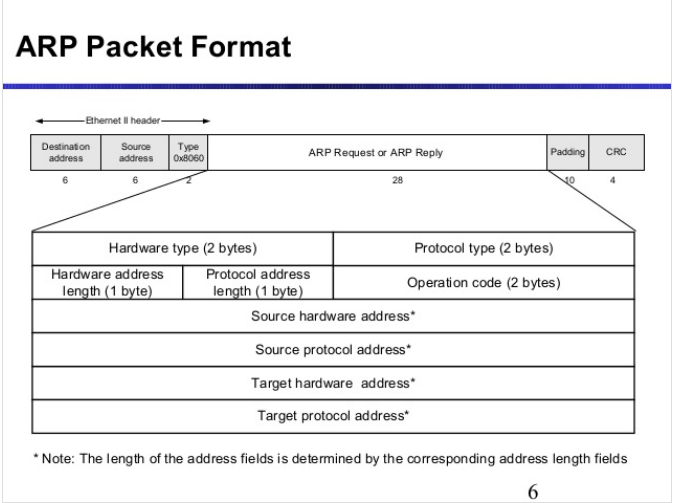
Il est possible qu'un ordinateur envoie une requête ARP à destination de lui-même, vers sa propre adresse IP ! De telles requêtes ARP sont appelées des **requêtes ARP gratuites**. L'intérêt de telles requêtes n'est pas de découvrir leur propre adresse MAC : chaque ordinateur y a accès à travers le pilote de la carte réseau. L'utilité est tout autre.

- Premièrement, elles servent à savoir si un autre poste est configuré avec leur adresse IP. Si cela arrive, l'ordinateur émetteur reçoit une réponse et sait que son IP doit être changée. Une telle situation n'est pas rare quand l'ordinateur émetteur s'attribue une IP ou lors d'une configuration quelconque.
- Une autre utilisation des requêtes ARP gratuites est suite à un changement de carte réseau. La requête gratuite permet de mettre à jour les caches ARP des autres machines automatiquement.

Les segments ARP

Tout paquet ARP est structuré comme indiqué dans le schéma ci-dessous. Il contient plusieurs champs, qui font entre un et huit octets.

- Le premier champ, nommé **Hardware Type**, indique quel est le protocole de couche liaison utilisé. Sa valeur la plus utilisée est 01, qui indique l'usage du protocole Ethernet.
- Les deux octets suivants forment le champ **Protocol Type**, qui indique le protocole de couche 3 utilisé.
- Le champ **Hardware Adress Length** donne le nombre d'octets des adresses physiques (6 pour les adresses MAC).
- Le champ **Protocol Adress Lenght** précise la longueur en octets des adresses logiques (ici, des adresses IP).
- Le champ **Opcode** est un nombre qui code l'opération à effectuer : 01 pour une requête ARP, 02 pour une réponse.
- À la suite, on trouve les adresses physiques et logiques source et destination.



Segment ARP.

Toute requête ARP est encapsulée à l'intérieur d'une trame réseau, au même titre que les paquets IP.

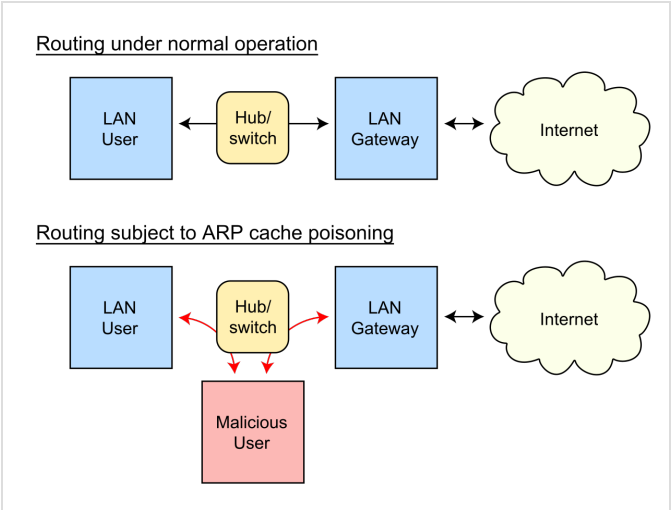
Le temps de péremption du cache ARP

Le protocole ARP permet de prendre en compte assez rapidement l'ajout d'un ordinateur dans un réseau local, en permettant de découvrir son adresse MAC. Mais il faut aussi gérer le cas où un ordinateur voit son adresse MAC changer, suite à un changement de carte réseau ou une reconfiguration quelconque. Quand cela arrive, l'ancienne association IP-MAC n'est plus valide : l'IP est restée la même, mais pas l'adresse MAC. Le cache ARP doit alors être mis à jour.

Pour gérer cette situation, le cache ARP est régulièrement purgé des associations IP-MAC trop anciennes. Dans les implémentations les plus simples, l'ensemble du cache ARP est vidé de son contenu régulièrement. Toutes les 20 minutes, le cache est remis à 0 et le protocole redécouvre les correspondances IP-MAC. Si un changement a eu lieu dans le réseau, il est pris en compte après la purge/reconstitution du cache ARP.

L'*ARP Spoofing*

Le protocole ARP a quelques failles de sécurité. Notamment, strictement rien n'est fait pour s'assurer de la validité des correspondances IP - adresse MAC. Ainsi, une machine peut se faire passer pour une autre : il lui suffit de modifier une correspondance dans la table ARP en envoyant un paquet ARP au bon moment. Prenons le cas où une IP x . x . x . x est attribuée à une machine d'un réseau local. Imaginons qu'une machine du réseau local envoie un paquet ARP de son cru, dans lequel elle prétend que l'IP x . x . x . x lui appartient. Vu qu'aucun mécanisme n'est prévu pour vérifier la validité de cette correspondance, ARP n'y verra que du feu. Ainsi, toute émission vers l'IP x . x . x . x sera redirigée vers la mauvaise adresse MAC, vers la machine qui se fait passer pour l'IP x . x . x . x.Cette machine aura accès à tout ce qui est envoyé vers l'IP x . x . x . x. Si la machine en question est configurée de manière à renvoyer les paquets vers son destinataire habituel, l'attaque peut passer inaperçu durant un moment.



ARP Spoofing.

L'obtention d'une IPv4 : le protocole DHCP

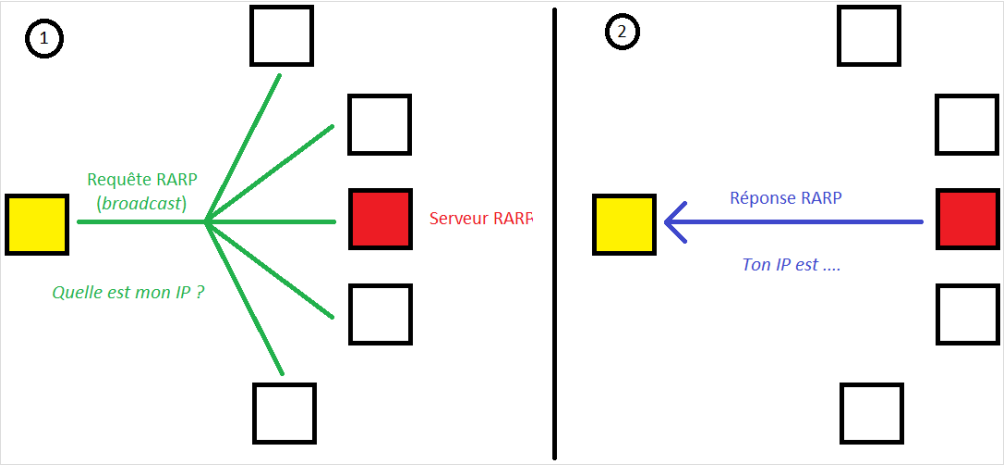
Dans le chapitre précédent, nous avons vu des protocoles qui permettent de trouver l'adresse MAC qui correspond à une IP. Mais il existe des protocoles qui font la traduction inverse, à savoir qu'ils permettent de retrouver l'IP associée à une adresse MAC. Mieux, certains permettent d'attribuer une adresse IP à un ordinateur, ce qui permet de configurer les IP d'un réseau local. On peut, par exemple, citer le protocole RARP. Mais il a rapidement montré ses limites et a été remplacé respectivement par BOOTP, puis par des processus d'assignation d'IP plus modernes (comme le DHCP du chapitre précédent). De nos jours, les protocoles du type ARP/BOOTP sont très peu utilisés, pour diverses raisons : ils nécessitent un serveur pour fonctionner, ne sont pas forcément très utiles, etc. Par contre, DHCP est clairement le plus utilisé pour attribuer des adresses IP dans un réseau (local ou non), ce qui fait que nous allons en parler plus abondamment.

Le protocole RARP

Le protocole RARP (*Reverse ARP*) fonctionne d'une manière similaire au protocole ARP, mais en sens inverse. On l'utilisait autrefois sur les terminaux sans disque dur, qui ne pouvaient pas mémoriser leur propre adresse IP, et qui ne connaissent donc que leur adresse MAC. De manière générale, il est utilisé quand des nœuds ne connaissent pas leur adresse IP, mais connaissent leur adresse MAC.

Le déroulement d'une transaction RARP

Le protocole nécessite un **serveur RARP**, qui mémorise l'adresse IP pour chaque adresse MAC (un équivalent du cache ARP, mais inversé). Quand un ordinateur/terminal veut connaitre son IP, il envoie une requête RARP de type *broadcast* sur le réseau local. Cette requête fournit l'adresse MAC de l'émetteur et demande aux autres ordinateurs s'ils connaissent l'IP du demandeur. Le serveur, ou tout autre ordinateur qui a la réponse, envoie une réponse RARP à au demandeur, qui contient l'adresse IP demandée.



Reverse ARP

L'en-tête RARP

Les paquets RARP ont un en-tête similaire à celui d'ARP, avec cependant quelques différences. L'en-tête est composé des champs suivants :

En-tête RARP

Champ	Description	Taille (en octets)
Network type	Indique le format du paquet RARP.	Deux octets
Protocol Type	Indique le protocole de couche réseau utilisé.	Deux octets
Hardware Adress Length	Nombre d'octets des adresses physiques (6 pour les adresses MAC).	Un octet
Protocol Adress Lenght	Précise la longueur en octets des adresses logiques (ici, des adresses IP).	Un octet
Opcode	Nombre qui code l'opération à effectuer.	Deux octets
Adresse physique de l'émetteur		6 octets pour une adresse MAC
Adresse logique de l'émetteur		4 à 16 octets pour une adresse IP
Adresse physique du récepteur		6 octets pour une adresse MAC
Adresse logique du récepteur		4 à 16 octets pour une adresse IP

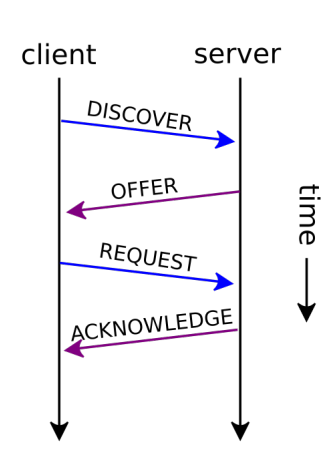
Un paquet ARP fait donc 28 octets, ce qui est plus petit que les 46 octets minimum d'une transmission Ethernet. Les 18 octets manquants sont comblés par des bits de bourrage, comme pour toute trame trop petite. Alors qu'on aurait pu utiliser ces 18 octets restants d'une manière plus utile...

Le protocole DHCP

On a vu que toute machine, ainsi que tout réseau, se voit attribuer une adresse IP, essentielle pour l'accès Internet. On peut cependant se demander qui attribue les adresses IP à une machine. La réponse est qu'il y a deux possibilités. La première est celle d'une intervention humaine : l'administrateur du réseau attribue lui-même les IP aux machines, à sa charge de bien faire attention à ce qu'un autre réseau local n'aie pas la même IP. Une autre solution ne nécessite pas d'intervention humaine, tout étant automatisé. L'IP est alors fournie par une source extérieure. Cette dernière solution demande cependant qu'un protocole se charge de l'attribution des IP. De nombreux protocoles de ce genre ont existé : BOOTP et RARP sont de loin les plus vieux. De nos jours, c'est le **protocole DHCP** (Dynamic Host Configuration Protocol) qui est utilisé pour les adresses IPv4. Et c'est celui que nous allons étudier.

Fonctionnement

Ce protocole permet à des ordinateurs qui veulent une IP d'acquérir celle-ci auprès de serveurs DHCP. Les clients connaissent les IP des serveurs DHCP, dont l'IP est fixé par le protocole DHCP. Les serveurs DHCP ont chacun une liste d'IP non-attribuées, qu'ils peuvent distribuer aux ordinateurs qui en font la demande. Pour lus de flexibilité, une IP est attribuée à un client/réseau durant un temps limité. Tout client qui veut une IP va envoyer une demande à plusieurs serveurs DHCP, en indiquant le temps durant lequel il veut réserver l'adresse IP. Les serveurs répondent en envoyant au client une offre, une proposition d'IP que les clients peuvent refuser ou accepter. Le client choisit une offre et renvoie un accusé de réception au serveur émetteur de l'offre. Ce serveur renvoie alors lui aussi un accusé de réception.



Obtention d'une adresse IP par le protocole DHCP : illustration des échanges entre client et serveur.

Format des messages DHCP

Toout message DHCP suit le format suivant :

Octet 0	Octet 1	Octet 2	Octet 3
Champ OP : vaut 1 pour une requête, 2 pour une réponse.	**HTYPE** : indique le type de réseau.	**HLEN** : longueur des adresses physiques.	**HOPS** : compteur de sauts (similaire au TTL).
XID : numéro de transaction. Généré aléatoirement par le client, répondu à l'identique par le serveur.			
SECS : Temps écoulé depuis le démarrage du client DHCP sur la machine.		**FLAGS** : ensemble de valeurs de 1 bit, aux utilités diverses.	
CIADDR : Adresse IP du client, si déjà connue du client (en cas de renouvellement d'IP, par exemple). Mis à 0 si inconnue, en cas de demande.			
YIADDR : Adresse IP du client, donnée par le serveur lors de sa réponse. Vaut 0 en cas de demande.			
SIADDR : Adresse IP du serveur.			
GIADDR : Adresse IP Gateways.			
CHADDR : Adresse physique du client (adresse MAC), de 64 octets.			
SNAME : Nom d'hôte du serveur (64 octets). Parfois remplacé par des bits qui précisent certaines options du protocole.			
FILE : Nom du fichier de démarrage (128 octets). Parfois remplacé par des bits qui précisent certaines options du protocole.			
Options DHCP.			

La gestion des erreurs : le protocole ICMP

Une transmission IPv4 peut ne pas aboutir, pour des raisons diverses : problème de formatage du paquet, machine de destination déconnectée, erreur du niveau du routeur, durée du datagramme expirée, etc. Quand un routeur détecte une telle erreur, le paquet qui a subi l'erreur est détruit par le routeur en question. De plus, le routeur doit prévenir la machine émettrice que le paquet n'a pas été transmis. Pour cela, le routeur émet un paquet ICMP, formaté suivant le **protocole ICMP**. Ce protocole sert à transmettre des messages d'erreur entre machines ou routeurs. ICMP est un protocole de couche 3 (couche Internet), tout comme IP.

Le format des paquets ICMP

Les messages ICMP sont encapsulés dans des paquets IP et sont routés comme tout autre paquet IP. Tout paquet ICMP commence donc par un en-tête IP, suivi par le contenu du paquet ICMP. Dans le schéma ci-dessous, l'en-tête IP est en violet, alors que le contenu du paquet ICMP est en rose. On voit que l'en-tête ICMP fait 8 octets, soit 64 bits.

Bit 0 - 7	Bit 8 - 15	Bit 16 - 23	Bit 24 - 31
Version/IHL	Type de service	Longueur totale	
Identification (fragmentation)		*flags* et *offset* (fragmentation)	
Durée de vie(TTL)	Protocole	Somme de contrôle de l'en-tête	
Adresse IP source			
Adresse IP destination			
Type de message	Code	Somme de contrôle	
Bourrage ou données			
Données (*optionnel et de longueur variable*)			

Paquet ICMP

On voit que le paquet ICMP contient quatre champs : TYPE, CODE, CODE DE CONTRÔLE et DONNÉES.

- Le champ TYPE est un octet qui indique quel est le type de message ICMP envoyé.
- Le champ CODE est un octet qui code des informations diverses.
- Le CODE DE CONTRÔLE est un mot de 16 bits qui permet de détecter et corriger les erreurs de transmission.
- Le champ DONNÉES correspond aux données du paquet ICMP, parfois précédées de bits de bourrage (pour que l'en-tête ICMP ait une taille fixe).

La signification des paquets ICMP

Le message ICMP utilise ses champs TYPE et CODE pour dire quelle est l'origine de l'erreur, à savoir est-ce que la machine destination est inaccessible, est-ce que le protocole utilisé est mauvais, etc. On peut classer les messages selon la valeur du champ TYPE.

Les paquets ECHO

Un TYPE valant 0 ou 8 signifie que le message n'est pas un message d'erreur, mais une demande ou résultat de ping. Pour rappel, le ping est une commande qui permet de savoir quel le temps mis par un paquet pour atteindre sa destination. Plus le ping est bas, plus le temps de transmission est faible. Le ping donne donc une idée des performances du réseau, et notamment de ce qu'on appelle sa latence. Ce ping a une grande importance dans la plupart des applications réseau, et surtout dans le cas des jeux vidéos multijoueurs. Le principe du Ping est d'envoyer un paquet requête vers la machine destination, qui doit répondre par un paquet réponse. Le temps de transmission est simplement le temps d'aller-retour entre l'émission du paquet requête et la réception du paquet réponse. Ces paquets sont appelés des paquets ECHO et leur champ CODE est toujours à 0. Le paquet requête est un paquet ICMP dont le TYPE vaut 8, le paquet réponse a quant à lui un champ TYPE à 0.

Les autres valeurs du champ TYPE

Le TYPE est mis à 3 si jamais la machine de destination est inaccessible ou qu'un routeur n'a pas réussit à router le paquet.

Le TYPE est mis à 4 si le volume de données transmis est trop important. Cela sert à signaler des erreurs de fragmentation de paquet.

Le TYPE est mis à 5 si le chemin pris par le paquet n'est pas le bon. Dit autrement, il indique une erreur de routage.

Un TYPE à 11 signifie que le paquet a dépassé son TTL.

Le TYPE est mis à 12 si l'en-tête du paquet IP est erroné ou corrompu.

Un TYPE à 13 ou 14 est utilisé pour les paquets de synchronisation entre routeurs.

Les autres TYPE servent pour des échanges d'informations entre routeurs. Les TYPE 15 et 16 sont aujourd'hui obsolètes. Ils servaient au démarrage d'une machine, pour lui attribuer une adresse IP, chose devenue inutile avec les protocoles DHCP et RARP. Les TYPE 7 et 18 servent à obtenir le masque de sous-réseau vers le routeur qui gère la machine de destination. On peut ainsi envoyer une adresse IP, dans un paquet de requête de masque de sous-réseau, de TYPE 17. Celui-ci répond alors par un paquet de TYPE 18, qui transmet le masque de sous-réseau associé.

La couche transport : UDP et TCP

Les protocoles de la couche transport les plus connus sont les protocoles TCP et UDP. Le protocole UDP est le plus simple des deux, alors que TCP est celui qui a le plus de fonctionnalités.

Les fonctions de la couche transport

Les fonctions de la couche transport sont multiples, mais les fonctions principales sont au nombre de deux :

- Découper des données, de taille variable, en paquets de taille fixe.
- Identifier les programmes destinataire/émetteur de la donnée.

La segmentation

D'ordinaire, les ordinateurs s'échangent des données de taille variable et non-bornée : des fichiers, des pages web, des images, etc. Cependant, on a vu que le matériel réseau ne gère que des paquets de données, qui ont une taille maximale. Pour résoudre cette incompatibilité apparente, on est obligé de les découper en paquets de taille fixe, qui ne peuvent pas dépasser une taille maximale (le MTU). Dans le jargon du réseau, ces paquets de taille fixe de couche transport sont appelées des **datagrammes** ou des **segments**.

Lors de l'envoi d'une donnée sur le réseau, l'équipement réseau doit segmenter les datagrammes en paquets et les envoyer sur le réseau individuellement. Un problème avec cette solution est que les segments/datagrammes sont envoyés séparément et qu'ils peuvent arriver dans le désordre. UDP et TCP gèrent ce problème différemment. UDP ignore totalement l'ordre des datagrammes et ne cherche pas à les remettre dans l'ordre à la réception. Cela marche bien pour certaines données, comme la transmission de la téléphonie par IP, ou un flux vidéo. De son côté, TCP dispose de mécanismes pour remettre les segments dans l'ordre d'envoi et reconstituer fidèlement la donnée transmise.

L'identification des processus émetteur/récepteur

Quand un ordinateur reçoit un paquet, il doit savoir à quel programme est destiné ce paquet : est-il destiné au navigateur web, à un jeu vidéo, ou au service de mise à jour de l'OS ? Pour cela, on définit ce qu'on appelle des **ports logiciels** : ce sont de simples numéros, que chaque application va réserver en émettant des données. Pour comprendre ce qu'est un port logiciel, on peut faire une analogie avec le courrier. Quand quelqu'un envoie une lettre, il ne précise pas seulement l'adresse postale, mais aussi la personne à laquelle elle est destinée, au cas où plusieurs personnes vivent à la même adresse. Dans le domaine du réseau, la lettre est un paquet réseau, le destinataire et l'émetteur de la lettre sont des programmes/processus, l'adresse postale est équivalente à l'adresse IP et le nom du destinataire est le port logiciel.

Les numéros de ports actuels font deux octets (16 bits), ce qui donne 65536 ports différents. L'IANA, l'organisme qui gère les noms de domaine, classe ces ports en trois types, illustrés dans le tableau ci-dessous.

Ports dédiés	0 à 1023	Ports réservés à des fonctions bien précises. ▪ 21: File Transfer Protocol (FTP) ▪ 22: Secure Shell (SSH) ▪ 25: Simple Mail Transfer Protocol (SMTP) ▪ 53: Domain Name System (DNS) service ▪ 80: Hypertext Transfer Protocol (HTTP) ▪ 110: Post Office Protocol (POP3) ▪ 143: Internet Message Access Protocol (IMAP) ▪ 443: HTTP Secure (HTTPS)
Ports réservés	1024 à 49151	Ports réservés à des applications propriétaires.
Ports dynamiques	49152 à 65535	Ports libres, non-réservés, utilisables à la demande.

Les pare-feu permettent d'interdire la communication sur certains ports et de filtrer les segments/datagrammes selon le numéro de port. Ils laissent passer les segments/datagrammes sur les ports logiciels autorisés, mais ils détruisent les segments/datagrammes envoyés/reçus sur les ports interdits. Cela permet d'éviter la réception ou l'envoi de segments/datagrammes dangereux, ou du moins non-souhaités. Par exemple, on peut configurer un pare-feu pour n'autoriser que les ports 80, 25, 110, 143 et 443 : seule la consultation de sites web et de mail sera possible. En général, les ports dynamiques font partie de ceux filtrés en priorité.

Le protocole UDP

UDP se contente du minimum syndical qu'on attend d'un protocole de couche transport : il gère les ports logiciels, mais ne vérifie pas que la donnée est bien arrivée à bon port, pas plus qu'il ne remet les datagrammes dans leur ordre d'envoi. Il est donc très adapté dans les situations où on se moque que les données arrivent dans l'ordre et où les pertes de données sont acceptables. Typiquement, un flux vidéo, un podcast, des jeux vidéos en ligne sont des utilisations les plus courantes de UDP.

UDP se contente d'ajouter un en-tête particulièrement simple aux datagrammes. Cet en-tête contient diverses informations, toutes codées sur deux octets. Il a une taille fixe, de 64 bits, soit 8 octets, quel que soit le paquet. On y trouve naturellement le port de l'application émettrice ainsi que le port de destination, deux informations essentielles pour tout protocole de la couche transport. L'en-tête contient aussi la longueur totale du paquet, en-tête compris, et des octets de contrôle d'erreur optionnels.

Port Source (16 bits)	Port Destination (16 bits)
Longueur (16 bits)	Somme de contrôle (16 bits)
Données (longueur variable)	

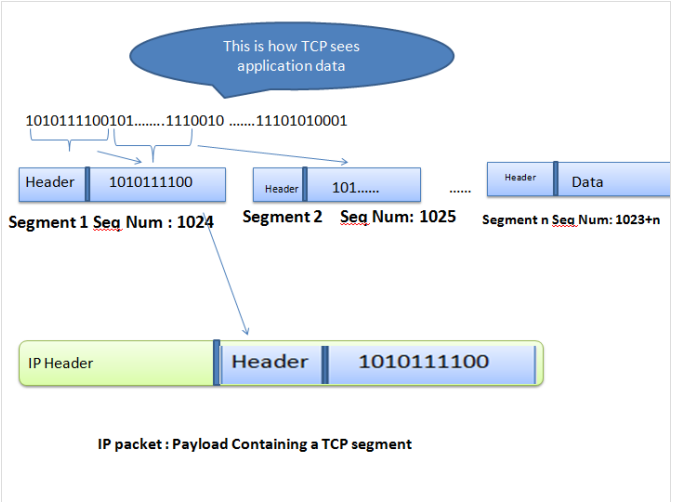
La seule subtilité de ce protocole tient dans le calcul de la somme de contrôle, que nous n'aborderons pas ici. Nous allons simplement dire que ce calcul se base sur l'en-tête IP, chose qui est en contradiction avec l'indépendance des couches ! Heureusement, UDP et IP sont pris en charge par le système d'exploitation, ce qui réduit quelque peu les problèmes engendrés par cette méthode de calcul.

Le protocole TCP

Le protocole TCP peut être vu comme un UDP sous stéroïdes. Non seulement il utilise des ports logiciels, mais il ajoute aussi des fonctionnalités qu'UDP n'a pas. Parmi ces fonctionnalités, on trouve la gestion des connexions, les accusés de réception et la détection des pertes de données et la gestion de l'ordre de réception. Ces fonctionnalités sont indispensables pour de nombreuses applications, comme les navigateurs web, le transfert de mails, et bien d'autres. Il n'est donc pas étonnant que TCP soit le protocole de couche transport le plus utilisé, loin devant UDP.

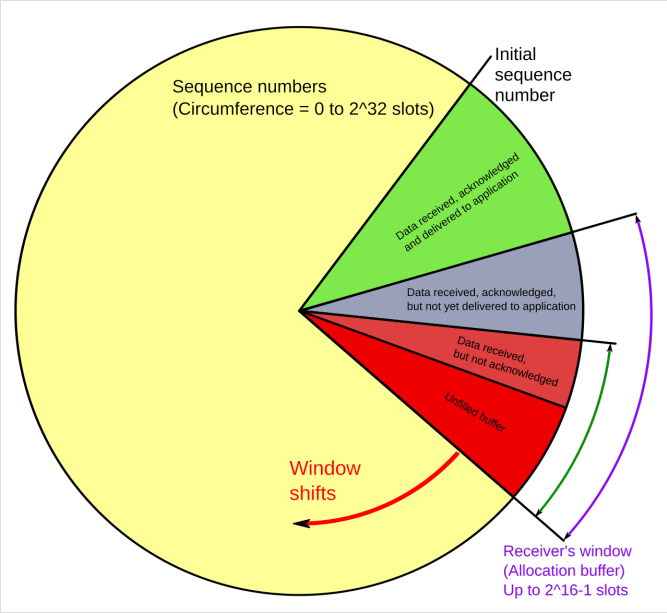
Le séquencement des paquets

Comme dit plus haut, TCP doit remettre les segments dans l'ordre pour reconstituer la donnée transmise. Le moyen le plus simple pour cela est de réutiliser la technique de la fenêtre glissante vue il y a quelques chapitres. Pour rappel, cette méthode demande que les paquets soient numérotés selon leur ordre d'envoi. Si un datagramme est découpé en N paquets, on peut simplement numéroter chaque paquet suivant son ordre dans la donnée initiale : le premier paquet sera le paquet numéro 1, le second paquet le numéro 2, etc. Ce numéro est transmis avec le paquet en question, à côté des numéros de ports logiciels.

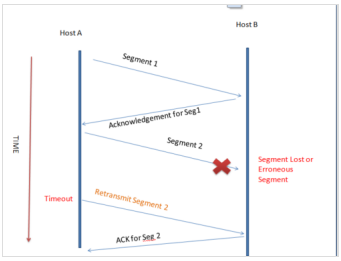


TCP segment.

Lors de la réception, l'ordinateur doit regrouper plusieurs paquets en un seul datagramme. Pour cela, il va accumuler les paquets reçus dans une portion de mémoire, la **fenêtre TCP**. Cette fenêtre permet d'utiliser des mécanismes pour détecter ou empêcher les pertes de données. Seulement, la fenêtre a une taille limitée, qui est comprise entre 2 et 65536 octets. Sans précaution particulière, il est possible que la fenêtre TCP devienne pleine, notamment lors de transferts de datagrammes imposants. Pour limiter la catastrophe, le récepteur peut émettre un paquet qui indique à l'émetteur la place disponible dans sa fenêtre. Pour cela, les paquets TCP contiennent un champ nommé *Window*, qui indique combien d'octets peuvent encore être envoyés avant que la fenêtre soit totalement remplie (autrement dit, la place libre en octets dans la fenêtre).



Fenêtre TCP.



ACK TCP

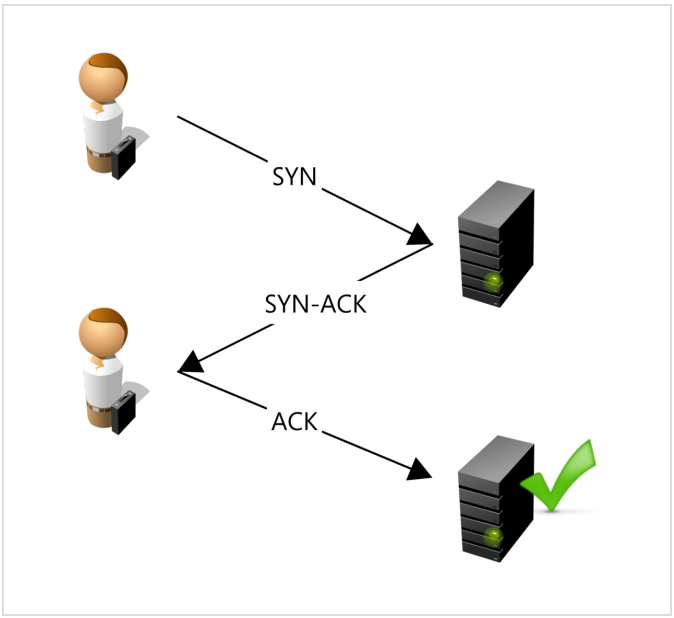
Les accusés de réception permettent de savoir si un paquet envoyé a bien été reçu. Lorsque le serveur reçoit un paquet, il envoie à l'émetteur un paquet ACK qui indique qu'il a bien reçu le paquet en question. Ces accusés de réception permettent de savoir si une donnée a été perdue lors de son transfert, que celle-ci n'est pas arrivée à destination à temps et a disparu. De telles pertes de données arrivent assez souvent, pour des raisons diverses. Chaque accusé de réception contient un numéro de séquence, qui indique que tous les paquets situés avant ce numéro de séquence ont été reçus : il permet donc d'accuser la réception de plusieurs paquets en même temps. Précisément, la fenêtre TCP commence au premier paquet non-acquitté, et contient tous les paquets suivants (acquittés ou non). La fenêtre TCP contient donc des paquets acquittés, des paquets non acquittés, et de l'espace vide.

Avec les accusés de réception, on sait qu'une donnée a disparu si on n'a pas reçu d'accusé de réception après un certain temps. Si une donnée est déclarée comme perdue, il suffit de la renvoyer en espérant que cette fois sera la bonne. Reste que la durée avant qu'on considère qu'un paquet est perdu varie suivant les circonstances. Tout dépend en réalité du serveur, de sa distance, du temps mis à transférer les données, et d'autres paramètres. Une donnée trop courte entraînera beaucoup de renvois de données inutiles, alors qu'une donnée trop longue fera attendre le client inutilement. Déterminer la durée idéale se fait par divers algorithmes logiciels, intégré dans le système d'exploitation.

Les connexions TCP

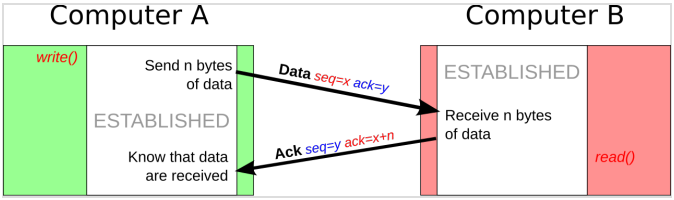
TCP est un protocole qui est dit en mode connecté. Ce qui signifie que tout transfert de donnée doit être précédé d'une négociation entre l'émetteur et le récepteur, cette négociation étant appelée une **connexion**. La connexion doit être ouverte pour que l'échange de donnée ait lieu et elle doit être fermée pour que l'échange de donnée cesse. La connexion s'effectue en trois étapes :

- le client initie la connexion au serveur en envoyant un paquet spécial (SYN) ;
- le serveur répond qu'il autorise la connexion avec un autre paquet spécial (SYN-ACK) ;
- le client envoie un accusé de réception au serveur.



Connexion TCP

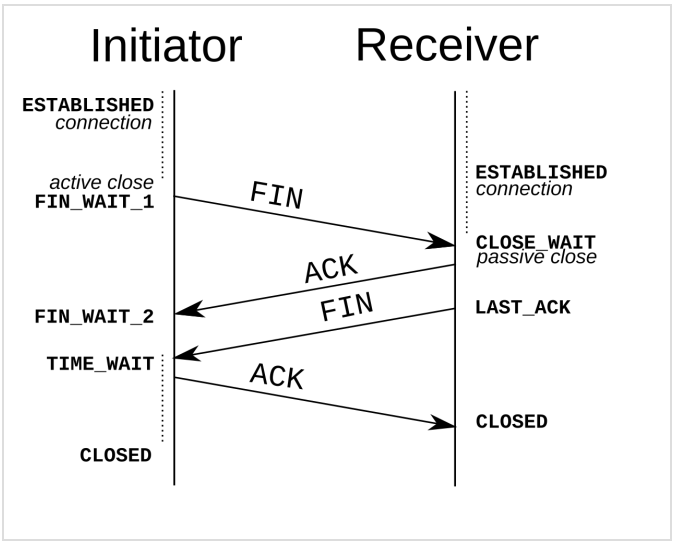
Lors de la phase d'envoi de données, l'émetteur envoie des paquets de type DATA, et le serveur répond par des accusés de réception.



Envoi de données via TCP.

La déconnexion s'effectue en quatre étapes. Dans les grandes lignes, le serveur et le client doivent se déconnecter chacun de leur côté. Chaque demande de déconnexion d'un ordinateur est autorisée par un accusé de réception. Pour se déconnecter, ils doivent envoyer un paquet spécial nommé FIN. Dans les grandes lignes, voici comment a lieu la déconnexion :

- un ordinateur envoie une demande de déconnexion ;
- l'autre ordinateur reçoit celle-ci et renvoie l'ACK qui va avec ;
- ce dernier envoie lui aussi une demande de déconnexion ;
- et son comparse reçoit celle-ci et renvoie l'ACK qui va avec.



Déconnexion TCP

Les segments TCP

Un segment TCP est composé d'un en-tête TCP suivi par le paquet de données de la couche application/session/présentation. Il a une taille variable, en raison de la présence d'un champ *Options* de taille variable à sa toute fin. Sa taille est systématiquement un multiple de 32 bits. Sa taille minimum est de 20 octets (absence de champ *Options*), et sa taille maximum est de 60 octets (champ *Options* le plus rempli possible).

Tout segment TCP commence naturellement par les **ports source et destination**.

Les deux champs suivants servent pour la gestion de la fenêtre glissante.

- Le **numéro de séquence** dit quel le numéro du segment envoyé.
- Le **numéro d'accusé de réception** sert dans les réponses ACK du récepteur : il indique quel est le numéro du segment accusé.

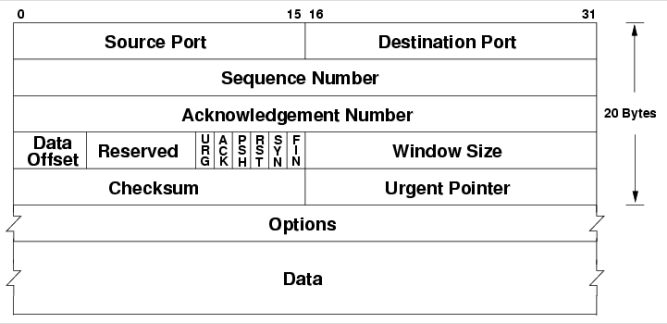
Le champ suivant donne la taille de l'en-tête, exprimée en mots de 32 bits. Il est nécessaire car l'en-tête TCP a une taille variable. Ce champ a une taille de 4 bits, de qui fait 16 valeurs différentes. allant de 0 à 15. Sachant que la taille de l'en-tête est exprimée en mots de 32 bits/4 octets, cela fait un en-tête qui fait entre 0 et 60 octets (15 * 4). Mais les valeurs inférieures à 5 sont interdites, ce qui fait un minimum de 5*4 = 20 octets. On retrouve les valeurs mentionnées au début de cette section.

Les indicateurs qui suivent sont des bits qui indiquent le type du paquet, son utilité. Ils servent notamment à dire si le paquet est un paquet SYN, un paquet ACK (accusé de réception), etc. Il est composé des 8 bits suivants :

- CWR : bit en lien avec la gestion de la congestion réseau ;
- ECE ('*Explicit Congestion Notification*) : idem ;
- URG : indique que le segment est de type urgent ;
- ACK : indique que le segment est un accusé de réception ;
- PSH : lié à la fonction PUSH ;
- RST : réinitialise la connexion ;
- SYN : indique que le segment est de type SYN (demande de connexion) ;
- FIN : fermeture de connexion, l'émetteur n'a plus rien à envoyer.

Le champ *window size* indique la taille de la fenêtre glissante, à savoir le nombre de segments pouvant être reçus sans accusés de réception.

Le reste des champs est une somme de contrôle, un champ qui indique que certaines données sont urgentes à traiter, et quelques bits de bourrage ou d'options.



TCP Header

La couche présentation

Après avoir vu les couches transport et session, il est temps de passer à la couche suivante : la **couche présentation**. Le rôle de cette couche est de gérer l'encodage des données envoyées et/ou reçues. Par encodage, on veut dire comment les données sont transformées en un paquet réseau, un flux de bits. Le problème que cherche à résoudre cette couche est comment faut-il découper et interpréter un paquet réseau, la suite de bits envoyée/reçue. Après tout, rien ne ressemble plus à une suite de bits qu'une autre suite de bits : celle-ci peut être aussi bien une image au format JPEG, un tableau de nombres entiers, un morceau de fichier son, etc. L'interprétation à donner à une suite de bits n'est pas possible sans informations supplémentaires : tout dépend de la manière dont l'émetteur l'a organisée, manière qui doit être connue du récepteur. Sans couche présentation, le récepteur ne saurait pas quel est le type de la donnée envoyée, ce qu'elle représente, comment l'interpréter. Pour résumer, la couche présentation prend en charge le codage des données au sens large, ce qui englobe aussi bien la compatibilité des données entre ordinateurs que leur compression et leur cryptage. Dans ce qui va suivre, nous verrons les protocoles les plus utilisés pour coder les données, ainsi que ceux utilisés pour compresser ou crypter des paquets réseau.

L'encodage des données

Le premier rôle de la couche présentation est de garantir la **compatibilité des données** entre les machines d'un réseau. Pour rappel, les données peuvent être codées de pleins de manières différentes : le texte peut utiliser des jeux de caractères très différents, les entiers peuvent être codés en petit ou grand boutisme, les formats de nombres flottants peuvent différer selon l'ordinateur, et ainsi de suite. Lorsque deux ordinateurs communiquent entre eux, il se peut qu'ils utilisent des formats de données différents. Dans ce cas, les données doivent être converties d'un format à un autre avant la transmission.

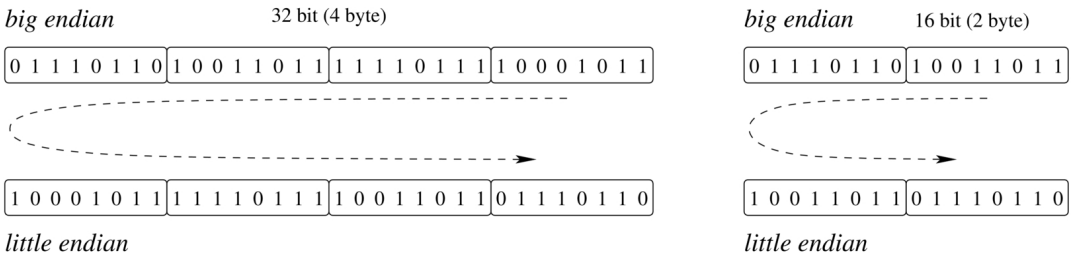
Une solution demande aux deux ordinateurs de s'échanger des informations sur les formats que chacun utilise : l'émetteur et le récepteur vont ainsi se mettre d'accord sur les formats supportés et choisir un format commun. Il s'agit d'une **phase de négociation** entre les deux machines, comme dans le protocole HTTP où le client indique les caractéristiques qu'il accepte (types de fichiers, encodage de transfert, ...) avec coefficient de préférence et où le serveur essayes de satisfaire le client en respectant ses souhaits. Une telle négociation existe dans beaucoup de protocoles tels le protocole SSL (Secure Socket Layer) permettant de négocier les algorithmes de cryptage employés et d'autres paramètres.

Pour se simplifier la tâche, beaucoup de protocoles réseau utilisent souvent d'autres méthodes. Généralement, on utilise des formats intermédiaires standardisés : les données à envoyer sont converties dans ce standard intermédiaire, sont transmises et le récepteur traduit ces données intermédiaires dans ses formats à lui. Cette méthode simplifie grandement les conversions et les échanges entre ordinateurs. Une autre solution est d'envoyer les données sans conversion au récepteur en précisant le format utilisé, le récepteur prenant en charge la conversion. Il s'agit de deux méthodes différentes : soit le récepteur fait la conversion, soit on utilise un format intermédiaire. La seconde méthode est de loin la plus utilisée aux heures actuelles.

Les conversions d'entiers

Le premier type de conversion inter-machines implique les nombres, surtout les nombres entiers. Selon les ordinateurs, ceux-ci peuvent être représentés différemment : en complément à deux, en signe-magnitude, sans signe, etc. De plus, la taille de ces entiers n'est pas la même selon les ordinateurs. Par exemple, les processeurs x86 récents utilisent des entiers de 64 bits, alors que les anciens effectuent des calculs sur des nombres de 32 bits. Mais une bonne partie de ces conversions de taille sont prises en charge par le matériel : le processeur peut souvent traduire à la volée des nombres selon leur taille. Cependant, il y a une manipulation qui ne peut pas être prise en charge par le matériel et qu'il faut donc prendre en compte lors des transferts réseaux : la conversion du boutisme. Pour la comprendre, nous allons devoir faire un rappel sur ce qu'est le boutisme.

On peut introduire cette notion par une analogie avec les langues humaines : certaines s'écrivent de gauche à droite et d'autres de droite à gauche. Dans un ordinateur, c'est pareil avec les octets des mots mémoire : on peut les écrire soit de gauche à droite, soit de droite à gauche. Quand on veut parler de cet ordre d'écriture, on parle de **boutisme** (endianness). Sur les processeurs gros-boutistes, l'octet de poids fort de l'entier est stocké dans l'adresse la plus faible (et inversement pour le poids faible). Sur les processeurs petit-boutistes, c'est l'inverse : l'octet de poids faible de notre donnée est stocké dans la case mémoire ayant l'adresse la plus faible. Si deux ordinateurs avec des boutismes différents échangent des données, le résultat envoyé ne sera pas interprété correctement : les octets des entiers seront lus dans l'ordre inverse. En conséquence, il faut faire la conversion entre petit et grand boutisme.

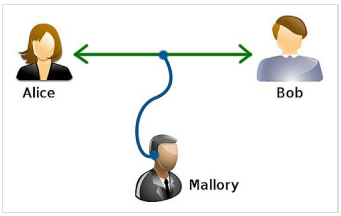


Le chiffrement des données

On a vu que les données envoyées sur internet transitent par divers intermédiaires. En temps normal, ces intermédiaires ne font que transmettre les données sans les analyser en profondeur. Mais il arrive que, suite à certaines failles de sécurités, qu'un individu malveillant ait accès aux données que vous transférez sur le net. Certains pirates utilisent le fait que les données sont transmises en clair, sans chiffrement, pour les intercepter et les utiliser à leur profit. Par exemple, un pirate pourrait capter les identifiants que vous utilisez pour faire des paiements sur internet ou vous connecter à tel ou tel compte. Sans chiffrement, les données sont envoyées en clair sur internet et un attaquant peut parfaitement les intercepter lors de sa transmission entre le serveur et l'utilisateur. Une telle attaque d'espionnage est appelée une **attaque de l'homme du milieu**.

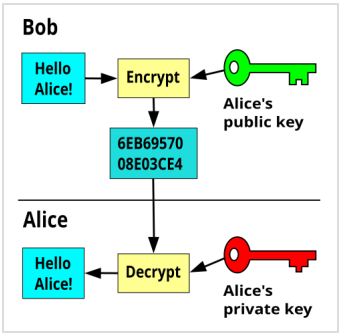
Pour éviter ce genre d'attaque, les informations échangées doivent être cryptées avant leur transfert. Pour rappel, le cryptage consiste à transformer les données à transmettre grâce à une fonction mathématique et/ou un algorithme. Cette transformation se fait en manipulant les données et une ou plusieurs clés, une information connue seulement de l'émetteur et du destinataire. Une clé permet de crypter le message, à savoir le rendre incompréhensible, et une

autre permet de décrypter le message, à savoir retrouver la donnée initiale. Les données sont totalement incompréhensibles pour qui ne connait pas la clé de décryptage, ce qui fait que seuls l'émetteur et le destinataire peuvent lire le message. Les méthodes les plus simples n'utilisent qu'une seule clé, qui sert à la fois pour crypter les données et les décrypter. On parle de chiffrement symétrique. Mais sur le net, les techniques simples ne peuvent pas être utilisées. En effet, cela demanderait de transmettre la clé au destinataire du message, clé qui peut alors être interceptée au même titre que le message.

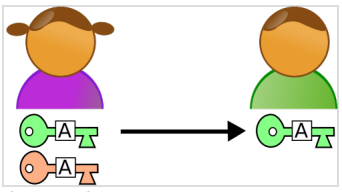


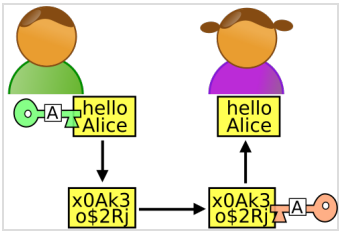
Attaque de l'homme du milieu.

Les seules méthodes qui permettent de transmettre des informations chiffrées sur le net sont des protocoles de chiffrement asymétrique, où la clé de cryptage et la clé de décryptage sont différentes. La clé de chiffrement est appelée la clé publique, alors que celle de déchiffrement est appelée clé privée. Le **chiffrement asymétrique** se base sur un principe simple pour l'échange des clés entre deux ordinateurs. L'ordinateur qui veut recevoir un message va envoyer à l'autre sa clé publique. L'autre ordinateur peut alors utiliser cette clé pour chiffrer le message à envoyer. Le message crypté est alors transmis au récepteur, qui utilise sa clé privée pour le déchiffrer. Le message réellement transmis est donc illisible pour tout attaquant, vu qu'il ne connait pas la clé privée du récepteur. La connaissance de la clé publique ne lui sert à rien.



Cryptographie à clé publique.


Étape 1 : Échange des clés publiques.



Étape 2 : Transfert de données cryptés sur internet.

L'inconvénient du chiffrement à clés asymétriques est qu'il est plus lent qu'un chiffrement à clé symétrique à cause du fait qu'il soit basé sur des problèmes mathématiques complexes. Pour pallier cette lenteur, les systèmes de chiffrement utilisent une combinaison des deux techniques : le chiffrement à clés symétriques est utilisé pour crypter les données et le chiffrement à clés asymétriques est employé uniquement pour chiffrer la clé symétrique employée qui est générée à la volée et à usage unique pour cette session.

La couche application : le web et ses protocoles

Nous arrivons à la fin de ce cours. Il ne nous reste plus qu'à voir la couche application, celle qui contient la plupart des protocoles liés aux sites web. Et pour bien introduire ce chapitre, commençons par voir quelques notions de base sur les sites web. Rappelons que la majorité des sites web fonctionnent sur le principe du client-serveur, vu dans le premier chapitre sur cours. Un *site web* n'est ni plus ni moins qu'un ensemble de fichiers stockés sur un *serveur web*. Chaque page correspond à un ou plusieurs fichiers : une page en pur texte est d'un seul tenant, alors que les pages avec des vidéos ou des images ont un fichier supplémentaire par image/vidéo. Les ordinateurs qui veulent consulter un site web vont se connecter au serveur : ce sont les *clients web*. Les *navigateurs web* sont des applications qui permettent à un client web de consulter le contenu des serveurs web, pour les consulter pages, télécharger des fichiers, ou toute autre action du même genre.

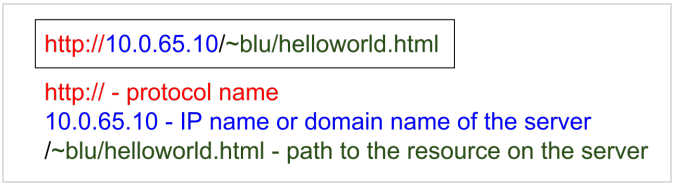
Au tout début d'internet, les pages web étaient ce qu'on appelle des **pages web statiques**. L'ensemble de la page est stocké dans un fichier, qui est interprété tel quel par un navigateur web. Ce fichier n'est cependant pas un fichier image ou un fichier texte habituel, mais est codé dans un langage de description de documents particulier : le HTML, très souvent couplé avec du CSS. Le premier définit une syntaxe particulière pour afficher différents widgets, listes, tableaux texte, et autres. Le second définit diverses options d'apparence, qui permettent de modifier l'allure et les graphismes des widgets. Dans tous les cas, les pages statiques ont un contenu immuable, déterminé lors de leur conception, incapable de s'adapter aux circonstances. Par exemple, elles ne permettent pas de créer des comptes clients, pour se connecter à un site, ou des fioritures similaires.

De nos jours, le HTML et le CSS sont complétés par d'autres langages de programmation, qui permettent d'obtenir des sites plus évolués. Ces sites ont des **pages web dynamiques**, qui peuvent s'adapter au client. Ces pages web sont calculées à la volée par le serveur, ce qui est utile dans divers scénarios. Par exemple, un site web sur lequel on peut s'inscrire ou avec un forum sera systématiquement une page web dynamique. Le langage de programmation utilisé par sur ces pages est souvent le PHP, éventuellement du Javascript. Le PHP est un langage qui s'exécute sur le serveur, ce qui permet notamment l'accès à une BDD quelconque. La page web est alors construite, calculée à la volée par le serveur. En comparaison, le Javascript est un langage qui s'exécute sur le client, dans son navigateur web. Il faut noter que le HTML et le CSS sont cependant utilisés de concert avec PHP/Javascript sur les pages dynamiques.

Les adresses web (URL)

Toute page web a une adresse appelée **adresse URL**, qui est utilisée pour y accéder ou pour créer des liens vers celui-ci. Les adresses URL ressemblent plus ou moins à ceci : *http://www.example.com*.

Prenons une adresse URL, https://www.fr.wikipedia.org/wiki/Wikip%C3%A9dia:Accueil_principal par exemple. Il faut savoir que seule une partie de l'adresse URL est utile pour identifier l'adresse IP du serveur. Le reste de l'adresse sert à indiquer quel est le fichier demandé, mais ne sert pas à spécifier le serveur. La partie de l'URL qui détermine l'IP est ce qu'on appelle le **nom de domaine**. Généralement, le nom de domaine est situé après www. et avant le symbole "/" qui suit. En reprenant l'adresse suivante, le nom de domaine est www.fr.wikipedia.org et le reste de l'adresse, /wiki/Wikip%C3%A9dia:Accueil_principal, sert à indiquer où on doit aller chercher le fichier.

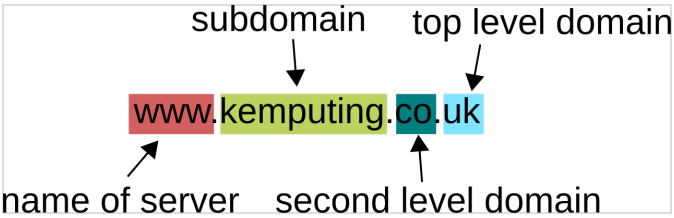


URL example

L'organisation hiérarchique des domaines

Un nom de domaine est composé de plusieurs noms, séparés par des points. Chaque nom fait référence à ce qu'on appelle un **domaine internet**, quelque chose qui regroupe plusieurs sites ou pages web qui ont un lien.

L'ensemble des domaines est organisée de manière hiérarchique, avec des domaines de 1er niveau, de 2nd niveau, de 3ème niveau, etc.



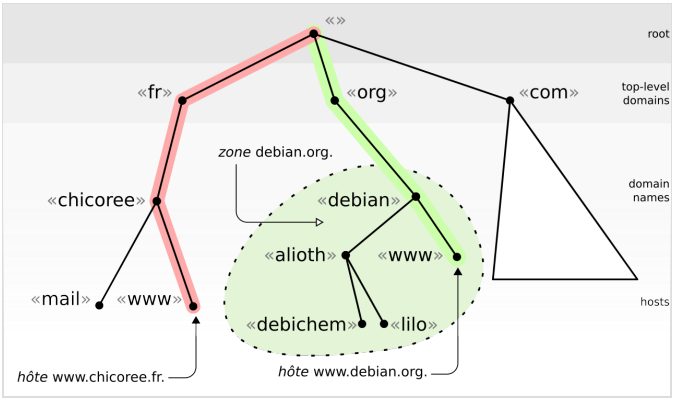
Organisation hiérarchique des domaines internet.

Les domaines principaux, appelés **domaines de premier niveau**, sont les fameux domaines .fr, .uk, .com, .org, et ainsi de suite. Ils sont gérés par l'ICANN (Internet Corporation for Assigned Names and Numbers), un organisme américain, depuis 1998. L'ICANN distingue deux types de noms de domaines : les domaines géographiques (.fr, par exemple) sont liés à un pays, tandis que les autres sont liés à des organisations (.gov pour les gouvernements, .edu pour les établissements scolaires, .mil pour les organismes militaires, et quelques autres).

En dessous de ces domaines, on trouve des **domaines de second et de troisième niveau**, qui sont subordonnées aux domaines de niveau inférieur. Par exemple, le .gouv est un domaine de second niveau (un sous-domaine) : tous les sites en .gouv seront des sites en .fr. Ces sous-domaines sont des sortes de subdivisions d'un domaine de premier niveau. Même chose pour les domaines de troisième niveau, qui sont des subdivisions d'un domaine de second niveau.

Enfin, les domaines de plus bas niveau regroupent les pages web d'un même site. En effet, dans le cas général, toutes les pages d'un site web font partie du même domaine. Par exemple, dans le nom de domaine *www.fr.wikipedia.org*, c'est le cas du domaine *wikipedia*.

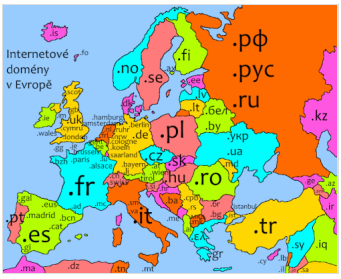
Chaque domaine de second niveau appartient à un domaine de 1er niveau, de même que les domaines de 3ème niveau appartiennent à un domaine de 2nd niveau et ainsi de suite. On peut relier les domaines en fonction de leurs relations d'inclusion, ce qui donne un arbre qui organise les domaines de manière hiérarchique. Cet arbre est ce qu'on appelle la **hiérarchie des noms de domaine**. Une partie de cet arbre est illustrée ci-dessous.



Hiérarchie des noms de domaine.

Les domaines les plus courants

Les noms de domaines peuvent être lié à la localisation géographique ou la langue du site : c'est par exemple le cas du domaine .fr ou .uk. De tels domaines sont appelés des **domaines géographiques**. La plupart de ces domaines sont liés à des pays, comme le .fr pour la France, l'Angleterre pour le .uk, l'Espagne pour le .es, etc. Ce sont des domaines dits nationaux. En tout, il existe près de 206 noms de domaines nationaux. Il existe aussi quelques pays liés à des régions à l'intérieur d'un pays, voire à des villes, comme le .paris ou le .alsace en France. Les domaines liés à des pays sont tous des domaines de 1er niveau, alors que ceux des régions/villes sont des domaines de 2nd niveau.



Liste de quelques domaines géographiques européens.

D'autres domaines ne sont pas liés à un pays, mais ont une utilisation plus large. En voici quelques-uns :

- Le *.com* a été conçu pour être le nom de domaine pour les sites commerciaux. D'ailleurs, *.com* est l'abréviation de *.commercial*. Mais son rôle a changé depuis et il est depuis ouvert à tous. Il regroupe donc des sites très différents, sans règles particulières.
- Le *.org* a été conçu pour être utilisé pour les organisations privées à but non-commercial. Comme pour le *.com*, il est depuis ouvert à tous les sites qui en font la demande. À ce propos, les sites de la wikifondation (wikipédia, wikilivres, wikicommons), sont dans le domaine*.org*.
- Le *.gov* est utilisé pour les organismes gouvernementaux américains. Les autres pays ont bien un domaine pour leurs organismes gouvernementaux, mais il s'agit d'un domaine de second niveau. Par exemple, en France, le domaine pour les organismes gouvernementaux est le *.gouv*, qui est un sous-domaine du *.fr*.
- Le *.edu* est utilisé pour les organismes d'enseignement de grande ampleur, comme les universités ou les grandes écoles. Il est surtout utilisé par les organismes américains, mais quelques organismes non-américains ont un domaine en *.edu*.
- Et il y en a bien d'autres, comme le *.info*, le *.net*, le *.arpa*, le *.int*, le *.mail*, etc.

La conversion des noms de domaine en adresses IP

Comme dit plus haut, un nom de domaine identifie un ordinateur/serveur bien précis sur le net. Mais l'accès à un fichier sur ce dit serveur doit se faire par des communications basées sur le réseau IP, les noms de domaine ne pouvant pas servir directement pour communiquer. Pour lire une page web, le navigateur web doit découvrir l'adresse IP du serveur, à partir de son nom de domaine. Cette traduction d'un nom de domaine en IP peut se faire de plusieurs manières différentes : soit avec un fichier pré-configuré, soit avec un protocole dédié (le protocole DNS).

Notons que distinguer les adresses IP et les noms de domaine est plus simple pour les être humains (une suite de mots est plus facile à retenir qu'une suite de nombres), mais a aussi d'autres avantages. Par exemple, cela permet de changer plus facilement un serveur pour un autre, tout en gardant le même nom de domaine. Pour en donner un exemple, cela permet d'utiliser un serveur de sauvegarde en complément du serveur principal. Si le serveur principal tombe en panne, la mise à jour des correspondances *ip* <-> *nom de domaine* suffit pour faire le remplacement.

Le fichier HOST

Au tout début d'internet, les correspondances entre IP et URL étaient mémorisées dans des fichiers HOSTS.TXT, qui existe toujours sur certains systèmes d'exploitation.

IP-Address	Name	Synonym
213.128.193.115 144.206.160.40	Msk-server Apollo	mail www.apollo.com

Exemple d'entrée dans le fichier Host.txt.

Ce fichier était accessible sur un serveur dédié, maintenu par le Network Information Center. Mais cette méthode a rapidement montré ses limites avec l'augmentation du nombre de sites. Vu le nombre actuel de sites web, on ne peut pas en garder un gros annuaire dans un seul et unique fichier sur chaque ordinateur.

Cependant, le fichier host.txt est toujours utilisé par les systèmes d'exploitation modernes et les navigateurs web peuvent l'utiliser comme bon leur semble. Modifier le fichier host.txt permet de bloquer des sites web : il suffit de leur attribuer une adresse IP invalide. Cette technique est une des techniques utilisée par certains logiciels de contrôle parental. Elle est aussi utilisée par des antispywares comme Spybot : celui-ci bloque des sites web conçus pour propager des spywares, en les bloquant via le fichier Host.txt.

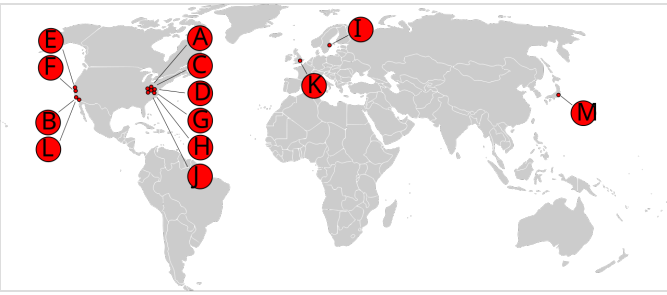
Le protocole DNS

De nos jours, le navigateur ne connait pas l'IP qui correspond à l'URL, et il doit la récupérer sur le net. Des serveurs naissent et meurent tous les jours, et l'IP associée à une URL peut ainsi changer. Pour récupérer cette IP, le navigateur va devoir utiliser un protocole (oui, encore un) : le **protocole Domain Name System** (DNS). Ce protocole mémorise les correspondances IP - URL sur plusieurs serveurs DNS. L'ensemble de ces serveurs DNS contient en quelque sorte l'annuaire d'internet. Le protocole DNS est utilisé avant toute communication avec un serveur web. D'abord, le navigateur web communique avec des serveurs DNS pour récupérer l'IP du serveur, et ensuite il utilise cette IP pour télécharger la page web demandée. Le protocole DNS indique comment on doit interroger les serveurs DNS et comment ceux-ci doivent répondre aux demandes qui leur sont envoyées. C'est un simple standard de communication, du moins pour simplifier.

Les serveurs racine

Si les noms de domaines sont structurés de manière hiérarchique, il est évident que cette organisation se retrouve pour la traduction en adresses IP. Quand un navigateur veut traduire un nom de domaine en IP, il va interroger un **serveur racine**.

Au début d'Internet, les serveurs principaux étaient au nombre strict de 13 et étaient nommés serveurs A, B, C, D, E, F, G, H, I, J, K, L, et M.

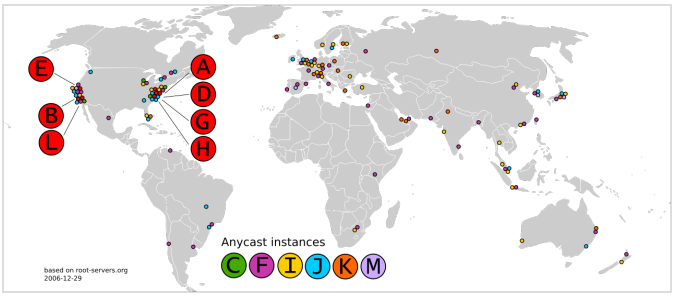


Serveurs DNS racine historiques.

Name	Organization	City, State/Province	Country
A	Network Solutions, Inc	Herndon, VA	USA
B	Information Sciences Institute, University of Southern California	Marina Del Rey, CA	USA
C	PSINet	Herndon, VA	USA
D	University of Maryland	College Park, MD	USA
E	National Aeronautics and Space Administration	Mountain View, CA	USA
F	Internet Software Consortium	Palo Alto, CA	USA
G	Defense Information Systems Agency	Vienna, VA	USA
H	Army Research Laboratory	Aberdeen, MD	USA
I	NORDUNet	Stockholm	Sweden
J	(TBD)	Herndon, VA	USA
K	RIPE-NCC	London	UK
L	(TBD)	Marina Del Rey, CA	USA
M	WIDE	Tokyo	Japan

Liste des 13 serveurs racines principaux.

De nos jours, les serveurs racine principaux sont complétés par des serveurs relais, ainsi que par des serveurs de rechange, qui s'activent quand un serveur tombe en panne ou est surchargé.

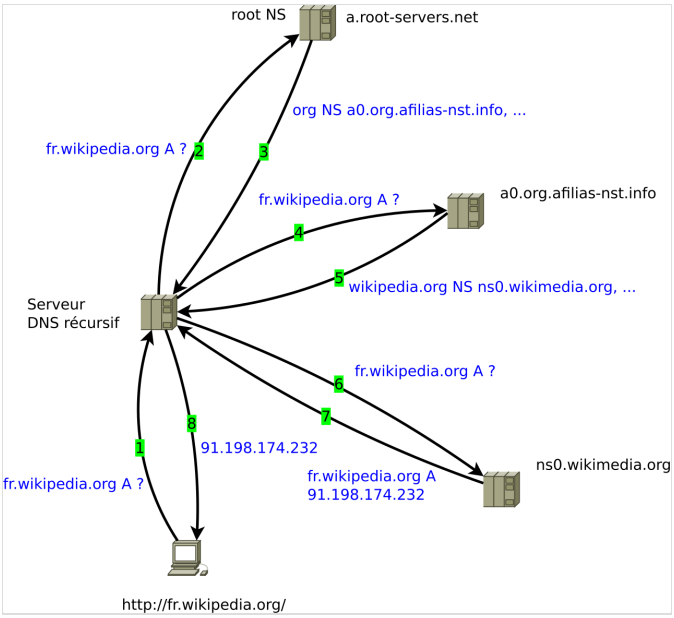


Serveurs DNS racine actuels.

Le parcours récursif de la hiérarchie des domaines

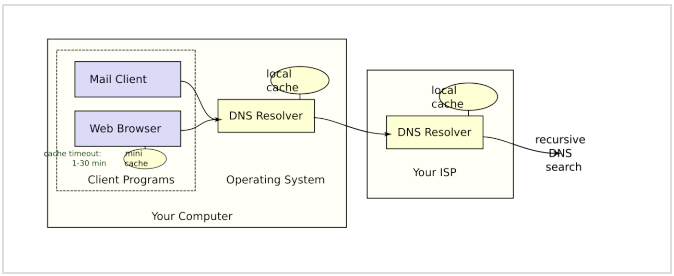
Prenons un client DNS qui veut consulter le site fr.wikipedia.org, mais qui ne connait pas l'IP qui correspond. Le navigateur connait les adresses IP des serveurs racine, qui sont des adresses fixes et réservées. Il peut donc interroger le serveur racine et lui demander quel est l'IP du site. Il lui envoie une requête DNS, qui transmet le nom de domaine et auquel le serveur doit répondre. Le serveur DNS peut répondre de deux manières : soit le serveur connait l'IP qui correspond au nom de domaine, soit il peut donner l'IP d'un serveur qui devrait connaitre cette adresse. Dans le premier cas, l'IP est renvoyée par le serveur et la recherche s'arrête. Dans le second cas, le serveur racine renvoyer l'IP d'un autre serveur DNS, qui peut correspondre au nom de domaine associé. Le processus de recherche continue tant qu'on lui fournit une adresse de serveur DNS qui peut contenir l'IP voulue.

Dans le détail, tout commence par une question envoyée au serveur racine. Le serveur racine ne connait pas l'IP du site web demandé, quel qu'il soit. Par contre, il sait quels sont les serveurs associés à chaque domaine de premier niveau, et peut renvoyer leur adresse IP. Le client reçoit la réponse et renvoie sa requête à l'IP envoyée, au serveur de premier niveau. Le serveur DNS de premier niveau renvoie alors l'adresse d'un serveur chargé du domaine de second niveau. Et ainsi de suite : le processus se poursuit jusqu'à ce qu'on tombe sur l'IP recherchée.



Example of an iterative DNS resolver

La plupart des nœuds réseaux mémorise les réponses aux demandes les plus courantes dans une portion de RAM : le **cache DNS**. Par exemple, tout ordinateur dispose d'un cache DNS dans sa mémoire, qui permet de mémoriser les IP des sites récemment visités. Quand un site est visité la première fois, on conserve son IP, afin de la réutiliser lors des prochains accès. Pas besoin de demander l'IP du site à chaque fois, seul le premier accès entraine une requête DNS. Ainsi, les réponses aux requêtes les plus fréquentes peuvent être lues depuis ce cache, ce qui est plus rapide. Cependant, les résultats sont effacés du cache après un certain temps, au cas où l'IP associée à un nom de domaine change.

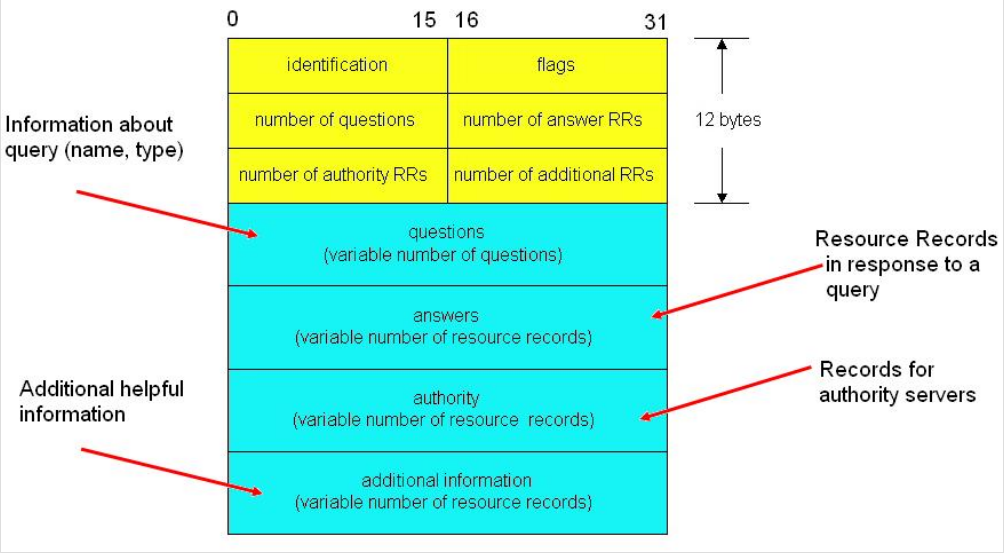


Intégration du DNS dans les navigateurs web et les FAI.

Les requêtes et réponses DNS

Les requêtes envoyées aux serveurs DNS ont le même format que les réponses de ces serveurs. Le paquet DNS est composé d'un en-tête suivi par une ou plusieurs requêtes/réponses de taille fixe.

- L'*en-tête DNS* est structuré comme suit :
  - Il commence par un champ d'identification qui contient le numéro à la transaction DNS. Ce numéro permet de savoir quelle réponse est associée à quelle requête : par exemple, la réponse numéro 15 contient les IP demandées par la requête numéro 15.
  - Il est suivi par quelques *flags*, des bits qui précisent quelques options déterminées.
  - Le reste donne le nombre de requêtes et de réponses DNS dans le paquet DNS. Le nombre de requêtes est variable dans le paquet, et il n'est pas rare qu'il y en ait plusieurs dans le même paquet, mais elles sont toutes placées les unes à la suite des autres. Il en est de même avec le nombre de réponses : le serveur peut rassembler plusieurs réponses dans un seul paquet.
- L'en-tête est suivi par les requêtes et les réponses DNS, qui forment le *contenu du paquet DNS*.
  - Chaque requête contient l'adresse URL à traduire, ainsi que quelques informations annexes.
  - Les requêtes sont ensuite suivies par les réponses de la part du serveur.
  - D'autres informations additionnelles sont placées à la fin du paquet.



Message DNS, valable autant pour les réponses que pour les requêtes.

Pour ceux qui veulent en savoir plus, je conseille la lecture du wikilivre sur le DNS, disponible à cette adresse :

- Système de noms de domaine

Le protocole HTTP

Une fois que le DNS a permis d'obtenir l'IP du serveur web, il est temps d'échanger des informations avec le serveur. Pour les communications entre serveur et client web, il existe un protocole dédié : l'**HyperText Transfert Protocol**, aussi appelé HTTP. Le HTTP est pris en charge par les navigateurs web et par les serveurs web. Les navigateurs web sont installés sur l'ordinateur client, les serveurs étant installés...sur les serveurs. Les échanges client-serveur se basent sur le protocole TCP : on dit que HTTP est basé sur le TCP. Il a existé plusieurs versions du protocole HTTP, la toute première étant la version 0.9 et la dernière la 1.1 (à l'heure où j'écris ces lignes, janvier 2016). Il peut paraitre bizarre que la première version soit la 0.9 et non la 1.0. Mais il faut savoir que la 0.9 n'avait pas de numéro de version à la base, l'introduction des numéros de version s'étant fait en catastrophe à partir de la version 1.0. L'ancienne version sans numéro a alors été renommée en HTTP 0.9.

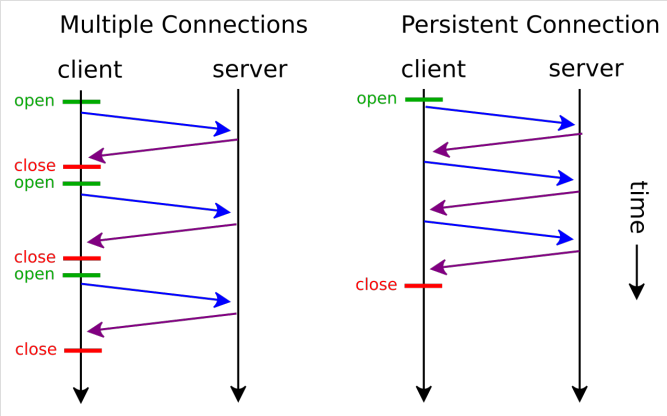
Le **HTTPS** est une version chiffrée du HTTP, où les commandes sont transmises après avoir étés cryptées. Cette version du HTTP existe pour une raison simple : avec le HTTP normal, un attaquant peut parfaitement intercepter les paquets et avoir accès à leur contenu. Et si ce contenu est votre code de carte bleu, que vous avez saisi pour faire des achats en ligne, cela peut donner une belle catastrophe. Pour éviter cela, le HTTP utilise un système de chiffrement dit asymétrique, qui empêche toute attaque de ce genre. Ce système de chiffrement est basé sur le protocole **Transport Layer Security** aussi appelé TLS.

Les connexions HTTP

Comme on l'a dit plus haut, HTTP est un protocole en mode connecté : le client doit se connecter au serveur avant de pouvoir lui envoyer des commandes HTTP. C'est pour cette raison qu'HTTP est basé sur le protocole TCP, qui a un mode connecté, et non sur UDP, protocole sans connexion. Il arrive que le client et le serveur doivent faire plusieurs échanges successifs et communiquer sur une période de temps assez longue. On se retrouve alors avec un choix à faire : est-ce que chaque échange ouvre et ferme une connexion dédiée, ou est-ce que ces échanges sont pris en charge par une seule connexion ?

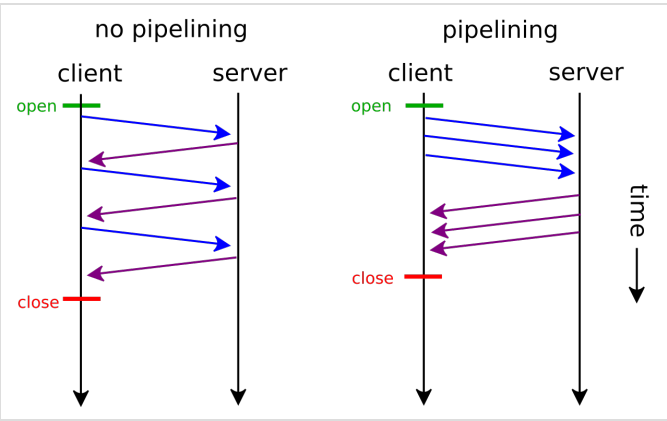
Dans le premier cas, chaque échange a sa propre connexion. Tout envoi de commande HTTP ouvre une connexion, qui sera fermée lors de la réponse du serveur. C'est ce qu'on appelle des **connexions non-persistantes**.

Dans le second cas, le client ouvre une connexion avec le serveur, effectue autant d'échanges qu'il souhaite avec cette connexion et ne la ferme qu'une fois l'ensemble des échanges terminés. C'est ce qu'on appelle des **connexions persistantes**. Avec elles, les connexions TCP ne sont pas fermées après que le serveur a répondu à une commande : elles peuvent servir pour plusieurs commandes successives. Cela permet d'économiser des connexions TCP, chose qui améliore quelque peu les performances. Les termes persistants et non-persistants caractérisent bien la nature de ces connexions. Avec les versions 0.9 et 1.0 du HTTP, les connexions sont non-persistantes. Les versions suivantes du HTTP utilisent les connexions persistantes.



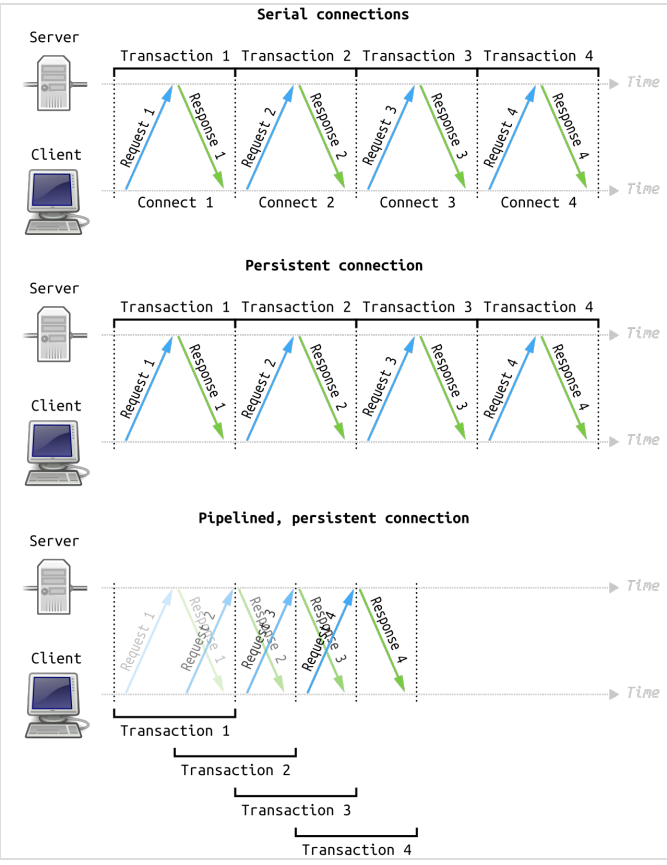
Connexions persistantes HTTP.

Une autre optimisation permise par le HTTP 1.1 est le **pipelining HTTP**. Avec cette technique, un client HTTP peut envoyer une nouvelle commande, sans attendre que le serveur réponde à la précédente. Au lieu d'envoyer les commandes unes par unes, on peut les envoyer en rafales.



Pipelining HTTP.

Le tout est résumé dans ce tableau :



Connexions HTTP.

Les commandes HTTP

Chaque échange du client vers le serveur, ou inversement, prend la forme d'un **message HTTP**. Les messages envoyés du client vers le serveur sont des **requêtes HTTP**, alors que ceux allant dans l'autre sens sont des **réponses HTTP**. Ces messages HTTP sont de simples fichiers texte encodés en ASCII, lisibles par tout être humain avec un cerveau en état de marche.

Les requêtes HTTP

Le protocole HTTP normalise différentes requêtes HTTP, comme GET, HEAD ou POST, qui agissent chacune sur une URL. Ces commandes sont envoyées dans des paquets TCP, le contenu de la commande étant placé dans les données du paquet. Ces commandes sont envoyées par le client, et le serveur doit obligatoirement répondre à celles-ci : ces commandes sont des ordres envoyés au serveur.

Ces commandes sont les suivantes :

- GET : obtenir la page web demandée ;
- HEAD : obtenir des informations sur la page, sans la consulter ;
- POST : envoyer une ressource sur le serveur (un message sur un forum, par exemple) ;
- PUT : remplace ou ajoute une ressource sur le serveur ;
- DELETE : supprime une ressource sur le serveur ;
- OPTIONS : obtenir les options de communications utilisées par le serveur ;
- CONNECT : commande spécialisée pour les proxys ;
- TRACE : permet de tester la liaison entre serveur et client.

Le message HTTP envoyé au serveur est un fichier texte formaté d'une certaine manière. Voici un exemple de requête HTTP :

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

On voit que la première ligne donne toutes les informations pour traiter la requête. Elle est appelée la **ligne de requête**. Elle commence par le nom de la commande, suivi de l'adresse URL demandée, elle-même suivie par la version de HTTP utilisée. La ligne de requête est suivie par une ou plusieurs **ligne d'en-tête**, qui fournissent des informations diverses. La ligne *Host* donne le nom de domaine qui doit répondre à la requête. La ligne *Connection* précise s'il faut utiliser des connexions persistantes ou non : close signifie que les connexions ne doivent pas être persistantes, alors que open les autorise. La ligne User-agent donne des informations sur le navigateur web à l'origine de la requête. La ligne *Accept-language* précise la langue préférée pour la réponse.

Les réponses HTTP

Le serveur répond à ces commandes en envoyant un paquet au contenu standardisé. Celui-ci peut contenir la page web demandée, pour répondre aux commandes GET ou HEAD, par exemple. Mais dans tous les cas, le serveur web indique si tout s'est bien passé ou si une erreur a eu lieu. Pour cela, il renvoie un **code de statut HTTP**, qui indique si tout s'est bien passé et quelles sont les erreurs qui ont eu lieu1. Par exemple, il va renvoyer une 404 si la ressource demandée n'a pas été trouvée sur le serveur. Les plus courants sont :

- 200 : tout s'est bien passé ;
- 301 et 302 : redirection vers une autre page ;
- 403 : accès refusé ;
- 404 : page non trouvée ;
- 500 : erreur interne au serveur.

Le format des messages de réponse est assez simple à comprendre. Comme pour les requêtes, on trouve une ligne de requête, suivie d'une ligne d'en-tête, et de la page demandée. La première ligne indique la version de HTTP utilisée, suivie du code de statut et d'une phrase. Les lignes d'en-tête fournissent d'autres informations, comme le statut des connexions, la date d'envoi, le type de serveur, etc.

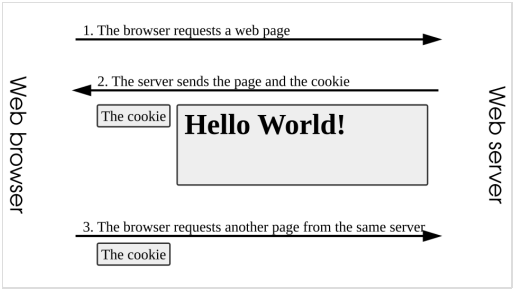
```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 09 Aug 2011 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
Données de la page
```

Les cookies HTTP

HTTP est ce qu'on appelle un **protocole sans état**, à savoir que le serveur ne mémorise aucune information sur le client. En conséquence, quand le serveur reçoit une requête, il la traite toujours de la même manière. Un client peut ainsi envoyer plusieurs fois de suite la même requête et recevoir la même réponse : le serveur ne va refuser les requêtes suivantes sous prétexte qu'elles ont déjà été servies. Cette particularité permet de fortement simplifier le protocole, sans compter qu'elle permet un gain de performance non-négligeable par rapport à un protocole avec état. Le traitement d'une requête ne demande pas l'accès à une liste d'informations client ou à un historique de connexions, n'a pas besoin de gaspiller de la RAM pour stocker des informations client, etc. C'est pour cela qu'il existe de nos jours des serveurs capables de traiter plusieurs millions, voire milliards, de connexions simultanées.

Cependant, certaines applications ont besoin de stocker des informations spécifiques à chaque client. Pour cela, HTTP déporte le stockage de ces informations sur le client, au lieu de le mettre sur le serveur. Dans le détail, HTTP permet le stockage sur le client de fichiers utiles pour le serveur, qui sont nommés **cookies**. Ceux-ci sont des fichiers que le serveur envoie au client et que ce dernier conserve dans sa mémoire. Dit autrement, les cookies sont des fichiers enregistrés sur votre ordinateur, par les sites web. Leur utilité est de mémoriser des informations sur votre ordinateur, et non sur le serveur lui-même. Ils peuvent contenir absolument tout et n'importe quoi, tant que le site aura jugé utile d'enregistrer ces informations dans le cookie. Par exemple, ils peuvent y enregistrer vos MDP et login, ce qui vous permet de ne pas avoir à les saisir à chaque connexion, de vous maintenir connecté. Les sites d'e-commerce sauvegardent aussi les paniers de produits réservés/sélectionnés avant qu'on passe commande. Ce sont des fichiers texte, et ne sont donc pas des programmes exécutables (ce ne sont donc pas des virus).

À chaque fois que vous envoyez une requête à un serveur, celui-ci lira les cookies qu'il a déposés sur votre ordinateur et peut les mettre à jour. La seule exception est pour la première requête, vu que le serveur n'a pas pût déposer de cookie sur le client. Pour résumer, tout se déroule comme suit : le client envoie une première requête au serveur, puis le serveur renvoie le fichier demandé (une page web, par exemple), ainsi que le cookie. On dit que le serveur dépose le cookie sur votre ordinateur. Les requêtes suivantes renvoient le cookie en plus de la requête proprement dite.



Échange de cookies HTTP entre client et serveur.

Les cookies peuvent être conservés dans la mémoire RAM ou sur le disque dur, ce qui permet de distinguer deux types de cookies : les temporaires et les persistants. Les **cookies temporaires** (aussi appelés cookies de session) sont maintenus dans la mémoire RAM de l'ordinateur et ne sont pas sauvegardés sur le disque dur. Lors de la fermeture du navigateur, ils sont effacés ou perdus. Ce n'est pas le cas des **cookies persistants**, qui sont sauvegardés sur le disque dur et ne s'effacent pas quand on ferme le navigateur. Le seul moyen de les supprimer est d'utiliser les options du navigateur, qui contiennent systématiquement de quoi les effacer. On peut aussi utiliser des utilitaires de nettoyage de disque, mais c'est quelque chose de déconseillé, pour diverses raisons.

Il est demandé aux navigateurs de supporter *a minima* :

- 300 cookies simultanés ;
- 4096 octets par cookie ;
- 20 cookies par « serveur » (hôte/domaine).

Les cookies traceurs

L'utilisation des cookies n'est pas forcément bienveillante, ni dans l'intérêt de l'utilisateur. Par exemple, certains services de publicité en ligne suivent les internautes sur plusieurs sites et collectent des informations grâce à un cookie de traçage enregistré sur l'ordinateur. Ces cookies permettent de tracer un utilisateur sur plusieurs sites qui partagent des éléments communs. Il suffit qu'une bannière publicitaire ou tout autre forme de publicité soit utilisée sur plusieurs sites. La visite du premier site dépose un cookie, que les autres sites consultés peuvent consulter. Le serveur de publicité sait, grâce à l'analyse du cookie de traçage, quels sont les sites visités par l'utilisateur.

Quelques extensions de navigateur web permettent de supprimer ou de bloquer ces cookies traceurs. On pourrait notamment citer les extensions Privacy Badger ou Ghostery, pour Firefox et Chrome. Les navigateurs internet récents ont commencé à prendre le problème au sérieux, en ajoutant une option qui empêche les sites de vous suivre à la trace avec de tels cookies. Cette option, appelée Do Not Track (ne me suivez pas), a cependant été mal implémentée : via cette option, on peut indiquer aux sites que l'on ne souhaite pas être pisté, mais ceux-ci font ce qu'ils veulent de cette information et peuvent parfaitement décider de la passer outre. En effet, il n'y a pas de contraintes légales quant à l'utilisation de cette option. De plus, cette option doit être activée dans les options du navigateur : elle est désactivée par défaut.

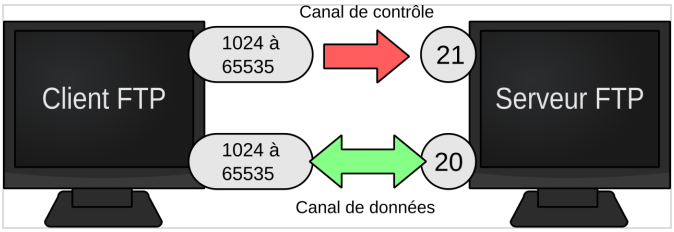
Le protocole FTP (*File Transfert Protocol*)

Le **protocole FTP** (*File Transfert Protocol*) est un protocole qui permet à un client de gérer les fichiers sur un serveur. Le protocole FTP permet au client de télécharger des fichiers sur un client (l'utilisation principale de FTP), mais aussi de modifier, remplacer ou supprimer des fichiers sur le serveur. FTP est un protocole de type client-serveur, dans le sens où il faut intervenir deux programmes : un logiciel sur le serveur, appelé serveur FTP, ainsi qu'un logiciel sur le client qui est appelé un client FTP. À l'instar d'HTTP, il existe des versions sécurisées de FTP, qui utilisent des systèmes de cryptage comme le SSL ou le TLS, qui sont appelées le FTPS.

Les connexions FTP : commande et données

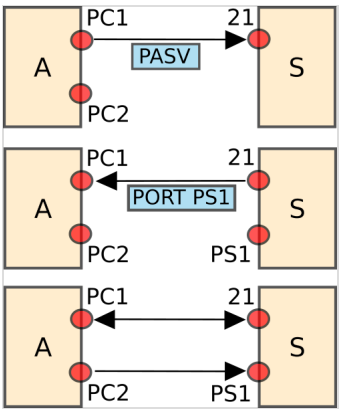
La communication entre les deux utilise des connexions TCP, deux en tout :

- Une connexion pour le transfert des données entre client et serveur, qui peut fonctionner pour transférer des fichiers dans les deux sens (du client vers le serveur, ou inversement).
- Une connexion de commande, par laquelle le client envoie des ordres au serveur (requête de téléchargement, de suppression ou de modification de fichier, autre).



Connexions du protocole FTP (en mode actif).

L'établissement des deux connexions se fait en deux étapes. Le processus est illustré ci-contre.



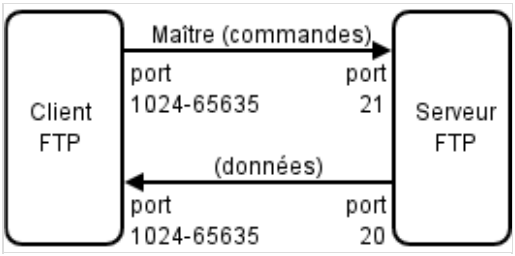
Établissement d'une connexion FTP.

- En premier lieu, le client envoie une **commande FTP** au serveur FTP. Elle prend la forme d'un paquet FTP spécial, nommé PASV ou ACTV selon le mode de connexion.
- Le serveur envoie alors une réponse au client FTP, qui dit si la connexion est acceptée et sur quelle port. Elle prend la forme d'un message composé : d'un **code réponse** de trois chiffres codés en ASCII, parfois suivie d'un message texte optionnel. Par exemple, la réponse 200 OK signifie que la connexion est acceptée.
- À la suite de cette réponse, la connexion est établie.

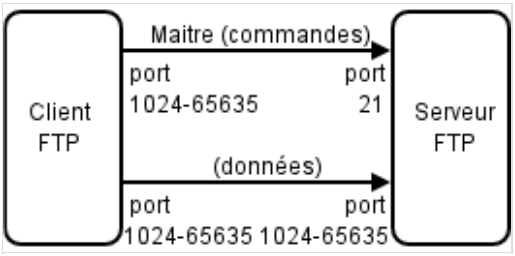
La liste des *codes réponses* des serveurs FTP et des *commandes FTP* est disponible via ce lien wikipédia :
* List of FTP commands. (https://en.wikipedia.org/wiki/List_of_FTP_commands)
* List of FTP server return code. (https://en.wikipedia.org/wiki/List_of_FTP_server_return_codes)

Le mode actif et le mode passif

Les deux connexions utilisent des ports distincts au niveau du serveur : le port 21 pour la réception des commandes et le port 20 pour le transfert des données. Cependant, il arrive que le port de transfert des données soit différent. Il faut dire qu'il peut faire l'objet d'une négociation entre client et serveur. Cela permet de distinguer deux types d'usage du FTP : le mode actif et le mode passif. Dans le mode actif, c'est le client FTP qui décide du port à utiliser pour le transfert de données. En mode passif, c'est le serveur qui décide sur quel port il émet les données. Le mode passif est surtout utilisé quand le client est protégé par un pare-feu, car il est le seul mode possible, le mode actif posant quelques problèmes avec les pare-feu.



FTP mode actif



FTP mode passif

1. https://slideplayer.fr/slide/1631716/
