Les réseaux informatiques

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

---

Introduction aux réseaux

Outre ces différences de protocoles, les réseaux sont tous très différents les uns des autres. 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.

Classification : LAN, WAN, PAN, et Internet

Certains réseaux sont limités à une salle, voire un bâtiment. D'autres sont tellement grands qu'il 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, tout 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.

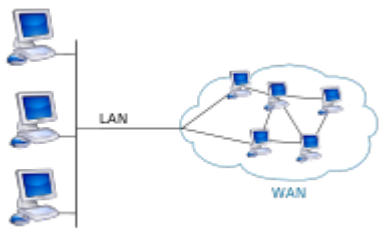
Réseaux locaux

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.

Réseaux interconnectés

A coté 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.

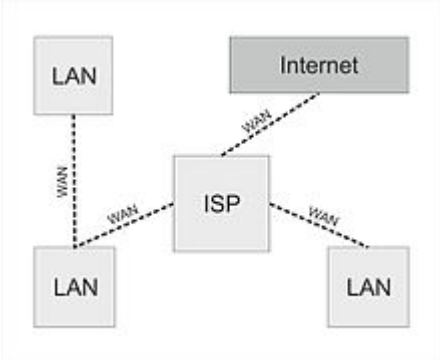


LAN et WAN

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 cout de ce genre de ligne.

Réseaux mondiaux



WAN, LAN et Internet

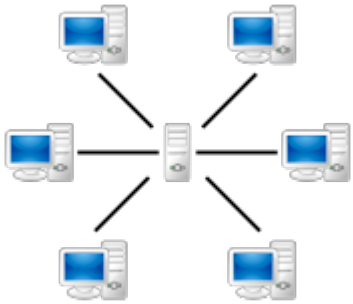
Internet est une interconnexion de réseaux à l'échelle mondiale. C'est d'ailleurs ce qui lui a valu son nom : internet est l'abréviation de interconnection of networks. Environ 47000 réseaux de grande envergure, des réseaux autonomes, sont interconnectés pour former internet. Ces 47000 réseaux sont souvent 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. Au début, il 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.

Client-serveur et pair à pair

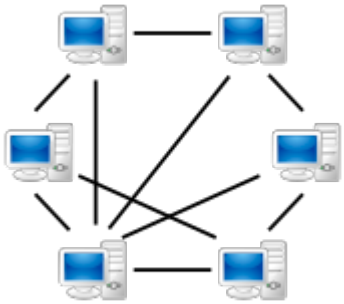
Les réseaux informatiques peuvent aussi être catégorisés selon la manière dont les ordinateurs échangent des données. On peut ainsi classer les réseaux en trois grands types :

- 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 Bitorrent utilisent des échanges en pair à pair.

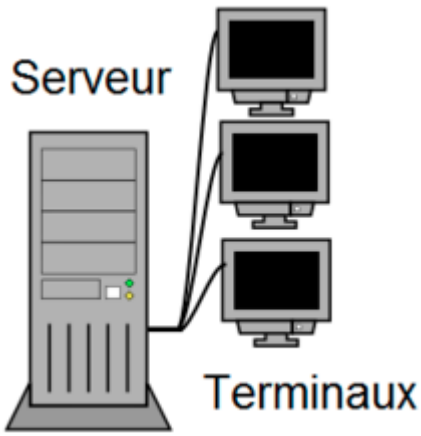


Un réseau de type client-serveur



Un réseau pair-à-pair

Clients et serveurs



Terminaux et serveur

Les premiers réseaux locaux datent des années 1960. A cette époque, les ordinateurs étaient très cher et prenaient énormément de place : un ordinateur pouvait facilement remplir une pièce complète. 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 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 se contenaient d'un clavier, d'un écran, et de quelques circuits très simples et rudimentaires. Ces terminaux se contentaient d'envoyer ou de recevoir des informations au gros ordinateurs 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 connections 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.

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, 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 est 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.

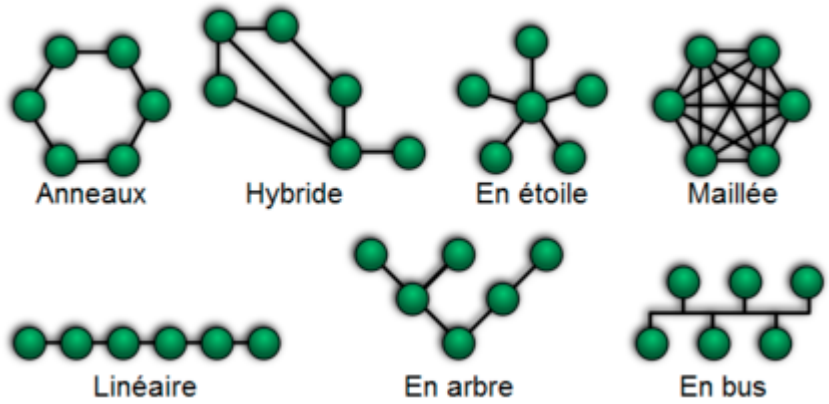
Cependant, ce système a une ambiguïté majeure. Outre l'affichage et le stockage des données, un programme doit effectuer un paquet de traitements. Dans le modèle avec des terminaux, ces traitements étaient pris en charge par le serveur central : un terminal est un matériel sans intelligence et qui n'a 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 déporté du côté du serveur, le client est qualifié de **client léger**. Dans le cas contraire, on parle de **client lourd**.

Pair à pair

L'architecture pair à pair 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. Les transferts de données se font de poste à poste, sans qu'il n'y aie de serveur bien précis.

Topologie du réseau

Les réseaux peuvent également être catégorisés par la manière dont les ordinateurs sont reliés dans un réseau, la topologie du réseau. L'image suivante présente les différentes topologies possibles. Ces différentes topologies ont chacune des avantages et inconvénients. Certaines utiliseront beaucoup de liens entre ordinateurs, alors que d'autres seront relativement économes : il faudra prévoir plus ou moins de câbles, de fils, et d'équipement réseau suivant la topologie. Suivant la topologie, les performances du réseau peuvent varier. En effet, certaines topologies ne permettent pas à deux ordinateurs de communiquer directement : les données doivent alors passer par des équipements réseau intermédiaire et se propager dans le réseau de proche en proche. 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.



Topologie des réseaux

Généralement, les réseaux locaux utilisent des topologies en étoile, en bus ou en anneau, rarement en arbre. La raison est que le nombre d'intermédiaire entre émetteur et récepteur est faible. Au plus un équipement réseau, parfois une dizaine, guère plus. Mais les réseaux mondiaux 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. Or, sur les réseaux maillés, le nombre d'intermédiaires est nettement plus important. Les données envoyées doivent donc trouver leur chemin vers le récepteur, trouver par quels intermédiaires passer pour arriver à destination. Cette problématique et celle du **routage**, à savoir 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. Nous reviendrons sur ces problématiques de routage dans quelques chapitres, lorsque nous verrons ce qu'on appelle le protocole IP.

Topologies de réseaux locaux

Un bus est un ensemble de fils qui permet de relier plusieurs ordinateurs ou composants électroniques ensemble. Avec un réseau en bus, les ordinateurs sont tous reliés à un même support de communication : un même fil, par exemple. Ils peuvent lire la donnée transmise par le support ou envoyer des données dessus. Ces réseaux utilisent peu de liens et connectent directement tous les ordinateurs entre eux. Seul un ordinateur peut émettre à un instant donné : les émissions ont lieu un ordinateur à la fois. Cependant, il se peut que plusieurs ordinateurs tentent d'émettre une donnée en même temps sur le bus, ce qui n'est pas possible : on dit qu'une collision a lieu. Dans ce cas, le bus doit contenir des mécanismes pour résoudre la situation.

Dans un **réseau en étoile**, les ordinateurs 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.

Avec la **topologie en anneaux**, les ordinateurs forment un anneau. Dans de tels réseaux, 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. Une fois arrivé, l'ordinateur de destination envoie un accusé de réception à l'émetteur. Une nouvelle émission est possible une fois l'accusé de réception reçu. La moindre panne d'un ordinateur peut mettre tout le réseau en panne, vu que l'ordinateur fautif ne peut plus

propager les données de proche en proche. Chaque ordinateur a accès à l'anneau a tour de rôle, un seul ordinateur ayant le droit d'envoyer des données sur l'anneau à la fois.

Topologies maillées

Avec la **topologie totalement maillée**, tous les ordinateurs sont reliés à tous les autres. Le nombre de liens est alors important. Plus précisémment, il est proportionnel au carré du nombre d'ordinateurs à relier, ce qui est énorme. Mais cette topologie est la plus rapide : deux PC peuvent communiquer directement, sans passer par des intermédiaires.

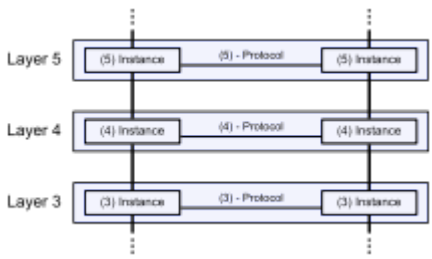
Pour économiser des liaisons point à point, il est possible de ne pas relier certains ordinateurs du réseau : on obtient un **réseau partiellement maillé**. Cela permet d'économiser des liens sans trop diminuer la performance du réseau.

Les modèles OSI et TCP

Gérer des réseaux est loin d'être une chose simple. Cela demande de gérer 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, toutes ces choses 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 compliqué, 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 réseaux 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 leur niveaux d'abstractions, dans ce qu'on appelle des **couches**. La définition d'une couche réseau est asse 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.

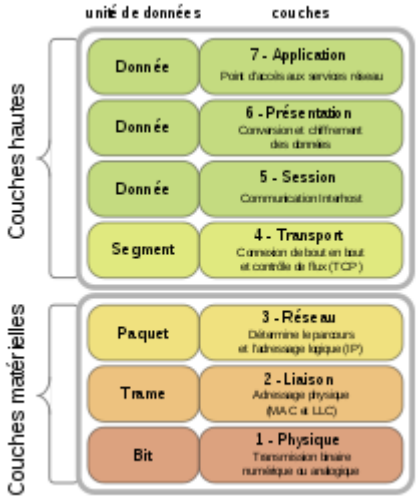


Communication entre couches.

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 aie 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.

Le modèle OSI

Le modèle OSI est de loin le plus simple. Il décrit sept couches Celles-ci portent le nom de couche liaison, internet, transport 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. De manière générale, les couches basses 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 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.



Modèle OSI

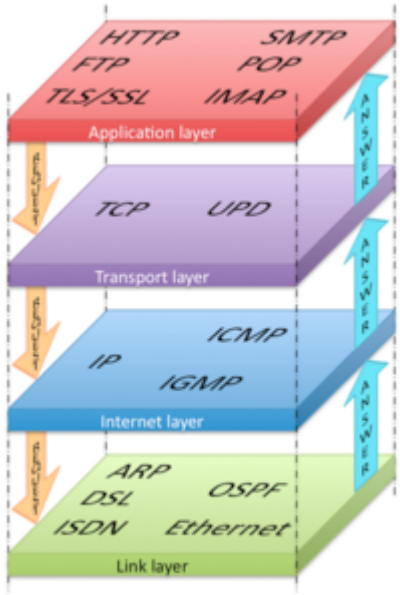
- La **couche physique** s'occupe de la transmission physique des données entre deux équipements réseaux. Elle s'occupe de tout ce qui a trait au bas-niveau, au matériel : la transmission des bits, leur encodage, la synchronisation entre deux cartes réseau, etc. Elle définit les standard des câbles réseaux, 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 des données sur un réseau local, ou entre deux ordinateurs. 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 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 cryptage ou de la compression des données transmises.
- La **couche application** prend en charge tout le reste.

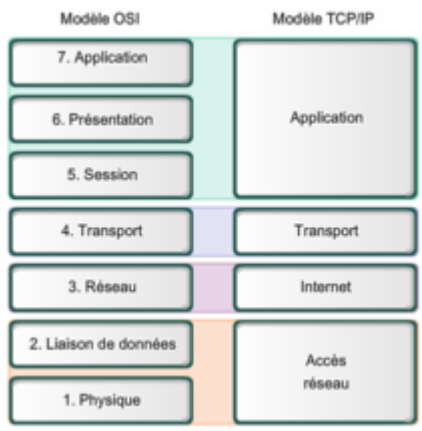
Modèle TCP/IP

Le modèle TCP/IP est plus simple qu'OSI, avec seulement quatre couches. Celles-ci sont nommées liaison, Internet, transport et application.



InternetProtocolStack

La différence avec OSI est simplement que certaines couches ont étés 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.

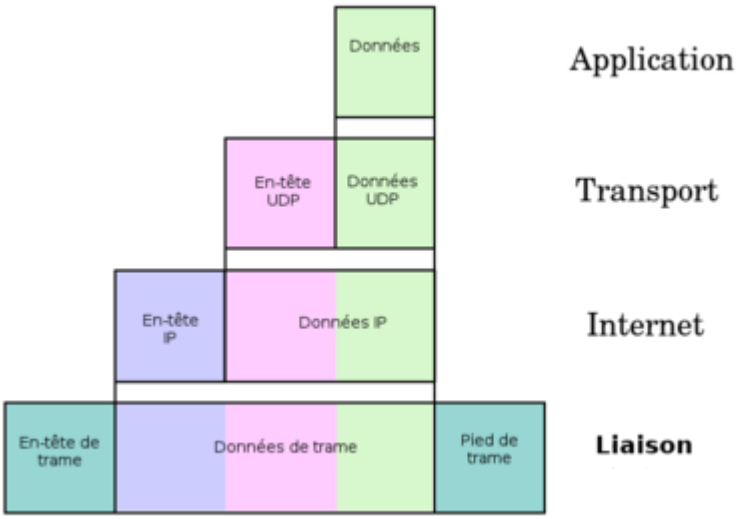


Comparaison des modèles OSI et TCP IP

L'encapsulation

Les données ne sont pas transmises telles quelles sur le réseau. Chaque protocole va devoir ajouter des informations pour garantir son fonctionnement. Par exemple, les protocoles de couche liaison vont devoir rajouter quelques données pour détecter les erreurs ou indiquer le récepteur. Même chose pour les protocoles TCP et UDP de la couche transport, qui doivent ajouter des informations sur le processus

émetteur et récepteur. Plus généralement, chaque couche va devoir ajouter des données aux données à transmettre. Pour cela, chaque couche va rajouter un en-tête aux données, cet en-tête étant simplement un ensemble structuré d'informations, placées au début des données à transmettre. Ce faisant, les en-têtes de chaque couche sont séparés les uns des autres. Lorsqu'un protocole prend en charge un paquet de données, celui-ci va ajouter son en-tête sur les données à envoyer. Lors de la réception, cet en-tête sera enlevé : les couches supérieures n'ont pas besoin des en-têtes des couches inférieures. Si la couche physique s'occupe de la transmission de bits individuels, les autres couches traitent des paquets de données contenant plusieurs bits. Ces paquets portent des noms différents selon la couche, vu qu'ils contiennent des en-tête différents : les noms de trames, paquets et segments sont utilisés pour les couches liaison, Internet et transport.



UDP encapsulation-fr

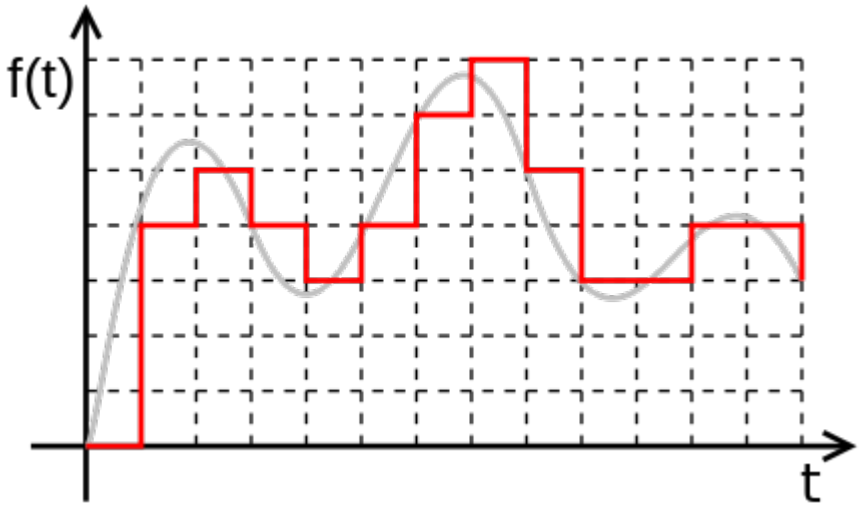
La couche physique

Dans le chapitre précédent, nous avons vu que le domaine du réseau incorporait des choses très diverses, allant du matériel réseau de base (couche physique) à des couches logicielles de bien plus haut niveau (couche transport, application, ...). La formalisation de cette idée, les couches OSI ou TCP/IP, sont certes difficiles à comprendre, mais vont structurer le cours qui va suivre. Dit autrement, nous allons voir chaque couche l'une après l'autre. Et il est tout naturel de commencer par la première couche : 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 standard des câbles réseaux, des fils de cuivre, du WIFI, de la fibre optique, ou de tout autre support électronique de transmission.

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 récepteurs, soit dans le vide (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.

Quelque soit le support, le transfert ne peut s'effectuer qu'un bit à la fois. 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). La première, le **codage NRZ-L**, devrait vous être familière : il s'agit d'utiliser l'état haut pour coder un 1 et l'état bas pour le zéro (ou l'inverse). 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. 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. 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). Ce codage s'obtient en faisant un OU logique entre l'horloge et le flux de bits à envoyer (codé en NRZ-L). Diverses variantes existent, qui codent un 1 ou un 0 avec un front, tandis que l'autre bit est codé comme en NRZ-L. Ces différentes méthodes 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. Évidemment, chaque codage a son propre version différentielle, à savoir avec deux fils de transmission.

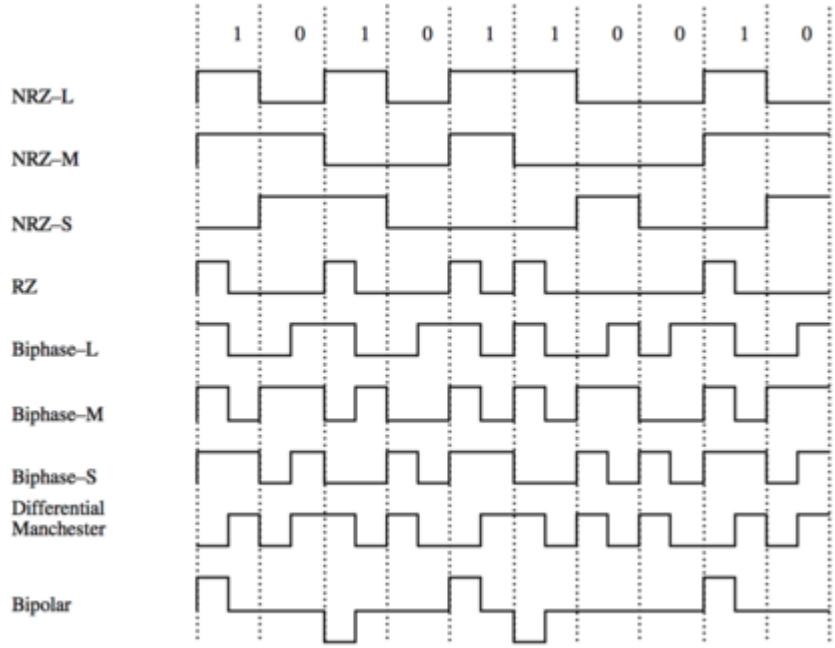
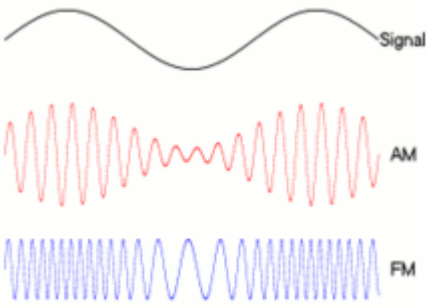


Illustration des différents codes en ligne.

Quelque soit la méthode, le bit transmis est maintenu durant un temps bien précis. Dit autrement, le fil transmet un bit (ou un paquet de bits) toutes les x millisecondes ou microsecondes. Le temps de maintien d'un bit sur le fil est ce qu'on appelle 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. Il est théoriquement possible d'utiliser plusieurs fils pour envoyer des bits en parallèle, en même temps. Par exemple, on pourrait regrouper huit fils dans un même câble pour transmettre les données octet par octet. Mais une telle transmission parallèle a des désavantages qui font qu'elle n'est plus utilisée de nos jours. Les concepteurs de câbles réseaux préfèrent transmettre les bits uns par un sur le support de transmission, à quelques exceptions près.

Le codage par modulation



Un signal modulant une porteuse en amplitude ou en fréquence.

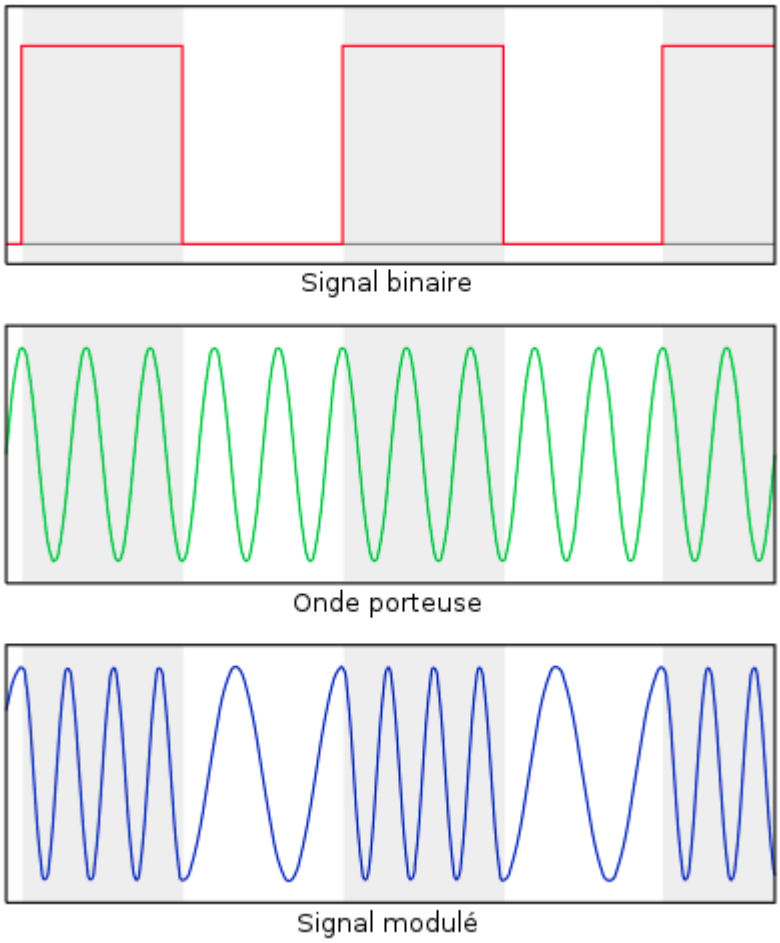
Généralement, les signaux analogiques les plus simples sont formés de tensions sinusoïdales ou cosinusoïdales. IL est possible de transmettre un signal numérique à travers un signal analogique comme une sinusoïde. Celle-ci doit simplement subir quelques transformations via des techniques dites de modulation. 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 toute les x µ-secondes, 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.

Techniques de modulation les plus simples

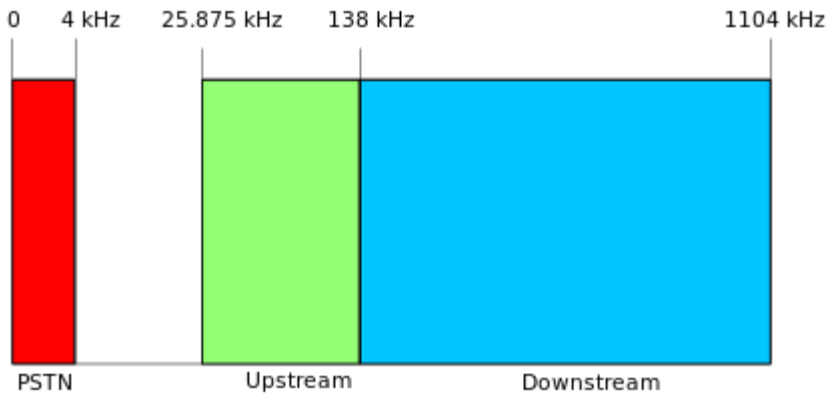
La méthode de modulation la plus simple est la **modulation d'amplitude**. Elle consiste à modifier l'amplitude de la sinusoïde en fonction du bit à transmettre. L'amplitude est alors maximale pour un 1 et minimale pour un 0. Il est aussi possible de transmettre un paquet de bits en une seule période : 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 fine, vu que la différence de tension entre deux paliers diminue avec le nombre de paliers, du moins pour une 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



Fréquences utilisées par l'ADSL.

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

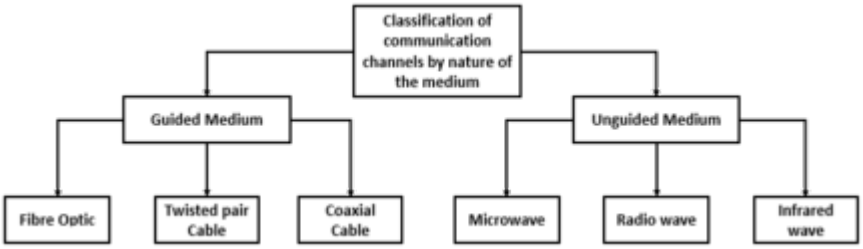
cuivre peuvent transmettre aussi bien un signal numérique (un flux de bits) qu'un signal analogique (codé avec des signaux sinusoïdaux). Que ce soit pour transmettre le signal téléphonique ou le signal Internet, le câble de cuivre est utilisé pour transmettre des tensions sinusoïdales, dans un intervalle 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.

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 et le sans-fils. Dans ce qui va suivre, nous allons voir les deux supports de transmission.

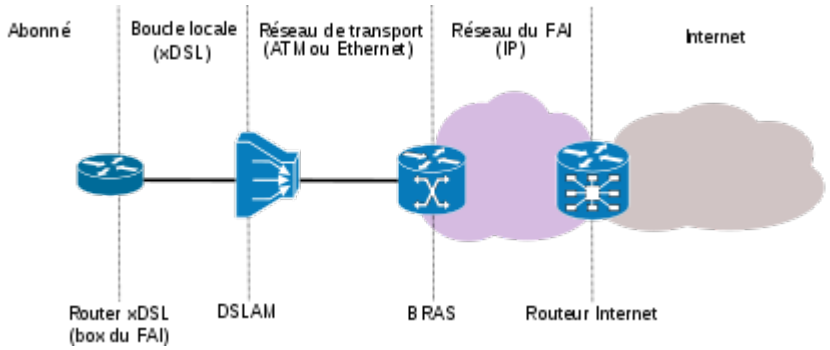
Il faut cependant noter que les spécialistes en électricité et en électronique ont cependant un jargon un peu plus complexe que "avec fils" et "sans-fils". Ils parlent à la place de liaison guidée (avec fils) et non-guidée (sans fils). Dans les deux cas, le support de transmission va propager des ondes 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'ets 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.

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, 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 sous 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

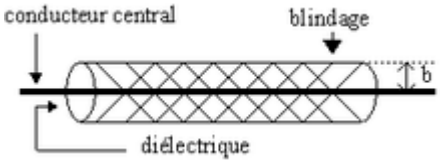
données transmises. Et il est alors impossible de savoir quelle était la donnée transmise de base.

Les lignes bifilaires (câbles torsadés)

Pour éviter l'impact des perturbations, on peut utiliser deux fils pour la transmission. 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. Ils sont composé 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. Ce sont les fameux **câbles coaxiaux**. 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 de ce type (du moins, si vous n'avez pas la fibre). Comme dit auparavant, ces câbles transmettent aussi bien les signaux analogiques que numériques. Au passage, qu'il s'agisse de paire torsadée ou de câbles coaxiaux, une paire de fils, réunie dans un câble, porte le nom de **ligne bifilaire**.

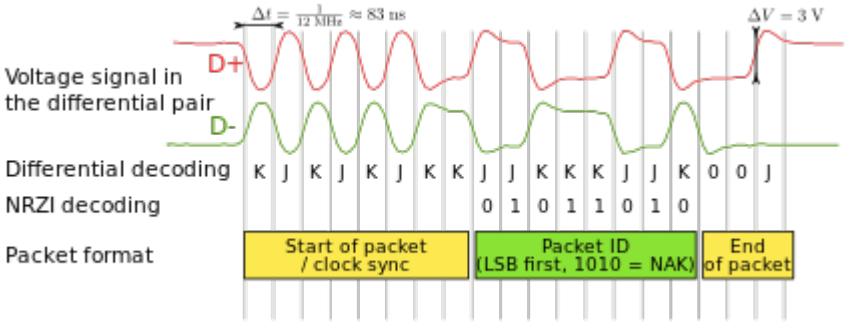
Ligne bifilaire

Schéma d'une paire torsadée.



Câble coaxial.

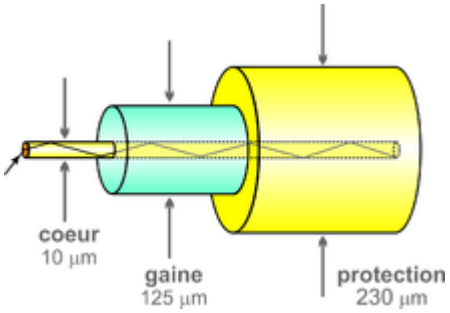
On peut se demander quelle est l'intérêt d'utiliser une paire de fils pour transmettre un signal. La raison est que cela permet une meilleure résistance aux perturbations électromagnétiques, aux parasite et autres formes de bruits qui peuvent modifier les bits transmis. Le premier fil transmet le signal, tandis que le second fil transmet une copie inversée du signal. Par exemple, le premier fil transmettra une tension positive proportionnelle au signal, tandis que l'autre fil transmettra une tension négative. Ce faisant, on utilise la différence de tension pour coder le bit. Le signal transis est égal à la différence entre la tension dans les deux fils : on l'obtient en soustrayant le signal du second fil au signal du premier. 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**. Il est notamment utilisé sur le protocole USB. Sur ce protocole, deux fils sont utilisés pour transmettre un bit, via codage différentiel.



Signal USB : exemple.

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 sur ses parois.



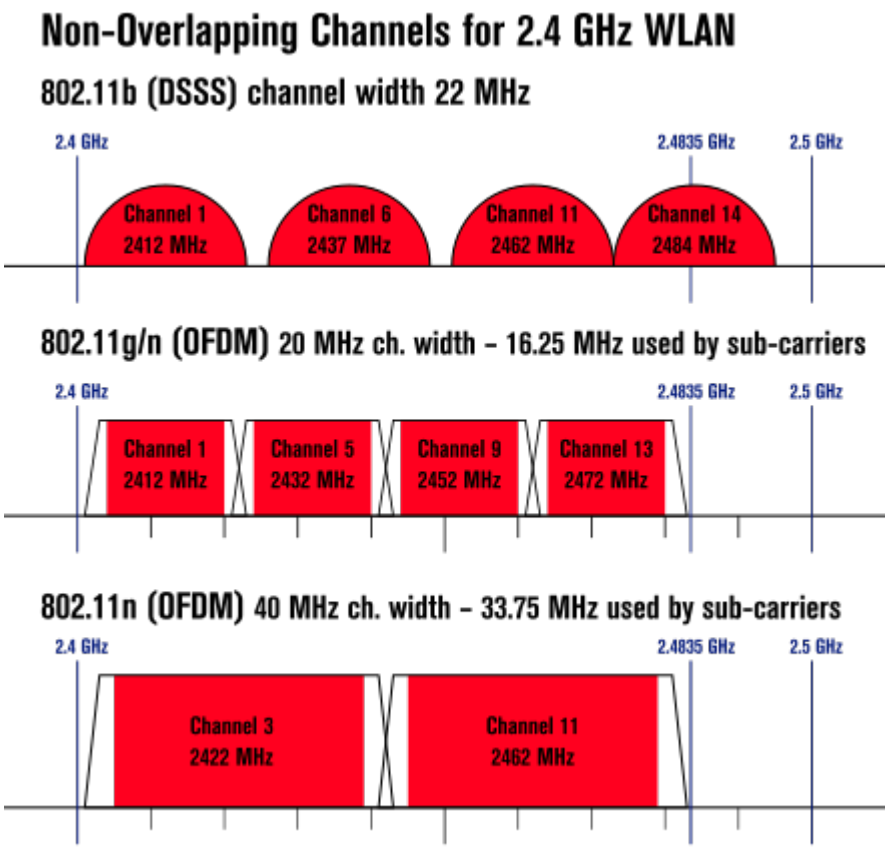
Principe d'une fibre optique.

Les supports non-guidés (technologies sans-fils)

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 faibles 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 performance 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 cannaux, ainsi que leurs limites maximales et minimales sont définie par la norme utilisée.



Canaux WIFI 2.4Ghz WLAN.

Avantages et inconvénients

Ces supports de transmission doivent transmettre les ondes électromagnétiques qui codent le flux de bit transmis. Cependant, chaque support a une réaction précise quand il est parcouru par des ondes électromagnétiques. Il va notamment en absorber une partie, ce qui fait que le signal transmis va progressivement s'atténuer avec la distance. Ce phénomène est surtout sensible sur les câbles réseaux formés de fils de cuivre, raison pour laquelle les fibres optiques sont surtout utilisées pour de grandes distances. Et c'est sans compter que les ondes électromagnétiques peuvent se disperser dans l'environnement. 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. Ce qui explique leur portée nettement meilleure.

La couche liaison

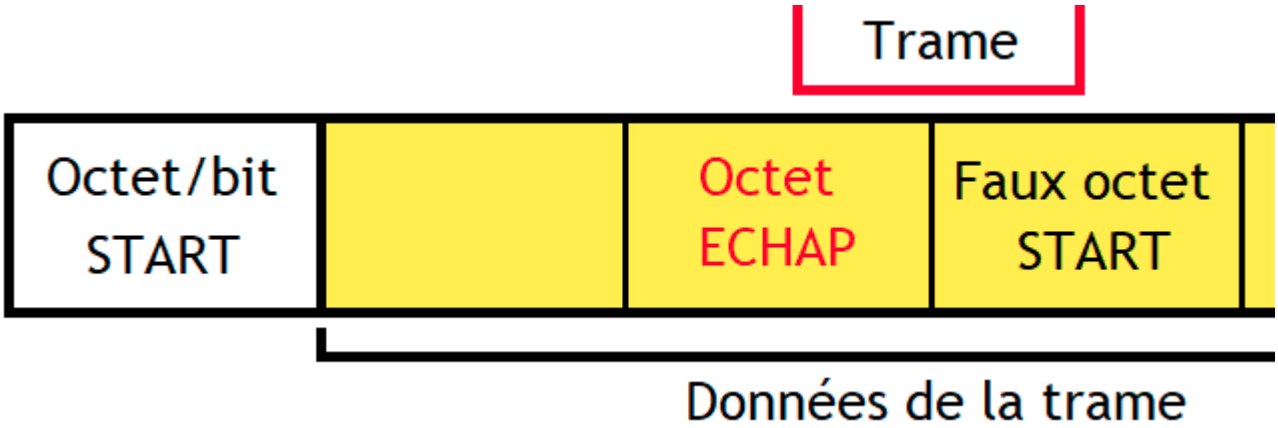
Comme dit dans le chapitre précédent, un réseau local est un réseau contenant peu d'ordinateurs, qui sont reliés entre eux par des câbles réseaux, des routeurs ou des switchs. Sur un réseau local, les données sont échangées sous la forme de paquets de données de taille fixe. Ces paquets sont émis par un ordinateur, et se propagent dans le réseau local jusqu'à l'ordinateur de destination. Ces paquets de données sont appelés des trames. La **couche liaison** prend donc en charge plusieurs fonctionnalités, qu'on peut regrouper en trois grandes fonctions : l'encapsulation des données envoyés sur le réseau et la détection/correction des erreurs de transmission.

L'encapsulation des trames

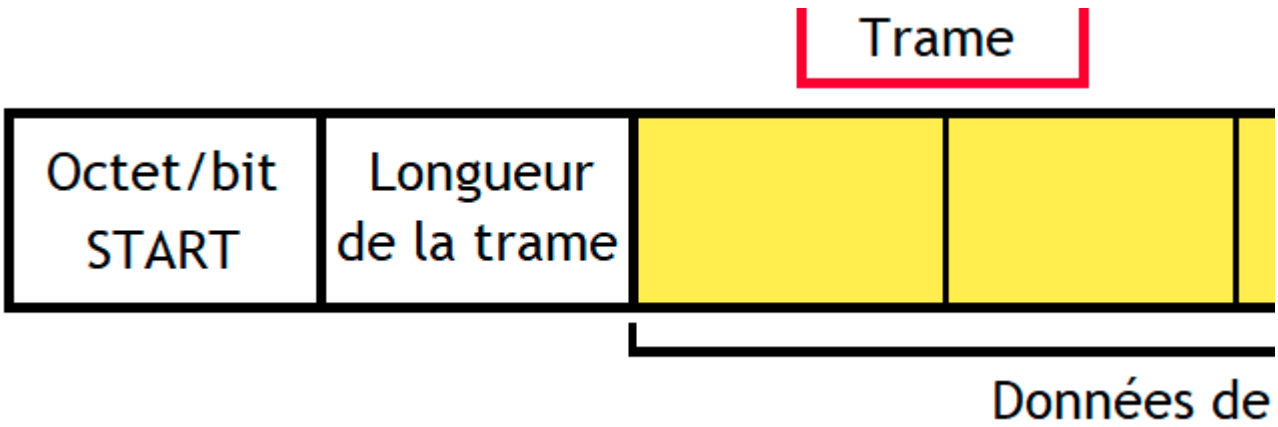
Le codage des trames indique comment l'émetteur doit envoyer les données sur le bus, ce qui permet aux récepteurs de détecter les données transmises et les interpréter. Le terme trame réfère aux données transmises : toutes les informations nécessaires pour une transmission sont regroupées dans ce qu'on appelle une **trame** qui est envoyée telle quelle sur le bus. 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.

Le codage du début et de la fin de transmission

Une transmission est un flux de bits qui a un début et une fin : le codage des trames doit indiquer quand commence une transmission et quand elle se termine. Le récepteur ne reçoit en effet qu'un flux de bits, et doit détecter le début et la fin des trames. Ce processus de segmentation d'un flux de bits en trames n'est cependant pas simple et l'émetteur doit fatalement ajouter des bits pour coder le début et la fin de la trame. De nos jours, la quasi-totalité des protocoles utilisent la même technique pour délimiter le début et la fin d'une trame. Ils utilisent un octet spécial (ou une suite d'octet), qui indique le début de la trame, et un autre octet pour la fin de la trame. Ces octets de synchronisation, respectivement nommés START et STOP, sont standardisés par le protocole. Entre ces octets de synchronisation, on trouve un en-tête standardisé, les données à transmettre et éventuellement des données de contrôle d'erreur. Un problème peut cependant survenir : il se peut qu'un octet de la trame soit identique à un octet START ou STOP. Mais il y a moyen de résoudre le problème assez simplement. Pour cela, il suffit de faire précéder ces pseudo-octets START/STOP par un octet d'échappement, lui aussi standardisé. Les vrais octets START et STOP ne sont pas précédés de cet octet d'échappement et seront pris en compte, là où les pseudo-START/STOP seront ignoré car précédés de l'octet d'échappement. Il arrive plus rarement que ces octets de START/STOP soient en réalité codés par des bits spéciaux. Un bon exemple est celui du bit I²C, où les bits de START et STOP sont codés par des fronts sur les bus de données et de commande. Il en est de même sur le BUS RS-232 et RS-485, comme nous le verrons plus tard.



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. Ce faisant, le récepteur a moins de travail d'analyse à faire : il n'a pas vraiment à analyser le contenu des octets pour savoir quoi faire, mais a juste à compter les octets qu'il reçoit. Cette méthode et la méthode précédente ont des conséquences différentes sur la taille des trames. Si la trame peut être très longue, la longueur doit être codée sur plusieurs octets : cela prend plus de place que l'octet/bit STOP. Par contre, les octets d'échappement sont économisés. Un autre défaut (très théorique) de cette approche est que 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. Mais ce défaut est plus théorique qu'autre chose.



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.

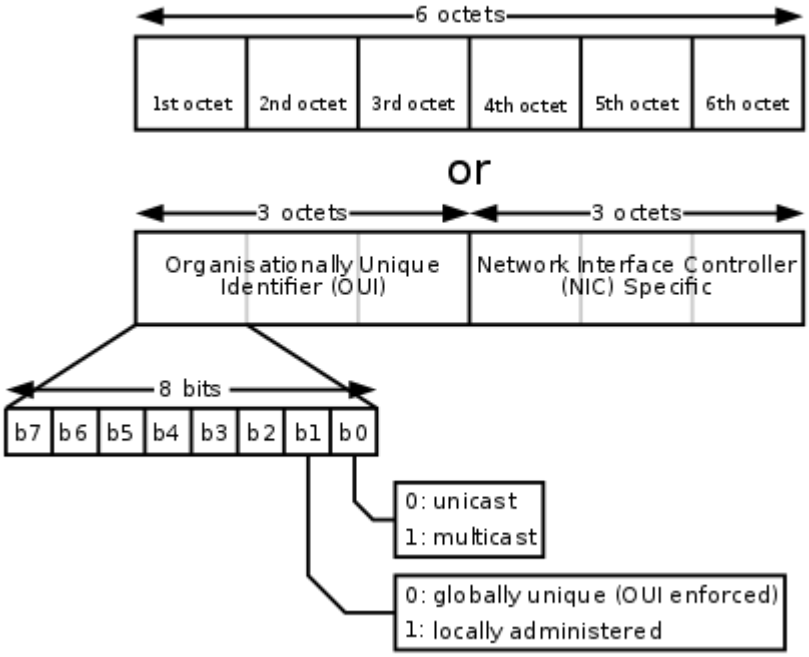
Le codage du récepteur

La trame doit naturellement être envoyée à un récepteur, seul destinataire de la trame. Or, on a vu que sur les réseaux diffusants, toute trame est envoyé à tous les ordinateurs du réseau. Ceux-ci doivent avoir un moyen quelconque pour savoir si la trame leur est destinée. Pour cela, chaque ordinateur d'un réseau diffusant reçoit un numéro qui permet de l'identifier : ce numéro est ce qu'on appelle une **adresse physique**. Toute trame contient l'adresse physique du récepteur, les ordinateurs ayant juste à comparer

cette adresse avec la leur pour savoir s'ils sont destinataires. Pour donner son destinataire, la trame doit indiquer quelle est son adresse. L'adresse en question est intégrée à la trame et est placée à un endroit précis, le plus souvent à son début. La raison à cela est que les récepteurs espionnent le support de transmission en permanence pour détecter les trames qui leur sont destinées. Ils lisent toutes les trames et en extraient l'adresse de destination : si celle-ci leur correspond, ils lisent le reste de la trame, ou l'ignorent sinon. 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 plus tôt.

De nos jours, les adresses physiques sont définies par le standard MAC. Avec ce standard, chaque adresse physique fait 6 octets (48 bits), et est appelée une **adresse MAC**. 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. Les réseaux locaux eux-même ont une adresse MAC : cela sert pour connecter des réseaux locaux entre eux, par exemple. 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. 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. Pour indiquer si l'adresse a été changée par l'utilisateur, l'adresse MAC contient un bit U/L.

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 22 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.



MAC-48 Address

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 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.

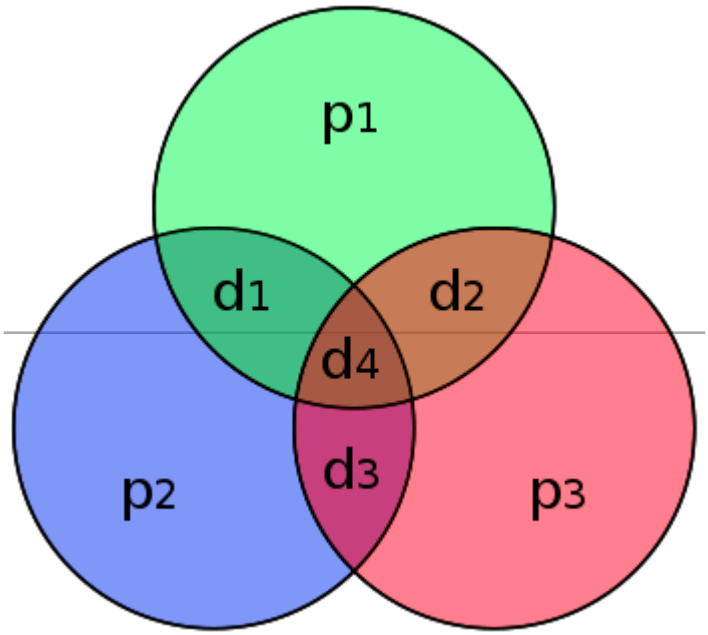
Le **bit d'imparité** est similaire au bit de parité, si ce n'est que le nombre total de bits doit être impair, et non pair comme avec un bit de parité. Sa valeur est l'inverse du bit de parité du nombre : quand le premier vaut 1, le second vaut 0, et réciproquement. Mais celui-ci n'est pas meilleur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

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 étés modifiés, sans qu'on sache lesquels.

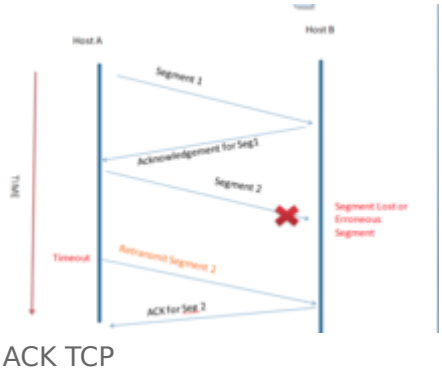
Sommes de contrôle

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***.

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.

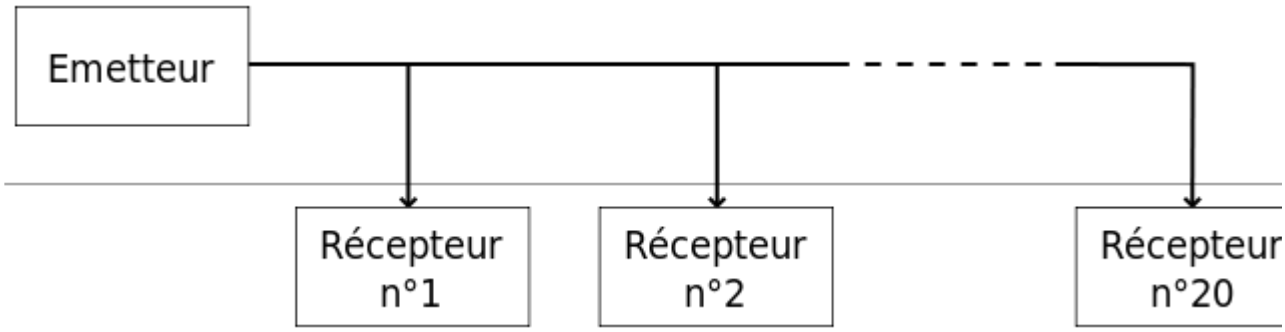
Les deux problèmes précédents peuvent être résolu en utilisant ce qu'on appelle une **fenêtre glissante**. Avec cette méthode, les trames sont envoyées les unes après les autres, sans attendre la réception des

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 étés reçues et celles à renvoyer (modulo les time-out de chaque trame). On peut remarquer qu'avec cette méthode, les trames sont parfois recues dans le désordre, alors qu'elles ont été envoyées dans le désordre. 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 recue. Lui aussi va devoir accumuler les trames recues dans une mémoire, pour les remettre dans l'ordre.

L'accès au support de communication

Réseaux sans fils et bus sont souvent regroupés sous le terme de **réseaux de diffusion**. Ceux-ci ont pour particularité de faire communiquer plusieurs machines avec un même support partagé. Quand une trame est envoyée sur un tel réseau, toutes les machines du réseau reçoivent la trame. Elles doivent alors décider si la trame leur est destinée, en comparant les adresses MAC, et agir en conséquence. Sur les réseaux non-commutés (organisés autour d'un hub, d'un anneau ou d'un bus), chaque paquet est envoyé à tous les ordinateurs du réseau. Mais seuls l'ordinateur de destination est censé prendre en compte la paquet envoyé. Pour cela, tous les ordinateurs du réseau vérifient que l'adresse MAC du destination du paquet est bien la leur. Cependant, tous les PC recevant les informations transmises, rien n'empêche un ordinateur (ou équipement réseau) d'espionner tout 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.

Réseaux sans fils et bus sont souvent regroupés sous le terme de **réseaux de diffusion**. Ceux-ci ont pour particularité de faire communiquer plusieurs machines avec un même support partagé. Les autres réseaux et topologies se basent sur un ensemble de connexions **points à points**, où deux machines sont reliées par un support de communication distinct, qui leur est dédié. La couche liaison permet à deux machines d'échanger des données soit à travers un réseau de diffusion, soit à travers une liaison point à point. Divers protocoles ont été inventés pour les liaisons point à point : le protocole PPP, ou le protocole HDLC en sont de bons exemples. Les protocoles pour réseaux de diffusion sont eux très différents, les plus connus étant Ethernet et les protocoles pour le Wifi. La principale différence entre ces protocoles est dans la gestion du partage du support de transmission, d'où leur nom de protocoles à accès multiples. Inutile par construction pour les liaisons point à point, elle est essentielle pour tout réseau de diffusion.



Réseaux en anneaux : Token Ring

Il y a peu à dire sur les réseaux non-diffusants. Cependant, les réseaux en anneaux ont besoin d'un contrôle d'accès. Sans cela, plusieurs ordinateurs peuvent envoyer des données dans l'anneau. Pour résoudre ce problème, chaque ordinateur a accès à l'anneau a tour de rôle, un seul ordinateur ayant le droit d'envoyer des données sur l'anneau à la fois. Ce droit est modélisé par un jeton, qui passe d'un ordinateur au suivant à chaque émission. Ce jeton est juste une trame spéciale : toutes les trames envoyées sur le réseau ont un bit qui indique si cette trame est le jeton ou pas. Ce jeton est conservé dans un temps déterminé et fixe, le même pour tous les ordinateurs.

L'utilisation d'un jeton permet à tous les ordinateurs d'avoir un accès équitable au réseau. On sait que tous les ordinateurs auront accès au réseau à un moment ou un autre. Évidemment, ils peuvent ne pas utiliser leur jeton s'ils n'ont rien à envoyer, et transmettent alors le jeton à l'ordinateur suivant. De plus, le temps avant de récupérer le jeton est obligatoirement fini et on peut lui attribuer une limite maximale : le temps que le jeton fasse le tour de l'anneau. C'est un avantage que les bus n'ont pas : un ordinateur peut, en théorie, monopoliser le bus durant un temps indéterminé si rien n'est prévu dans la conception du bus. Si peu d'ordinateurs accèdent au réseau, les réseaux en bus ou étoile ont une meilleure efficacité. Mais à forte charge, les réseaux en anneau ont clairement une efficacité nettement plus importante. En effet, le nombre de collisions augmente sur un réseau en bus, ce qui réduit le débit utilisé pour transmettre des données : le débit utilisé est plus important sur un réseau en anneau, vu qu'il n'y a pas de collisions.

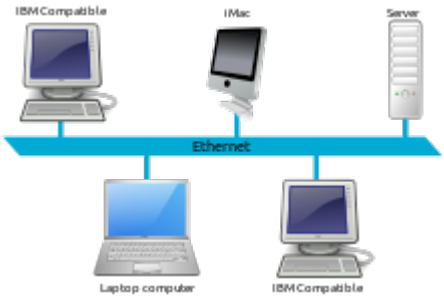
Réseaux de diffusion : détection de collisions

Sur les réseaux à base de bus, il arrive que plusieurs composants tentent d'envoyer ou de recevoir une donnée sur le bus en même temps : c'est un conflit d'accès au bus, aussi appelé une **collision**. Ce problème a lieu aussi sur les réseaux Wifi, où plusieurs machines peuvent émettre sur la même fréquence. 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 un 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 que le bus fonctionne correctement. Pour résoudre ce problème, diverses méthodes ont étés inventées et intégrées aux protocoles les plus connus.

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.



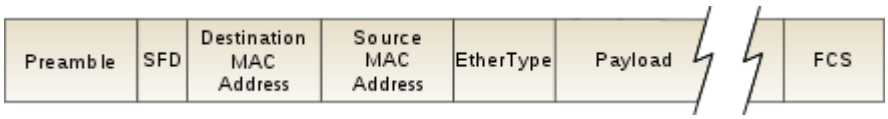
Ethernet LAN

Les trames Ethernet

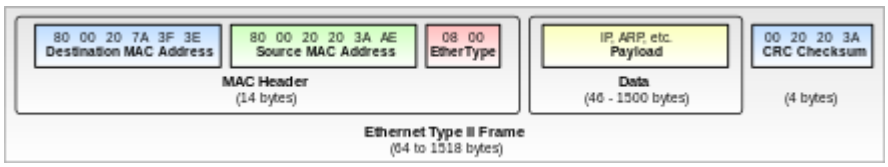
Toute trame Ethernet contient diverses informations, dont :

- l'adresse MAC du destinataire : sans cela, on ne sait pas à qui la donnée est destinée ;
- l'adresse de l'émetteur, qui peut se révéler utile ;
- 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 son nombre d'octets, sa longueur. 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 remplacé par un numéro qui indique quels sont les protocoles utilisés pour la transmission des données sur le net : 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.



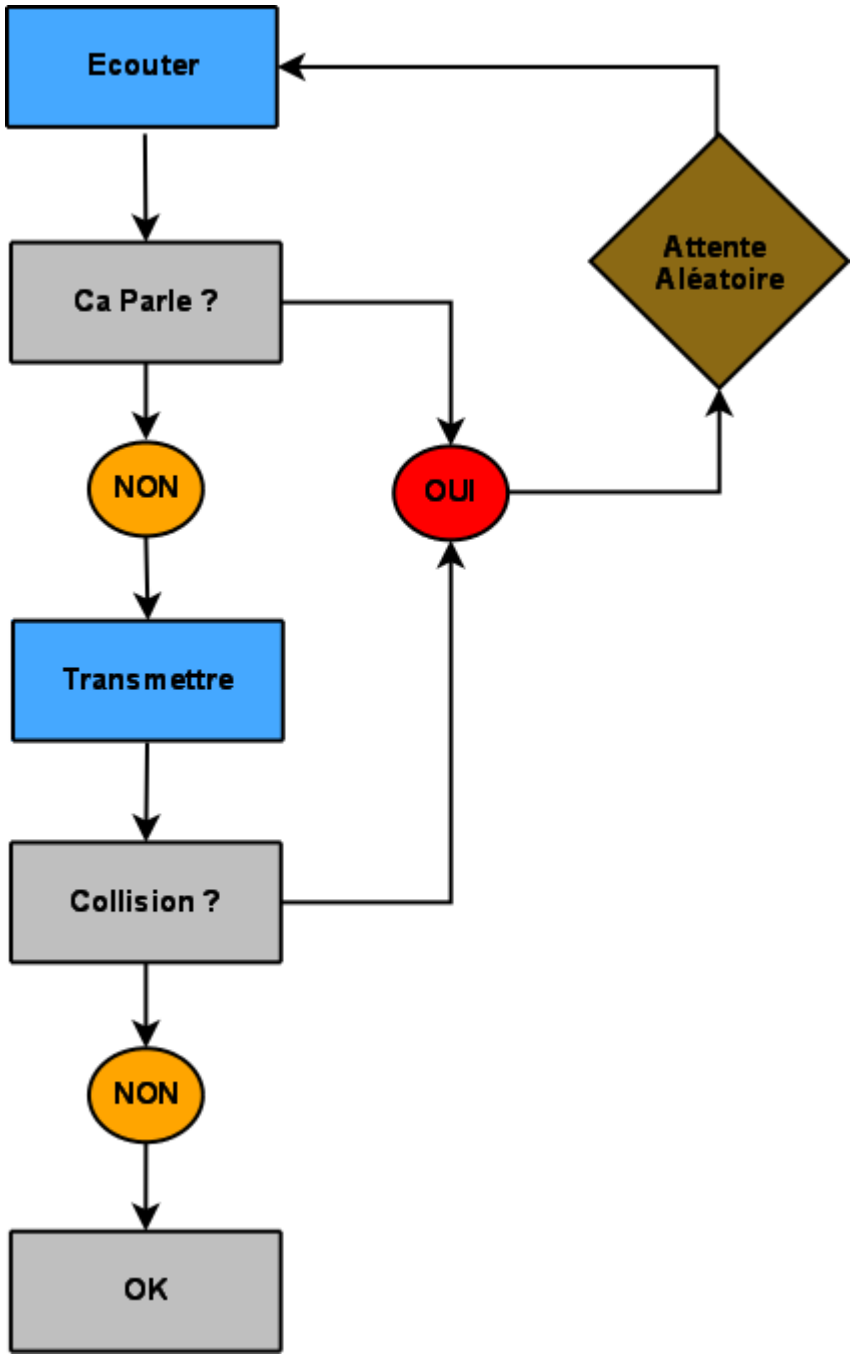
Trame Ethernet, version 2.

L'arbitrage du bus : le CSMA-CD

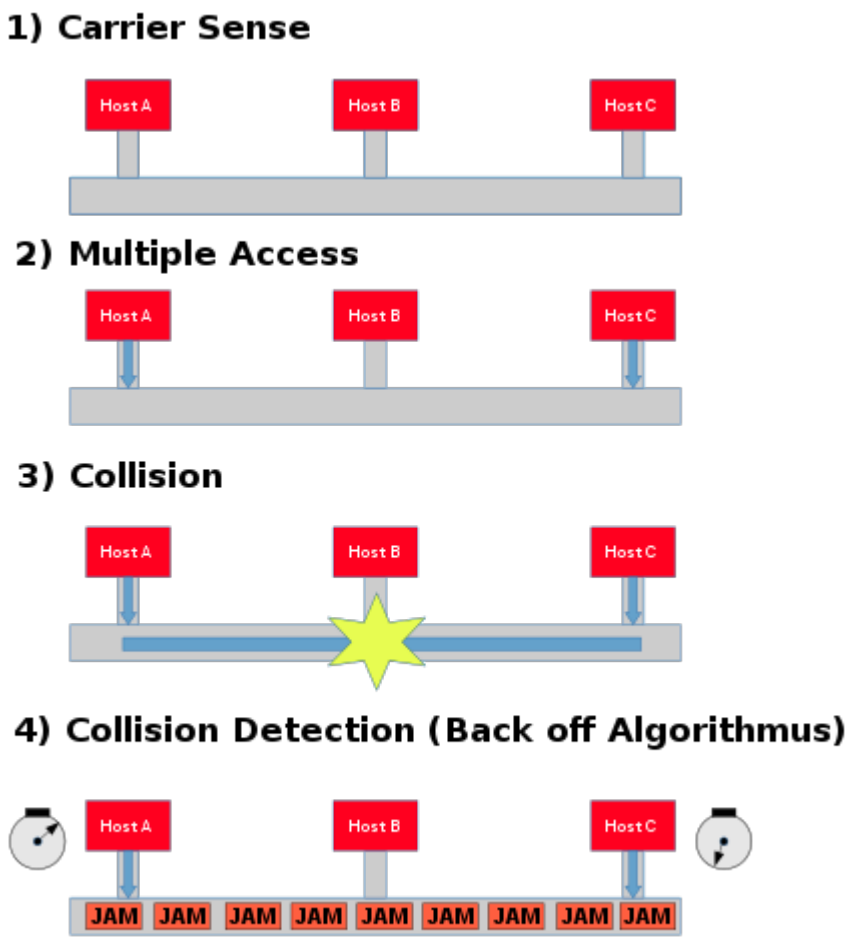
Comme tous les autres bus, le bus Ethernet doit contenir des mécanismes d'arbitrage pour résoudre la situation. Le protocole Ethernet utilise une méthode d'arbitrage spéciale : le **CSMA-CD**. De base, les machines attendent que le bus soit libre (personne n'émet) pour émettre une trame. Mais il arrive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

lieu.

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.



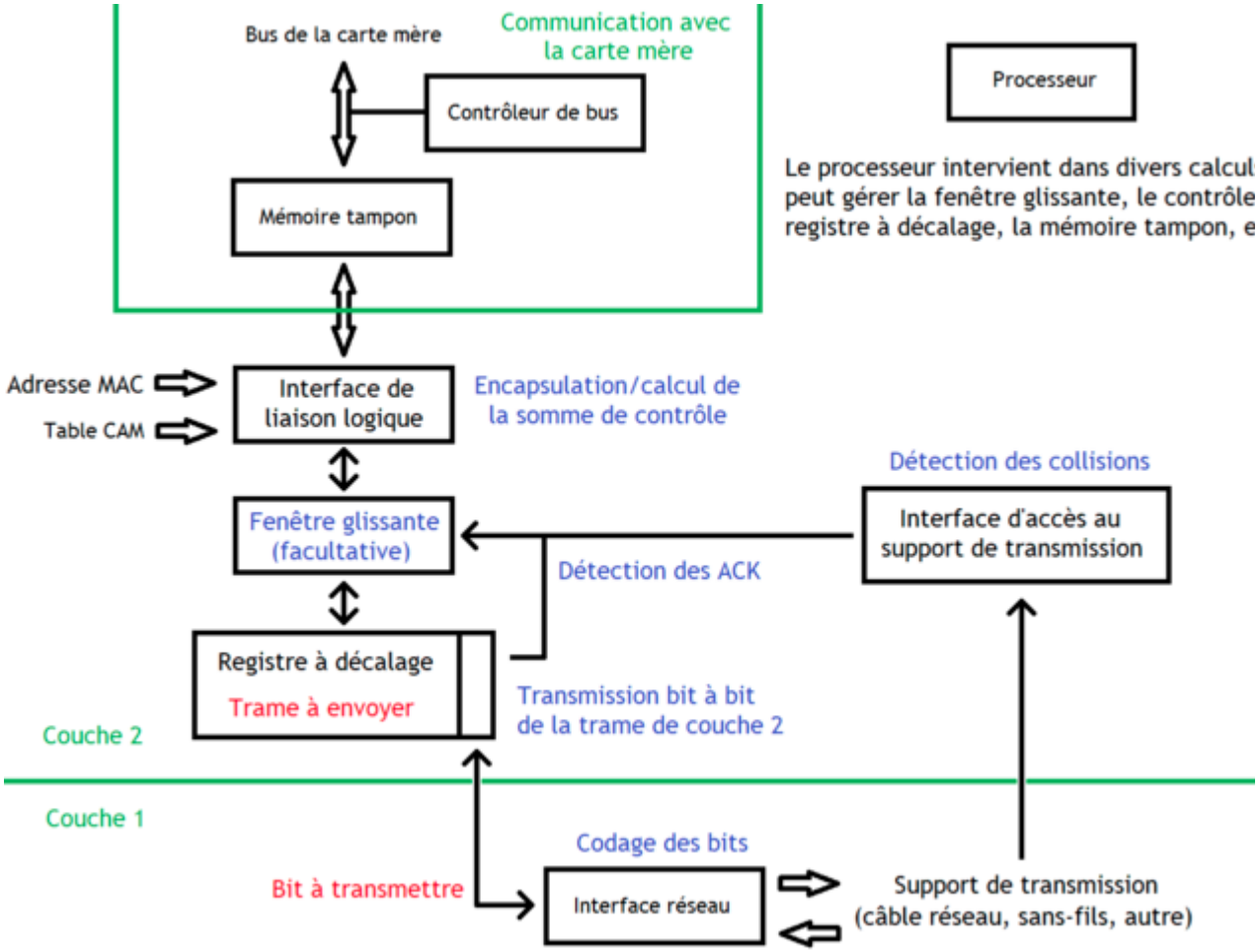
CMSA-CD



CSMA-CD : détection d'une collision.

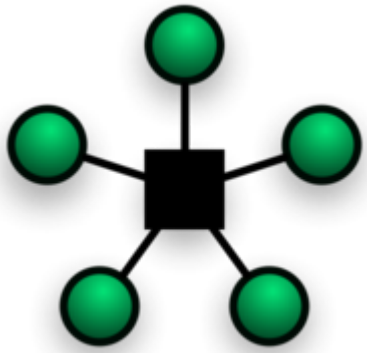
La carte réseau

Tous les traitements de la couche liaison sont, avec les traitements de la couche physique, 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.

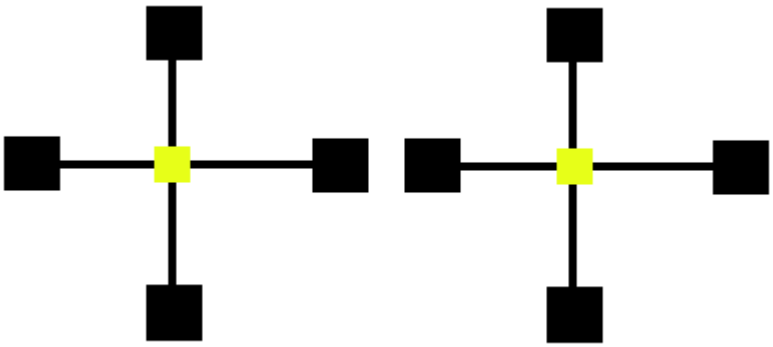
Les topologies logiques



Topologie en étoile.

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 surtout sur les liaisons en bus, et autres réseaux de diffusion. Il faut dire que les traitements de la couche liaison sont assez simples pour des liaisons point à point, une bonne partie de la gestion des trames et de l'accès au support étant alors fortement simplifié. Néanmoins, une bonne partie des réseaux locaux ne sont pas des réseaux en bus, mais 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.

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 laquelle on vient connecter des composants réseau : 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 : c'est l'équivalent réseau d'un bus. Pour le dire autrement, un hub simule une topologie en bus, alors que la topologie réelle est en étoile. Cela nous pousse à faire la distinction entre topologie physique, au niveau du matériel installée, et topologie logique, simulée par les concentrateurs et commutateurs. 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).

La topologie maillée et la commutation de trames

Dans cette section, nous allons voir comment il est possible de simuler un réseau maillé à partir d'un réseau local en étoile. Cela nous amènera à étudier en profondeur le fonctionnement des commutateurs, ainsi que les protocoles qu'ils utilisent. Pour introduire ce que nous allons voir, nous devons préciser ce que le commutateur doit faire pour simuler un réseau maillé. Pour faire simple, son rôle est de rediriger les trames reçues vers leur destinataire. Il reçoit des trames sur ses ports d'entrée et doit les renvoyer sur le port de sortie qui correspond à la destination. Il existe diverses méthodes pour faire cela, qui consistent toutes à mémoriser quel destinataire est connecté à tel ou tel port. Tout le problème est de savoir quel est le destinataire d'une trame reçue. Et aussi bizarre que cela puisse paraitre, il y a plusieurs méthodes différentes pour cela. A l'heure actuelle, deux méthodes sont devenues prédominantes : la technique des datagrammes et la commutation de circuits virtuels. Avec les datagrammes, chaque trame contient toutes les informations nécessaires pour identifier le destinataire. Le principal avantage de cette méthode est qu'elle ne nécessite pas de maintenir une connexion entre source et destinataire : une trame est envoyée sans concertation préalable avec le destinataire. Tel n'est pas le cas de la commutation de paquets virtuels, qui nécessite de maintenir une connexion active entre source et destinataire.

La commutation par datagrammes

Avec la méthode des **datagrammes**, c'est la trame qui précise son destinataire dans son en-tête de couche liaison. Comme vous l'avez sans doute deviné, elle contient l'adresse MAC du destinataire pour cela. 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**. Dans ce qui va suivre, nous allons nous concentrer sur la table CAM, bien qu'il y aie d'autres composants dans un commutateur. 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.

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. 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. Cette durée de péremption est appelée le **Time To Live**, ou TTL, et vaut entre 0 et 255 secondes.

La topologie en bus et les concentrateurs

L'usage des concentrateurs est moindre que celui des commutateurs. Néanmoins, ils sont suffisamment utilisés pour que nous devions en parler. De plus, parler des concentrateurs nous permet de parler plus en détail de deux concepts importants : les domaines de diffusion et de collision.

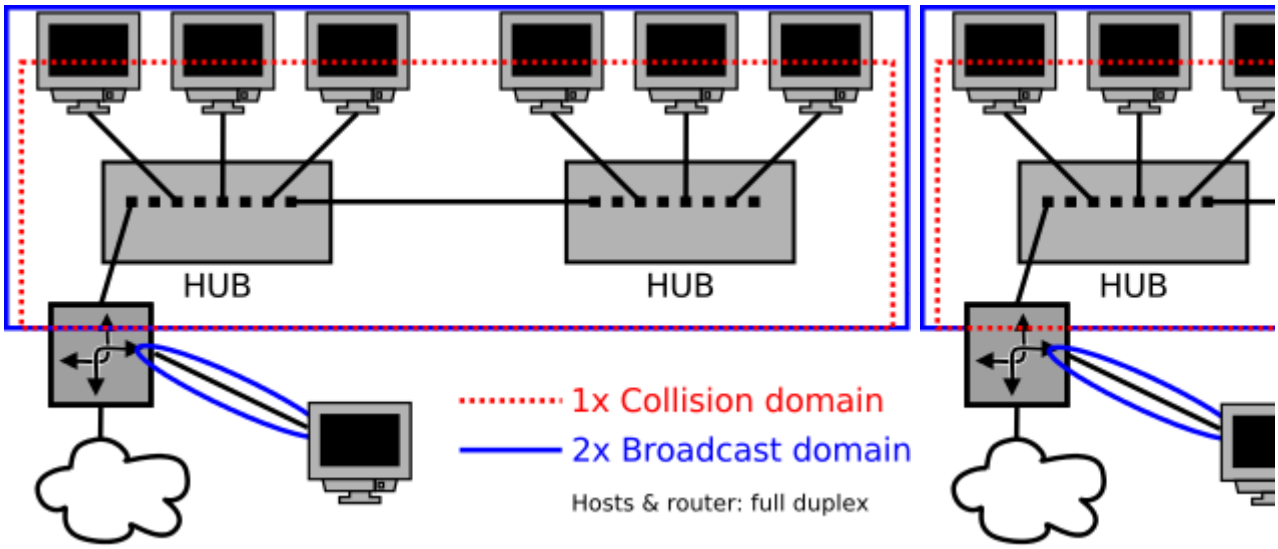
Le domaine de diffusion

Le **domaine de diffusion** est l'ensemble des ordinateurs qui peut communiquer avec des transmissions dites *broadcast*. Celles-ci sont des transmission spéciales, qui permettent à un ordinateur d'envoyer un paquet à tous les autres ordinateurs du réseau local. Cela arrive sur tous les réseaux de diffusion : chaque paquet est envoyé à tous les autres ordinateurs : c'est donc une transmission en *broadcast*. Les concentrateurs, en simulant une topologie en bus, vont faire que toute transmission sera une transmission *broadcast*. Le domaine de diffusion est donc l'ensemble des ordinateurs reliés directement et/ou indirectement au concentrateur. Si seuls des ordinateurs sont reliés au concentrateur, alors le domaine de diffusion comprend ces ordinateurs uniquement. Les subtilités surviennent quand on connecte des concentrateurs entre eux : dans ce cas, tous les ordinateurs reliés aux différents concentrateurs (eux-même connectés) forment le domaine de diffusion.

Le domaine de collision

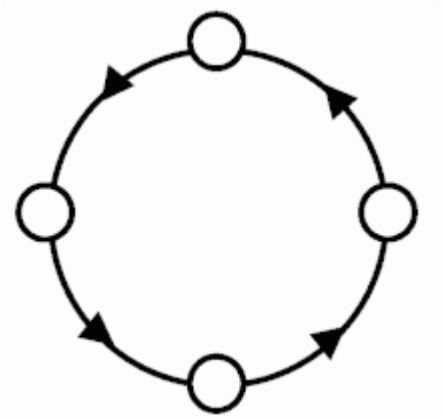
Le **domaine de collision** est un concept légèrement différent. Il partage cependant un point commun avec le domaine de diffusion : il comprend l'ensemble des ordinateurs connectés à un même bus logique et/ou physique. La différence n'est pas dans le fait qu'un ordinateur peut envoyer des données à tous les autres. Il tient dans le fait que deux ordinateur du même domaine de collision ne peuvent pas envoyer des données sur le bus en même temps : s'ils le font, cela entraine une collision. La différence est subtile, surtout que les deux sont confondus avec l'usage de concentrateurs. Mais on verra que les deux peuvent se séparer avec l'usage de commutateurs et/ou de routeurs. Par exemple, si on relie des ordinateurs avec un commutateur, ceux-ci ne font pas partie du même domaine de collision en raison du fonctionnement du commutateur. Pourtant, ils partagent le même domaine de diffusion : le commutateur peut parfaitement envoyer une trame à tous les ordinateurs d'un réseau. Mais les bus ne sont pas possible, vu que la topologie

simulée n'est pas un bus. Avec un commutateur, il y a un domaine de collision par port, contre un domaine de collision unique pour le concentrateur.



La topologie en anneau et le Token Ring

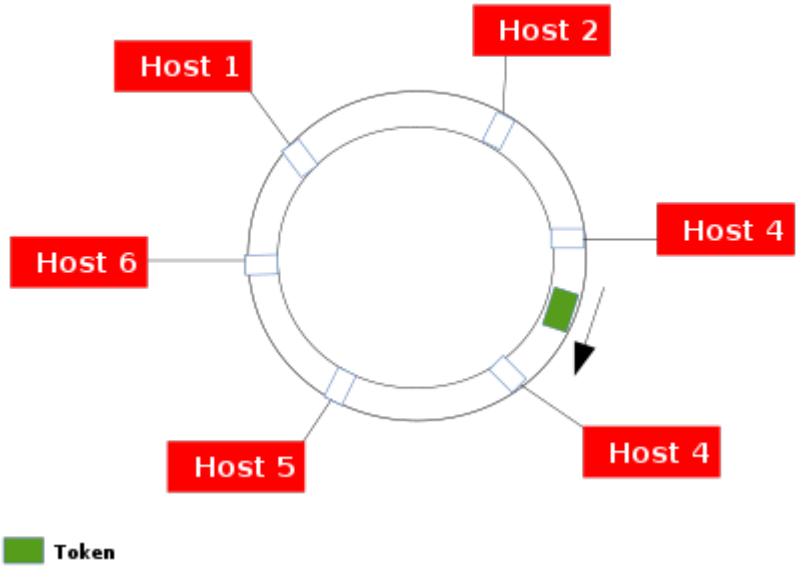
Avec la topologie linéaire, les ordinateurs sont reliés les uns à la suite des autres, en formant une chaine. Avec la topologie en anneaux, les ordinateurs forment un anneau : on peut dire que c'est une variation de la topologie linéaire, où on ferait boucler la chaine sur son début. Dans les réseaux en anneau, 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. Une fois arrivé, l'ordinateur de destination envoie un accusé de réception à l'émetteur. Une nouvelle émission est possible une fois l'accusé de réception reçu.



Topologie logique en anneau.

Méthode d'accès

Chaque ordinateur a accès à l'anneau a tour de rôle, un seul ordinateur ayant le droit d'envoyer des données sur l'anneau à la fois. Ce droit est modélisé par un **jeton**, qui passe d'un ordinateur au suivant à chaque émission. Ce jeton est juste une trame spéciale : toutes les trames envoyées sur le réseau ont un bit qui indique si cette trame est le jeton ou pas. Ce jeton est conservé dans un temps déterminé et fixe, le même pour tous les ordinateurs. L'utilisation d'un jeton permet à tous les ordinateurs d'avoir un accès équitable au réseau. On sait que tous les ordinateurs auront accès au réseau à un moment ou un autre. Évidemment, ils peuvent ne pas utiliser leur jeton s'ils n'ont rien à envoyer, et transmettent alors le jeton à l'ordinateur suivant. De plus, le temps avant de récupérer le jeton est obligatoirement fini et on peut lui attribuer une limite maximale : le temps que le jeton fasse le tour de l'anneau. C'est un avantage que les bus n'ont pas : un ordinateur peut, en théorie, monopoliser le bus durant un temps indéterminé si rien n'est prévu dans la conception du bus.

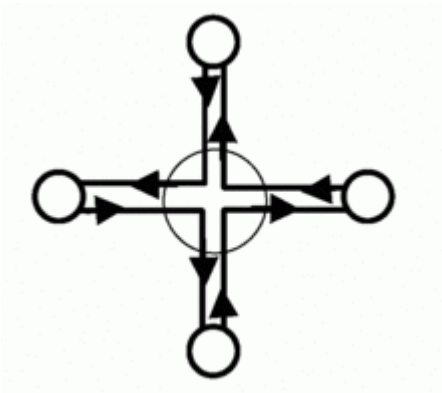


Propagation du jeton (Token) dans un réseau en anneau.

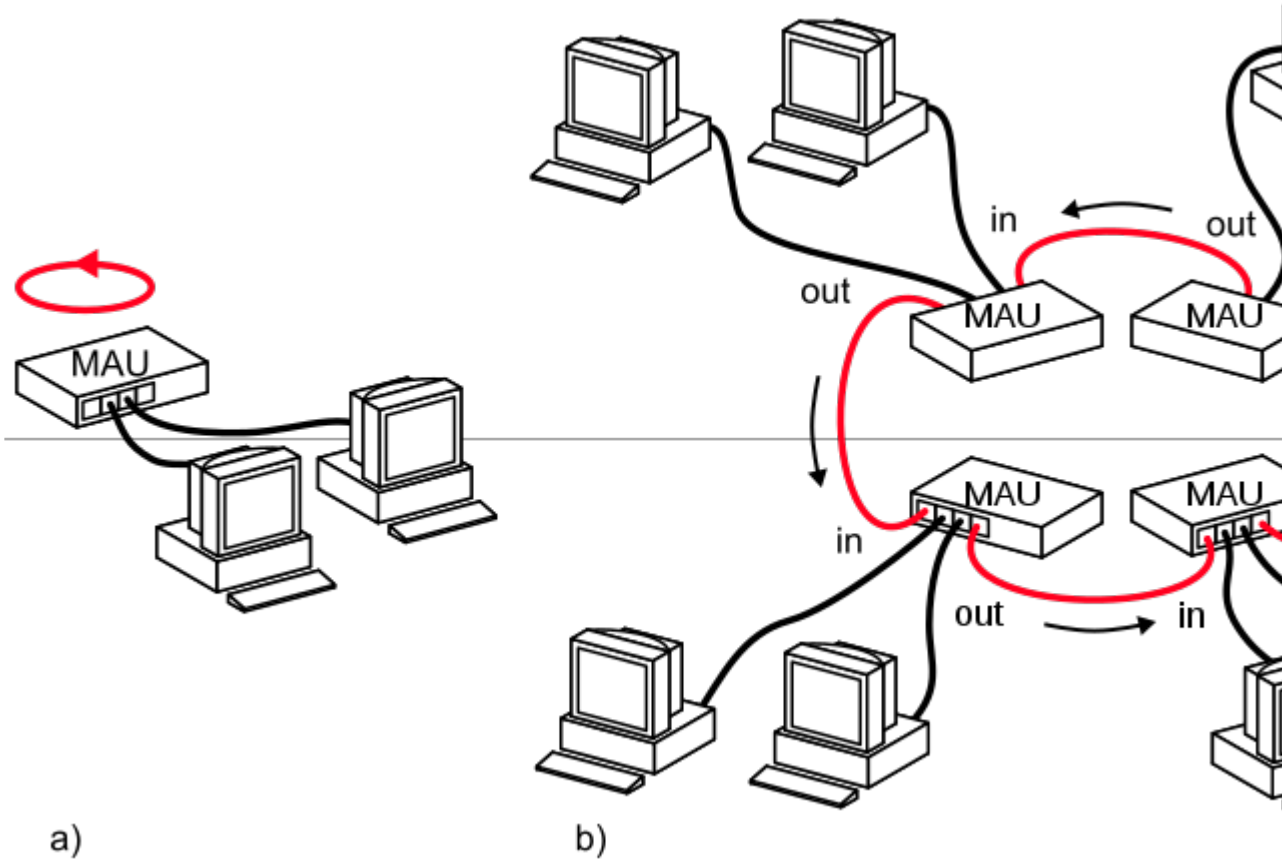
Si peu d'ordinateurs accèdent au réseau, les réseaux en bus ou étoile ont une meilleure efficacité. Mais à forte charge, les réseaux en anneau ont une efficacité nettement plus importante. En effet, les collisions sont impossibles avec un réseau en anneau, grâce à ce système de jeton. En comparaison, les collisions limitent la performance des réseaux en bus. Cela se fait sentir surtout quand on branche un grand nombre de machines sur le bus : le nombre de collisions augmente, ce qui réduit le débit utilisé pour transmettre des données. En clair, le débit utilisé est plus important sur un réseau en anneau, vu qu'il n'y a pas de collisions.

Multistation Access Unit

Encore une fois, on peut 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**. On connecte plusieurs ordinateurs sur ce boitier, 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.

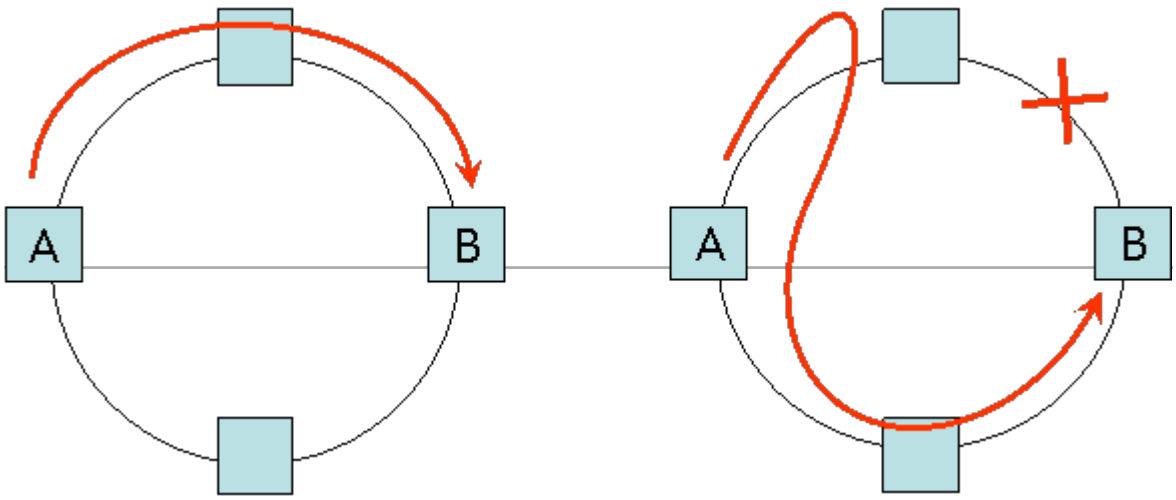


Token ring avec un éuipement central (MAU).



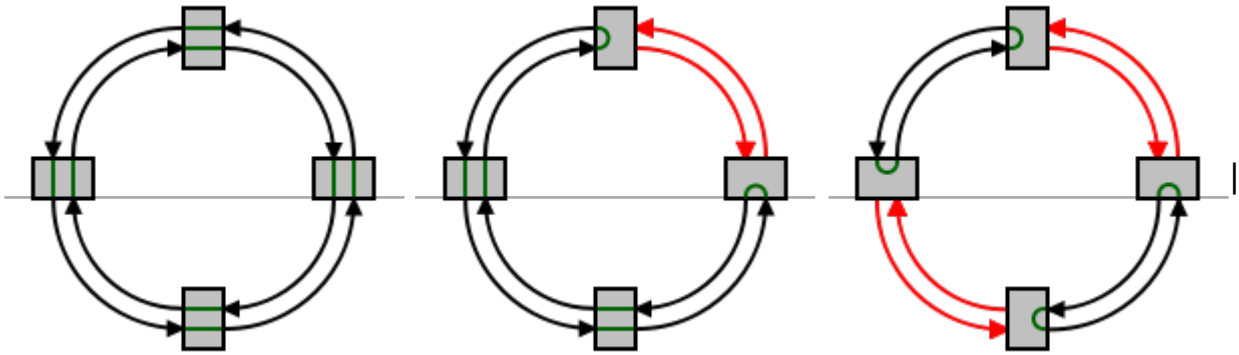
Les anneaux auto-réparants

Avec les réseaux en anneau précédent, la moindre panne d'un ordinateur peut mettre 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.



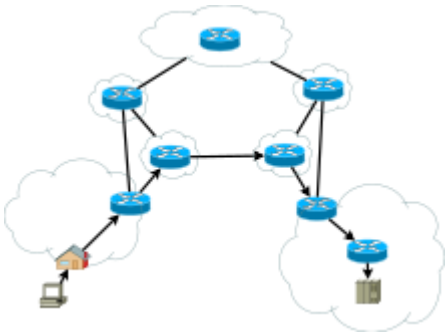
Les concepteurs de réseaux ont depuis longtemps trouvé comment propager les données dans les deux sens. Il suffit d'ajouter un second anneau qui propage les trames dans l'autre sens. Le réseau en anneau est donc composé de deux anneaux, qui propagent les jetons et trames dans des sens différents. Quand 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.



Le routage

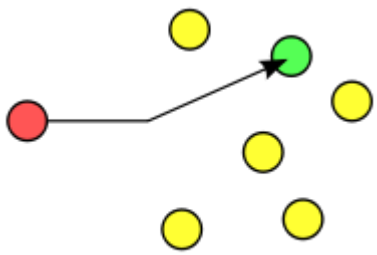
Il est maintenant temps de laisser les réseaux locaux derrière nous, et de voir plus grand. Par plus grand, je veux dire que nous allons aborder les réseaux composés d'une interconnexion de réseaux locaux plus simples. Internet est l'un d'entre eux, même si ce n'est pas le seul. Quoiqu'il en soit, ces interconnexions de réseaux ne fonctionnent que si divers mécanismes permettent à ces réseaux de communiquer.



Routage.

Dans un réseau partiellement maillé, tel Internet, la donnée est propagée de proche en proche, d'intermédiaire en intermédiaire : on appelle ces intermédiaires des **routeurs**. Cette propagation doit cependant être gérée, histoire que la donnée arrive bien à destination. Ces mécanismes demandent d'identifier les machines émettrices et réceptrices. Pour cela, l'émetteur et le récepteur se voient attribuer un numéro, l'adresse logique. On verra dans le chapitre suivant que cette adresse et une adresse dite IP, du nom du protocole IP qui standardise ces adresses et leur utilisation. Une fois les deux machines identifiées, il reste encore à propager les segments/paquets de données de proche en proche, de l'adresse IP source vers l'IP destinatrice. Déterminer quel est le chemin que doit parcourir la donnée pour arriver la destination est ce qu'on appelle le **routage**. Divers protocoles s'occupent de faire fonctionner ce routage, et les plus connus sont clairement BGP et IGP. Nous les verrons probablement dans la suite du cours, ce chapitre se concentrant surtout sur les mécanismes cachés derrière le routage.

Les différents modes de routage



Unicast

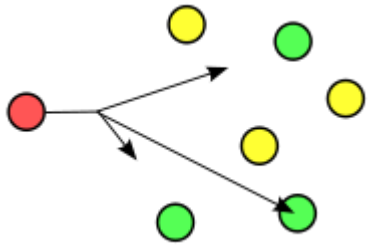
Le routage peut prendre différentes formes suivant le nombre de destinataires. Un ordinateur peut en effet vouloir communiquer avec un ordinateur bien précis, ou envoyer une donnée à plusieurs PC différents. Suivant le nombre de destinataire, on peut faire la différence entre Unicast, Anycast, Multicast 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 connait pas : l'émetteur ne connait pas la destination de la donnée. L'ordinateur de destination n'est cependant pas choisit au hasard : c'est le protocole de routage qui choisit vers quel ordinateur émettre la donnée.

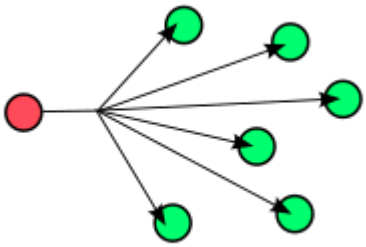
Avec le **multicast**, les données émises sont envoyées à un groupe d'ordinateur qui veulent recevoir cette donnée. Les ordinateurs qui veulent revoir la donnée se connectent à un serveur et s'inscrivent à un groupe de diffusion. Tous les ordinateur inscrits dans ce groupe recevront la donnée émise. 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 connait. 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.



Anycast

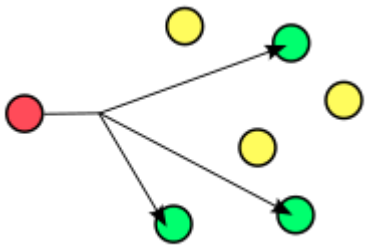
Avec le **broadcast**, un ordinateur émet des données à destination de tous les ordinateurs d'un réseau ou sous-réseau (un réseau local le plus souvent).



Broadcast

Les routeurs

Le routage est pris en charge par les routeurs qui propagent la donnée : chaque 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.



Multicast

La table de routage

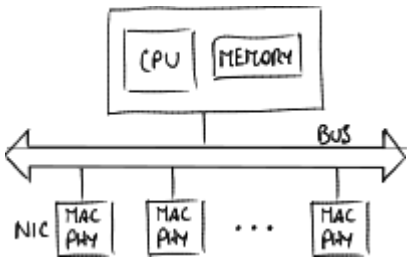
Un **routeur** peut être vu comme l'équivalent d'un commutateur, mais pour une interconnexion de réseaux : là où le commutateur connecte des machines dans un même réseau local, le routeur connecte et sert d'interface à deux réseaux différents. Il reçoit des paquets sur certains ports, et les renvoie 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. 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 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.

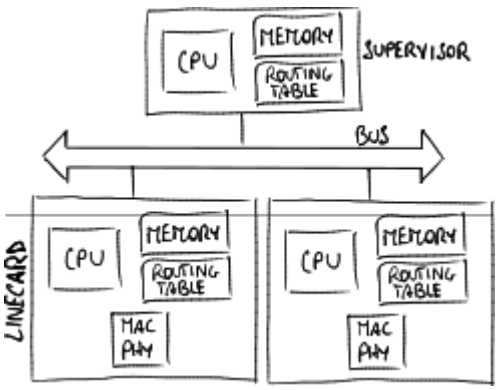
Cette trame doit être envoyée à l'ordinateur de destination, et donc envoyée sur un des ports de sortie du routeur, celui qui mènera ultimement la trame à destination. Son fonctionnement est similaire à celui d'un commutateur amélioré. Dans les grandes lignes, la table CAM est remplacée par une **table 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.

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.

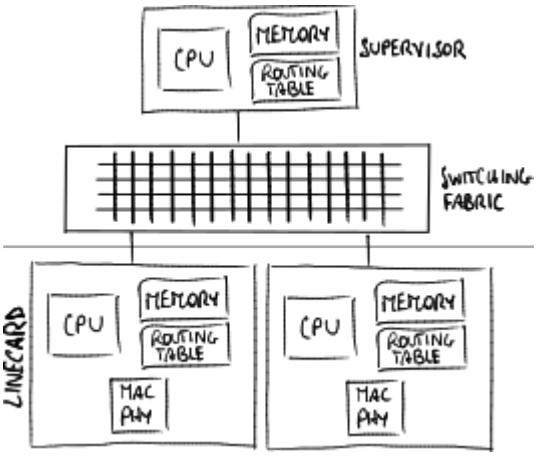


Les routeurs de seconde génération sont plus complexes. Ceux-ci multiplient le processeur, la RAM et la table de routage en plusieurs exemplaires : un exemplaire par port. Ainsi, les trames reçues sur un port sont directement traités dans les circuits de gestion de ce port. Une fois traitée, elles sont 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.



Une opération distribuée

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. 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 tellement ils sont complexes.

La mise à jour des tables routage peut permettre de trouver des chemins plus courts ou plus rapides pour acheminer une donnée à une IP précise. Les algorithmes de routage peuvent en effet être conçus pour tenir compte des performances des différents chemins entre deux routeurs. Ils peuvent aussi 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. Cette mise à jour peut être gouvernée par un routeur central, qui communique aux autres routeurs les informations de mise à jour : on parle alors de routage centralisé. Mais c'est assez inefficace sur la plupart des réseaux de grande taille, dont internet, où un tel mode de fonctionnement est inadapté. Dans ce cas, le routage est un routage décentralisé : chaque routeur met à jour sa table de routage individuellement, sans intervention d'un routeur central.

Le réseau Internet : le protocole IPv4

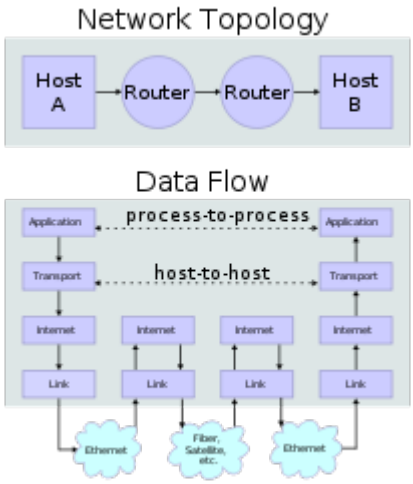


Illustration de la couche Internet.

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, aussi appelée **adresse IP**.

L'adresse logique 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. 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).

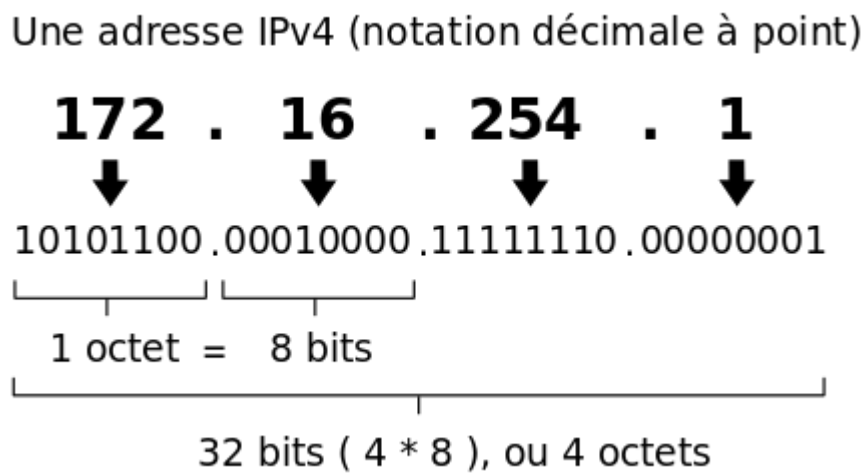
Parmi les adresses d'un réseau, deux sont spéciales :

- 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.

Actuellement, deux versions d'IP sont en circulation IPv4 et IPv6. Nous allons voir IPv4 dans ce chapitre et laisser IPv6 au chapitre suivant.

Adresses IPv4

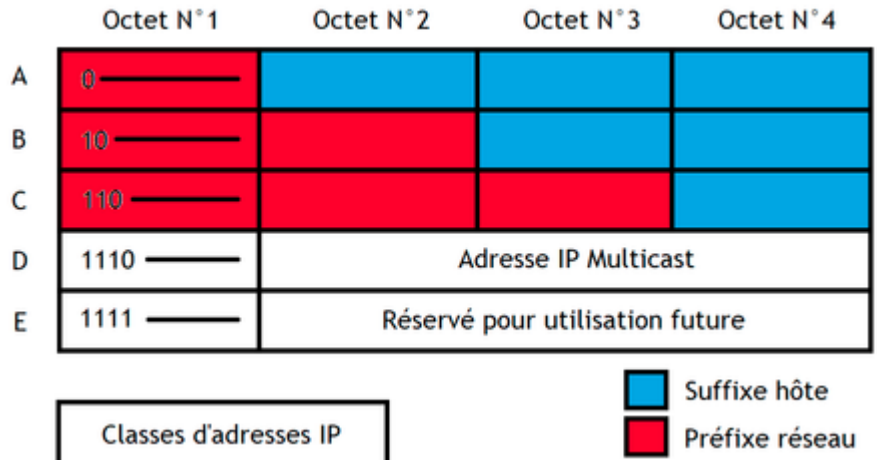
En IPv4, les adresses font exactement 32 bits (4 octets) : de quoi coder 4 294 967 296 adresses IP différentes. La notation des adresses IP est normalisée : on doit noter chaque octet en décimal et les séparer par des points.



Addresse Ipv4

Adressage par classes

Autrefois, les concepteurs du protocole IP ont inventé un mécanisme pour décomposer une adresse en réseaux indépendants : les **classes d'adresses IP**.



Classes d'adresses IPv4

Chaque classe définissait en réalité un préfixe réseau, ce qui fait que chaque classe est un ensemble d'adresses IP. La taille de cet ensemble était assez important, les intervalles et nombres d'IP étant indiqués ci-dessous.

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	

Mais 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. Le problème est que la taille du réseau doit être choisie dans quelques classes de taille déterminée : on ne peut pas choisir un intervalle d'adresses le plus proche possible du nombre de machine utilisée. Ce problème 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 technique est apparue en novembre 1992, dans une réunion de l'IESG, et l'une d'entre elle est encore en vigueur au moment où j'écris ces lignes (fin 2015).

Adressage par sous-réseaux

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.

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.

Validité du masque

Un masque valide doit respecter deux contraintes : les bits à 1 sont tous contiguës et regroupés dans les bits de gauche. Ainsi, les identifiants suivants sont invalides :

- 00011111 11111111 11111111 00000000 ;
- 00001111 11110000 00101001 00101010 ;
- 01111111 11111111 11111111 11111111.
- 10011111 11111111 11111111 00000000 ;
- 11001111 11110000 00101001 00101010 ;
- 11111111 11111111 11100011 11111111 ;
- 10011111 11111111 11111111 11111111 ;
- 11001111 11110000 00101001 00101010 ;
- 11111111 11111111 11111111 11111101.

Par contre, les masques suivants sont valides :

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

On peut facilement déduire des contraintes sur le masque la chose suivante :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).

Notation des masques

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.

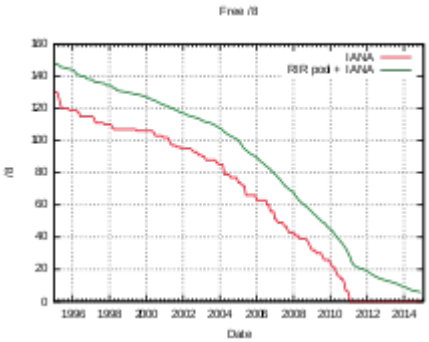
De même pour :

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

Network Address Translation

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 attribuées à un équipement réseau 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 :

Adresse de base et masque	Intervalle d'adresses
10.0.0.0/8	10.0.0.0 – 10.255.255.255
172.16.0.0/1	172.16.0.0 – 172.31.255.255
192.168.0.0/16	192.168.0.0 – 192.168.255.255



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

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.

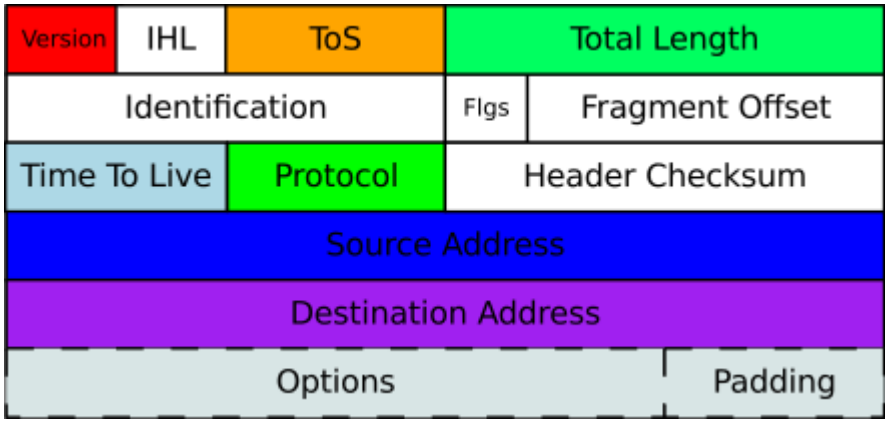
Adresses spéciales

Certaines adresses IPv4 sont cependant spéciales, à savoir qu'elles sont réservées pour des utilisations particulières. Les adresses des réseaux privés en font partie, bien évidemment, mais elles sont loin d'être les seules. Si l'on omet les adresses de réseaux privés, 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

En-tête IPv4

L'en-tête d'IPv4 est celui-ci :



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 quatres bits suivants indiquent la **longueur de l'en-tête**. Cette longueur n'est pas exprimée en bits ou en octets, mais en paquets de 32 bits (4 octets). L'en-tête a toujours une taille comprise entre 20 et 40 octets, par construction. En conséquence, la longueur de l'en-tête est comprise entre 5 et 15.

Le champ **type de service** servent pour la qualité de service.

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 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. 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.

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 du paquet est composé de bits de détection/correction d'erreurs, ainsi que des adresses sources et

destination.

L'obtention d'une IPv4 : DHCP et ARP

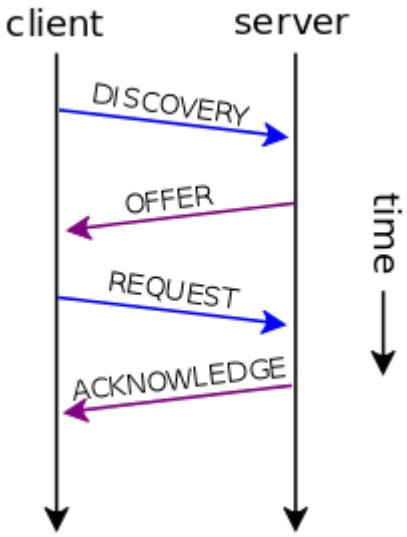
Le protocole IP ne peut pas faire fonctionner internet sur ses seules épaules. Il est secondé par tout un ensemble de protocoles annexes, qui complètent IP. Ces protocoles sont les protocoles DHCP, ARP, ICMP, et quelques autres. Pour donner un exemple, ICMP permet de détecter les pannes ou configurer un réseau à distance. Nous avons choisis de vous présenter les protocoles DHCP et ARP, qui nous semblent de loin les plus importants. Ceux-ci permettent de faire le lien entre une adresse physique (MAC) et une adresse logique (IP). Le premier permet d'obtenir une adresse IP à partir d'une adresse MAC, le second permet de faire l'inverse, à savoir trouver l'adresse MAC qui correspond à une adresse IP.

Obtenir une adresse IP : 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.			

Faire le lien entre IP et adresse MAC : le protocole ARP

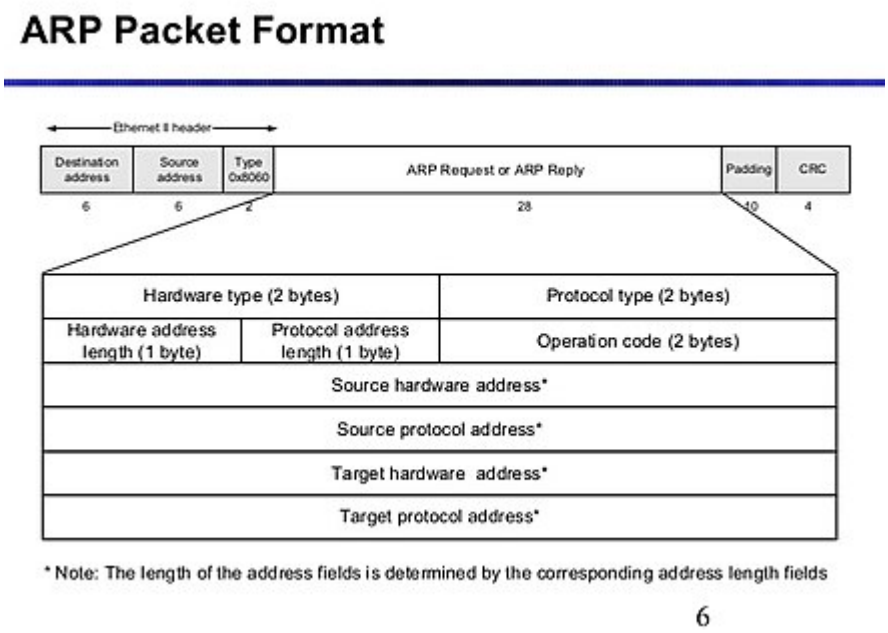
Comment savoir à quelle adresse MAC correspond telle adresse IP ? La solution à ce problème est très simple : il suffit de mémoriser les correspondances entre une IP et une adresse MAC dans une mémoire intégrée au routeur ou au matériel réseau. Cette table de correspondance est appelée la table MAC ou la table ARP suivant le protocole utilisé. Mais comment le routeur fait pour remplir et mettre à jour cette table ? Pour cela, il doit utiliser deux protocoles relativement simples : le **protocole ARP** pour les adresses IPV4, ou le **Neighbor Discovery Protocol** pour les adresses IPV6.

Fonctionnement

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 sa table ARP). Dans ce cas, 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. Une fois que le destinataire de l'IP a reçu le paquet, celui-ci répondra 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.

Segment ARP

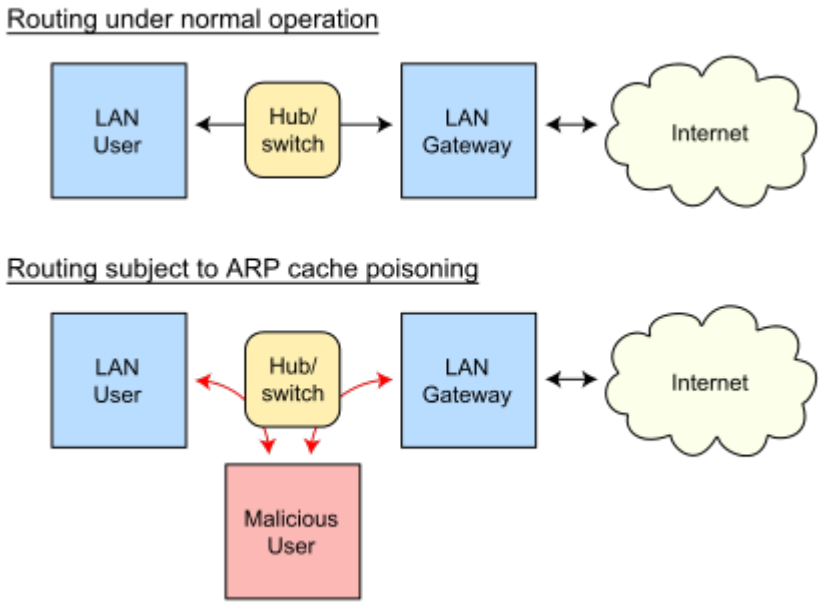
Tout paquet ARP est structuré comme indiqué dans le schéma ci-dessous. Il contient plusieurs champs, qui font entre un et huit octets. Le premier champ, nommé **Hardware Type**, indique quel est le protocole de couche liaison utilisé. Sa valeur la plus utilisée est 01, qui indique l'usage du protocole Ethernet. Les deux octets suivants forment le champ **Protocol Type**, qui indique le protocole de couche 3 utilisé. Le champ suivant, **Hardware Adress Length**, donne le nombre d'octets des adresses physiques (6 pour les adresses MAC). Le champ **Protocol Adress Lenght** donne lui 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. A la suite, on trouve les adresses physiques et logiques source et destination.



Segment ARP.

ARP Spoofing

Cependant, ce protocole a une grosse faille : 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.



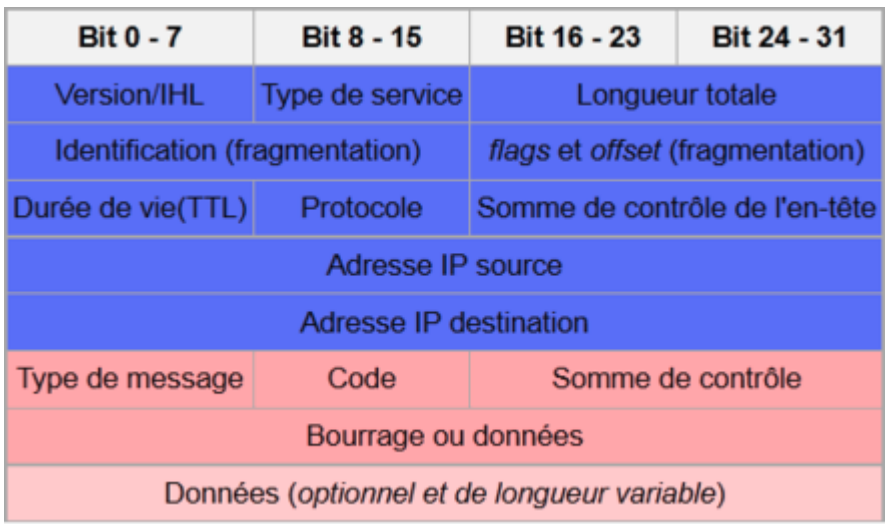
ARP Spoofing.

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 subit 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.

Format des paquet 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.



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 aie une taille fixe).

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.

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.

Autres 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.

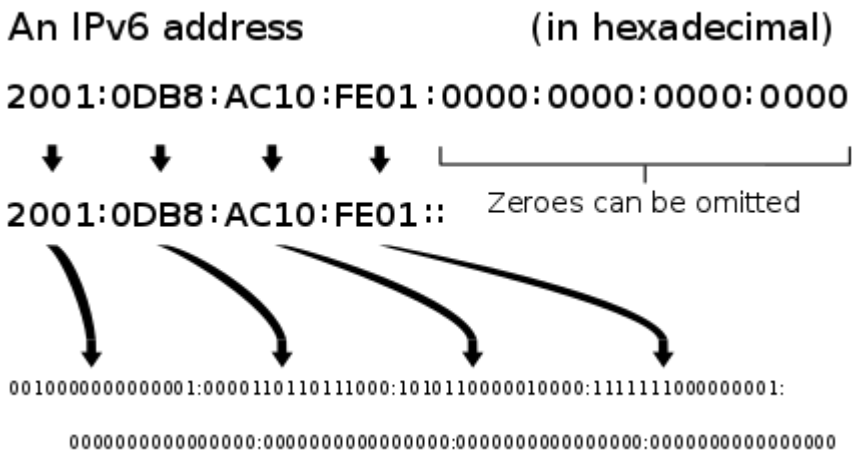
Les autres TYPE servent pour des échanges d'informations entre routeurs. Les TYPE 15 et 16 sont aujourd'hui obsolètes. Ils servaient au démarrage d'une machine, pour lui attribuer une adresse IP, chose devenue inutile avec les protocoles DHCP et RARP. Les TYPE 7 et 18 servent à obtenir le masque de sous-réseau 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.

Le réseau Internet : le protocole IPv6

La nouvelle version du protocole IP, l'IPv6, résout pour un bon moment la pénurie d'adresses IP.

Adresses IPv6

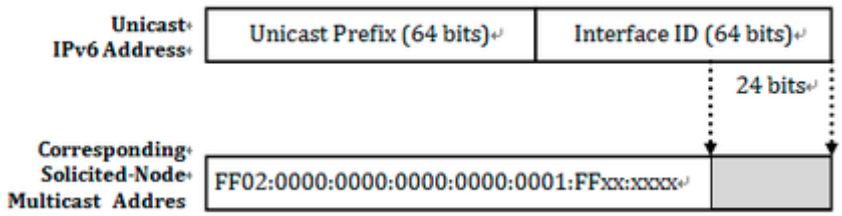
En IPv6, les adresses font quatre fois plus que les adresses IPv4 : exactement 128 bits, à savoir 16 octets. Plus de 256 milliards de milliards de milliards de milliards d'adresses IP différentes. Autant dire qu'on a le temps de voir venir la prochaine pénurie ! Pour noter une adresse IPv6, on n'utilise pas des nombres écrits en décimal : les octets sont notés en hexadécimal, pour économiser de la place. De plus, la séparation utilise le symbole : pour séparer des groupes de deux octets.



Ipv6 address

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.

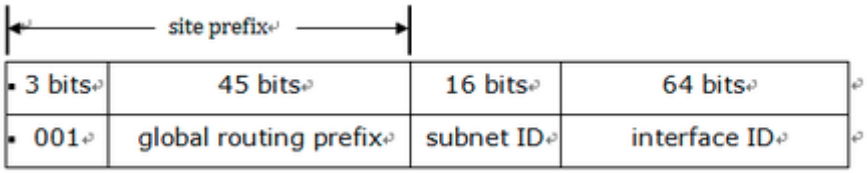


Structure d'une adresse IPv6 unicast.

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.



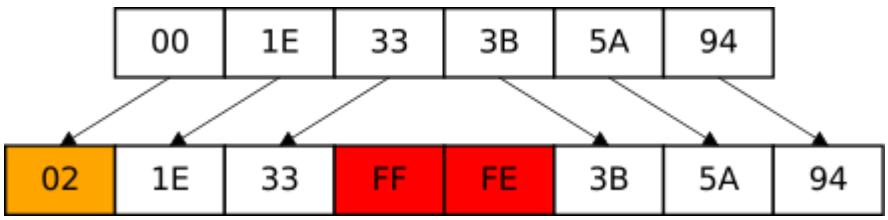
Structure d'une adresse IPv6 unicast.

Suffixe hôte

Le suffixe hôte est une portion d'adresse IP assez importante. A 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, entrainait 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.

EUI-64

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.

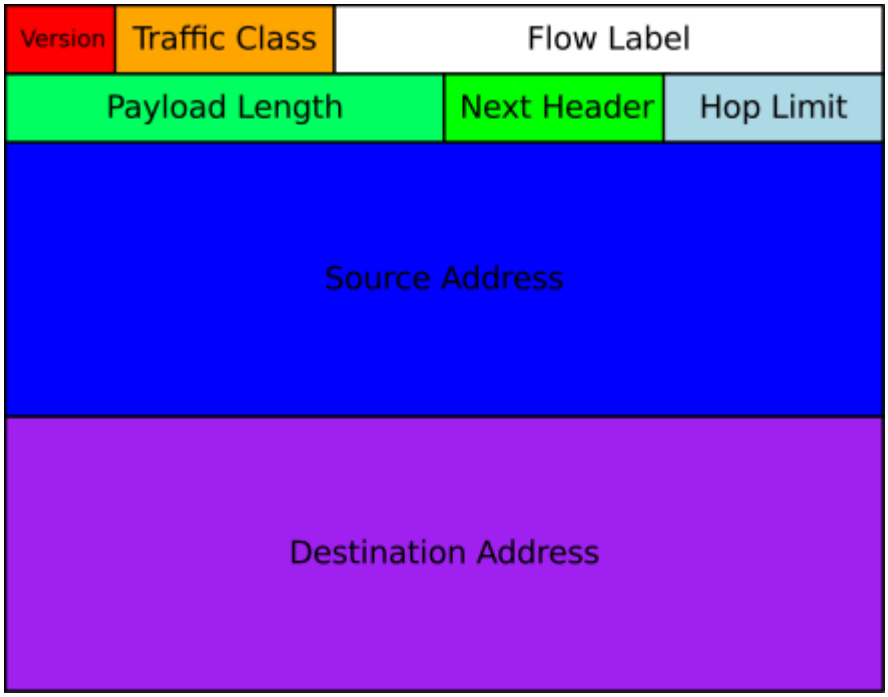


EUI 64

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.

En-tête IPv6

Voici l'en-tête utilisé par le protocole IPv6 :



En-tête IPv6.

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. 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, que voici :

- 00 – Réservé
- 01 – Non assigné
- 02 – Non assigné
- 03 – Non assigné
- 04 – IP V4
- 05 – ST Datagram Mode
- 06 – IP V6
- 07 – Non assigné
- 08 – Non assigné
- 09 – Non assigné
- 10 – Non assigné
- 11 – Non assigné
- 12 – Non assigné
- 13 – Non assigné
- 14 – Non assigné
- 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 transport des données

Quand deux ordinateurs s'échangent des données, ces données sont souvent des données de taille variable et non-bornée : dans le jargon du réseau, ces données sont appelées des **datagrammes**. 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 segmenter les datagrammes en paquets. C'est le rôle principal de la couche transport. De plus, quand un ordinateur reçoit un paquet, il doit savoir à quel programme est destiné ce paquet : est-il destiné au navigateur web, à un jeux 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. Les numéros de ports actuels font deux octets (16 bits), ce qui donne 65536 ports différents. Parmi ces ports, ceux dont le numéro est inférieur à 1024 sont réservés pour des services spécifiques alors que les autres sont attribués à la demande aux programmes. Voici une liste des ports les plus utilisés de nos jours :

- 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) used in the World Wide Web
- 110: Post Office Protocol (POP3)
- 143: Internet Message Access Protocol (IMAP)
- 443: HTTP Secure (HTTPS)

Protocole UDP

Le protocole UDP est le plus simple des deux, alors que TCP est celui qui a le plus de fonctionnalités. Pour faire simple, UDP se contente du minimum syndical qu'on attend d'un protocole de ce genre : il permet à deux programmes d'échanger des données sur le réseau, rien de plus. UDP ne vérifie pas que la donnée est bien arrivée à bon port, et ne vérifie pas non plus que les données ont été reçues dans l'ordre d'envoi : seul TCP en est capable. Il est donc très adapté dans les situations où on se moque que les données arrivent dans l'ordre et que les pertes de données sont acceptables. Typiquement, un stream 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 données à envoyer sur le réseau. Cet en-tête contient diverses informations, toutes codées sur deux octets. L'en-tête a une taille fixe, de 64 bits, soit 8 octets, quelque 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, et des octets de contrôle d'erreur optionnels. Le champ longueur code la longueur totale du paquet, en-tête compris. 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.

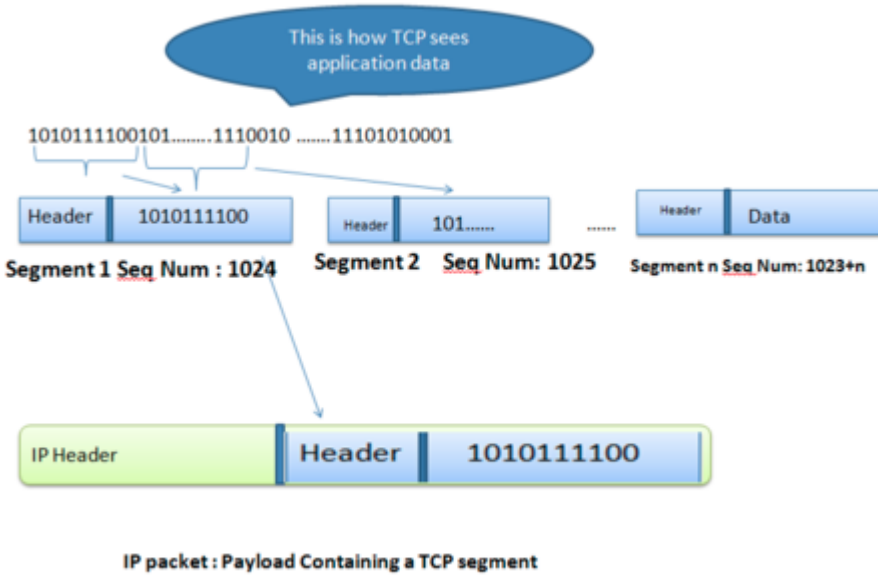
Port Source (16 bits)	Port Destination (16 bits)
Longueur (16 bits)	Somme de contrôle (16 bits)
Données (longueur variable)	

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 connections, 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.

Séquencement des paquets

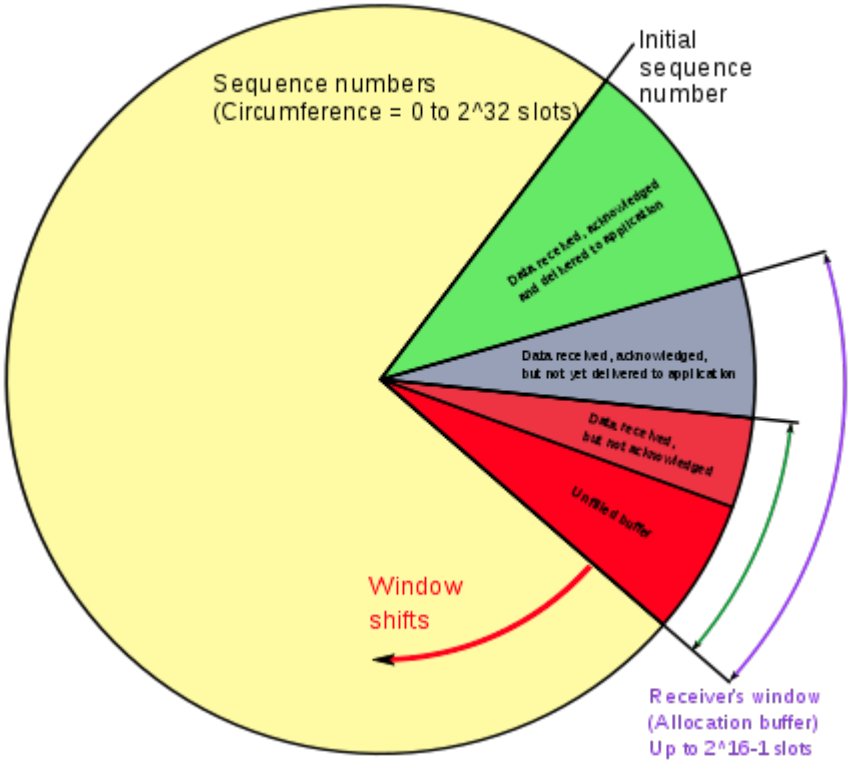
Lors de l'envoi d'une donnée sur le réseau, l'équipement réseau doit segmenter les datagrammes en paquets et les envoyer sur le réseau individuellement. Il faut savoir que les paquets envoyés ne sont pas forcément reçus dans l'ordre : TCP doit les remettre dans l'ordre. Le moyen le plus simple pour cela 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 coté des numéros de ports logiciels.



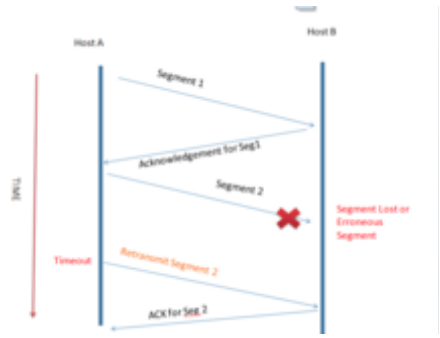
TCP segment.

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 disparue. 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.

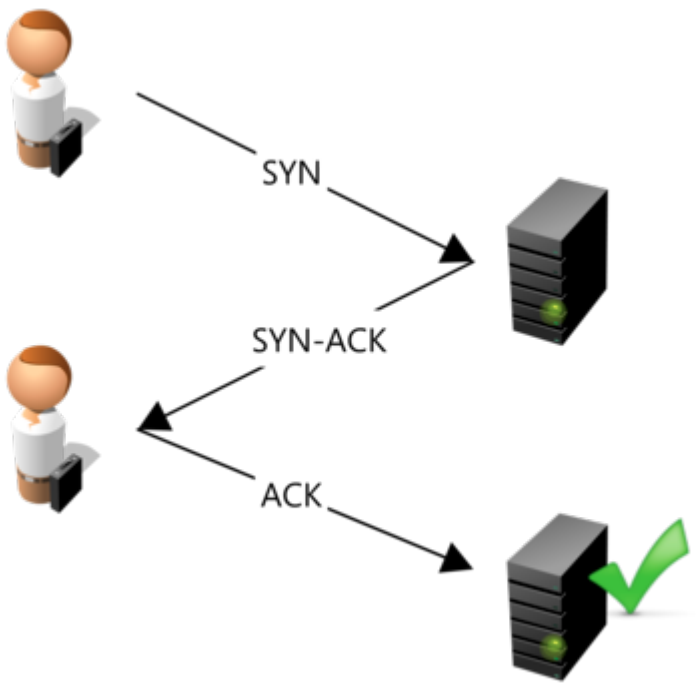
Avec les accusés de réception, on sait qu'une donnée a disparu si on n'a pas reçu d'accusé de réception

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 entrainera 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.

Connexion et déconnexion

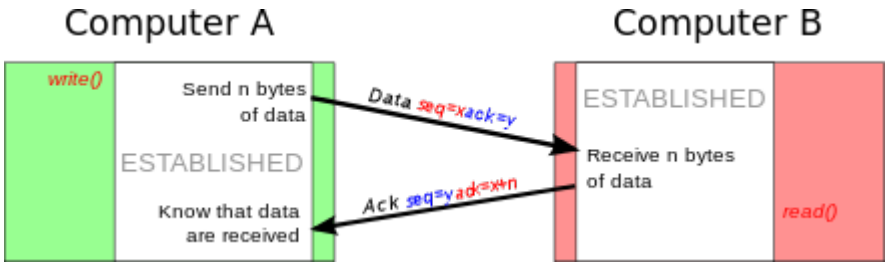
TCP est un protocole qui est dit en mode connecté. Ce qui signifie que tout transfert de donnée doit être précédé d'une négociation entre l'émetteur et le récepteur, cette négociation étant appelée une **connexion**. La connexion doit être ouverte pour que l'échange de donnée aie lieu et elle doit être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

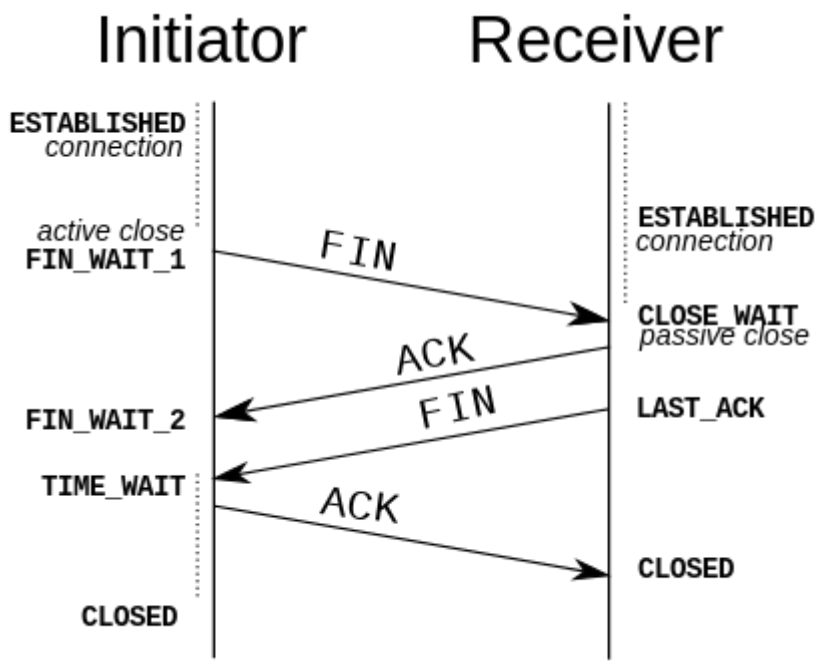
Lors de la phase d'envoi de données, l'émetteur envoie des paquets de type DATA, et le serveur répond par des accusés de réception.



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 coté. Chaque demande de déconnexion d'un ordinateur est autorisée par un accus de réception. Pour se déconnecter, ils doivent envoyer un paquet spécial nommé FIN. Dans les grandes lignes, voici comment a lieu la déconnexion :

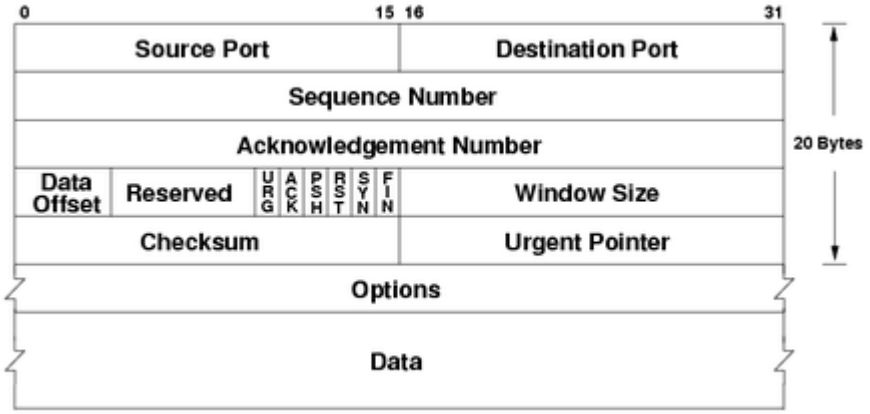
- un ordinateur envoie une demande de déconnexion ;
- l'autre ordinateur reçoit celle-ci et renvoie l'ACK qui va avec ;
- ce dernier envoie lui aussi une demande de déconnexion ;
- et son comparse reçoit celle-ci et renvoie l'ACK qui va avec.



Déconnexion TCP

Paquet TCP

Voici à quoi ressemble un segment TCP :



TCP Header

On voit que ce segment 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. 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. Le champ Windows 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.

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 de l'émetteur. 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

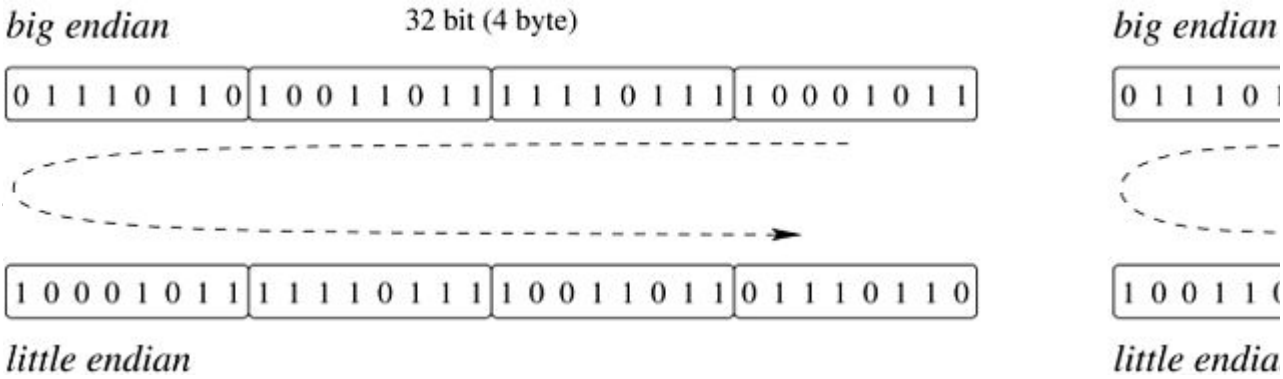
Le premier rôle de la couche présentation est de garantir la **compatibilité des données** entre les machines d'un réseau. Pour rappel, les données peuvent être codées de pleins de manières différentes : le texte peut utiliser des jeux de caractères très différents, les entiers peuvent être codés en 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. On pourrait croire que cela 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. Mais pour se simplifier la tâche, les transferts réseau utilisent souvent d'autres méthodes. Généralement, on utilise des formats intermédiaires standardisés : les données à envoyées 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 un 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

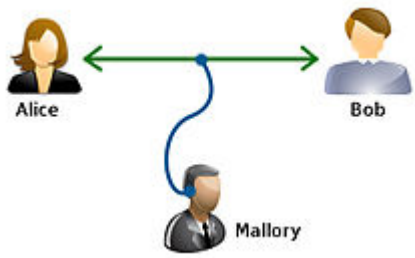
des transferts réseaux : la conversion du boutisme. Pour la comprendre, nous allons devoir faire un rappel sur ce qu'est le boutisme.

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 aie 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.



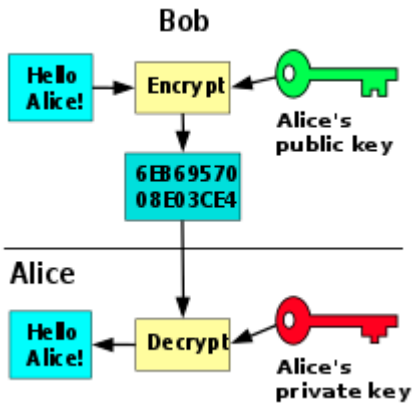
Attaque de l'homme du milieu.

Sans chiffrement, les données sont envoyés 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**.

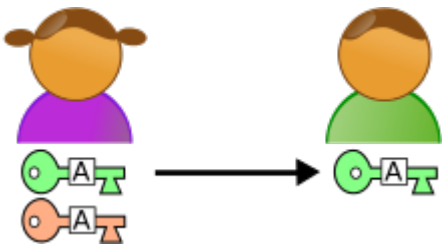
Pour éviter ce genre d'attaque, les informations échangées doivent être cryptées avant leur transfert. Pour rappel, le cryptage consiste à transformer les données à transmettre grâce à une fonction mathématique et/ou un algorithme. Cette transformation se fait en manipulant les données et une ou plusieurs 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

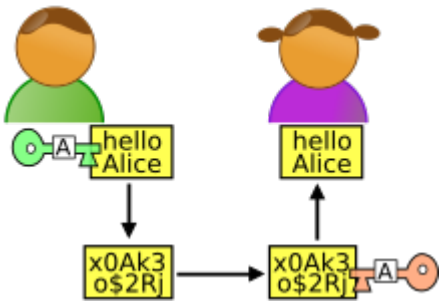
simples n'utilisent qu'une seule clé, qui sert à la fois pour crypter les données et les décrypter. On parle de chiffrement symétrique. Mais sur le net, les techniques simples ne peuvent pas être utilisées. En effet, cela demanderait de transmettre la clé au destinataire du message, clé qui peut alors être interceptée au même titre que le message.

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.

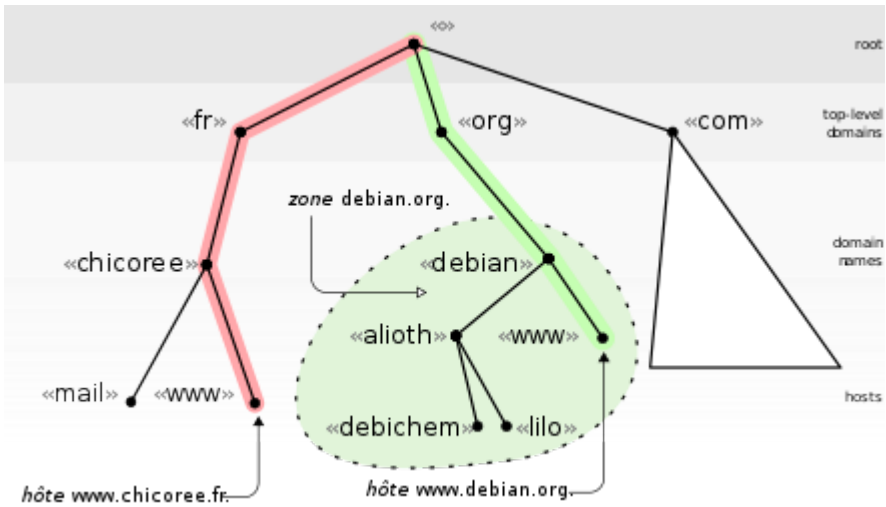
La couche application : le web et ses protocoles

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. 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. Les navigateurs web sont des applications qui récupèrent le contenu des pages, localisées sur les serveurs web, pour les consulter sur un autre ordinateur, ce dernier étant appelé un client web.

Les adresses web (URL)

Les adresses des sites web, aussi appelées adresses URL, ressemblent plus ou moins à ceci : http://www.example.com . Tout document, image, page web, a une **adresse URL**, qui est souvent utilisée pour créer des liens vers celui-ci. 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



Hiérarchie des noms de domaine.

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. Ce lien peut simplement être le fait qu'elles appartiennent à un même site. Toutes les pages d'un site web font généralement partie du même domaine. Dans le nom de domaine www.fr.wikipedia.org , c'est le cas du domaine wikipedia. Cela peut aussi être lié à la localisation géographique ou la langue du site : c'est par exemple le cas du domaine .fr ou .uk. Ces domaines ne sont pas gérés n'importe comment et diverses organisations gèrent certains domaines principaux, les **domaines de premier niveau**. Les domaines de premier niveau sont les fameux domaines .fr, .uk, .com, .org, et ainsi de suite. Certains sont des domaines géographiques (.fr, par exemple), tandis que d'autres sont liés à des organisations (.gov pour les gouvernements, .edu pour les établissements scolaires, .mil pour les organismes militaires, et quelques autres). Ces domaines de premier niveau sont gérés par l'ICANN (Internet Corporation for Assigned Names and Numbers), un organisme américain. 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.

Fichier HOST

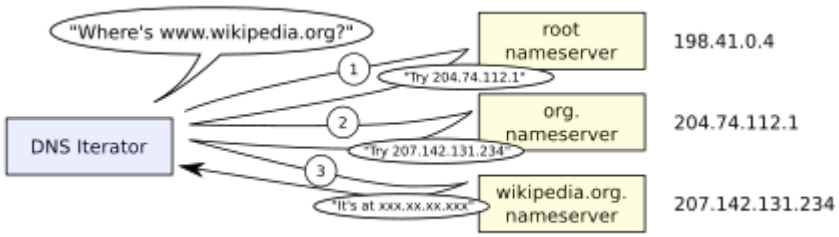
Au tout début d'internet, les correspondances entre IP et URL étaient mémorisées dans des fichiers HOSTS.TXT, qui existe toujours sur certains systèmes d'exploitation. Ce fichier était accessible sur un serveur dédié, maintenu par le Network Information Center. Mais cette méthode a rapidement montré 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 moderne et les navigateurs web peuvent l'utiliser comme bon leur semble. Modifier le fichiers 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.

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 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.

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**. Ce serveur racine est un serveur DNS qui ne connait pas la correspondance entre nom de domaine et IP, mais qui peut la rechercher sur d'autres serveurs. Le navigateur connait les adresses IP de ces serveurs DNS, qui sont des adresses fixes et réservées

uniquement pour les serveurs DNS. Le serveur racine va renvoyer l'adresse du serveur qui gère le domaine de premier niveau. Ces serveurs DNS de premier niveau renvoient l'adresse d'un serveur chargé du domaine de second niveau. Et ainsi de suite :



Example of an iterative DNS resolver

le processus se poursuit jusqu'à ce qu'on tombe sur l'IP recherchée. Prenons un serveur racine qui reçoit le nom de domaine zestedesavoir.com. Il ne connait pas l'IP qui correspond et il va d'abord interroger le serveur racine. Celui-ci va renvoyer l'adresse IP du serveur chargé des .com. Le navigateur va alors interroger ce serveur avec le nom de domaine zestedesavoir.com. Celui-ci connait l'IP associée, et il la renvoie directement.

Face à un nom de domaine, le serveur peut se retrouver face à deux situations :

- 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 va alors interroger le serveur dont il vient de récupérer l'IP. Le processus de recherche continue tant qu'on lui fournit une adresse de serveur DNS qui peut contenir l'IP voulue. La plupart de serveurs DNS mémorise les réponses aux demandes les plus courantes dans une portion de RAM : le cache DNS. Ainsi, les réponses aux requêtes les plus fréquentes peuvent être lues depuis ce cache, ce qui est plus rapide. Les résultats les plus fréquents sont effacés du cache après un certain temps, au cas où l'IP associée à un nom de domaine change.

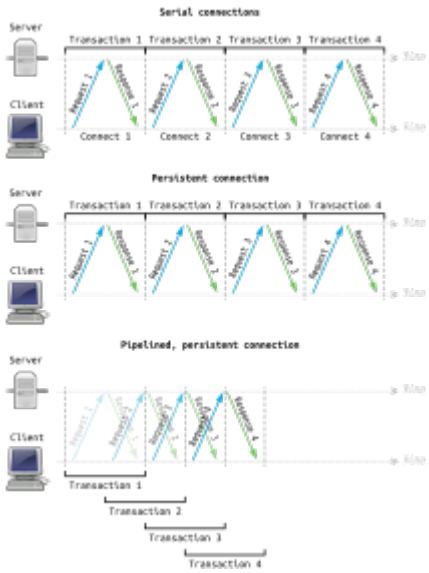
Le protocole HTTP

Une fois que le DNS a permit 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

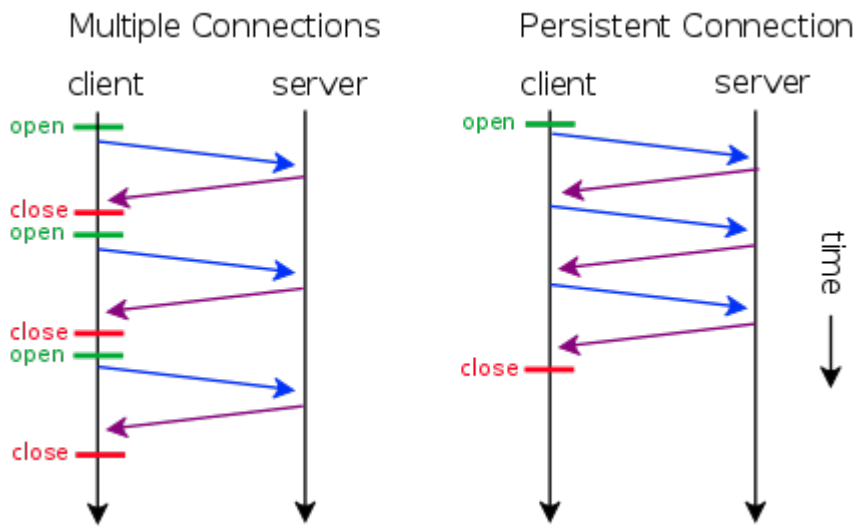
bleu, que vous avez saisi pour faire des achats en ligne, cela peut donner une belle catastrophe. Pour éviter cela, le HTTP utilise un système de chiffrement dit asymétrique, qui empêche toute attaque de ce genre. Ce système de chiffrement est basé sur le protocole **Transport Layer Security** aussi appelé TLS.

Les connexions HTTP



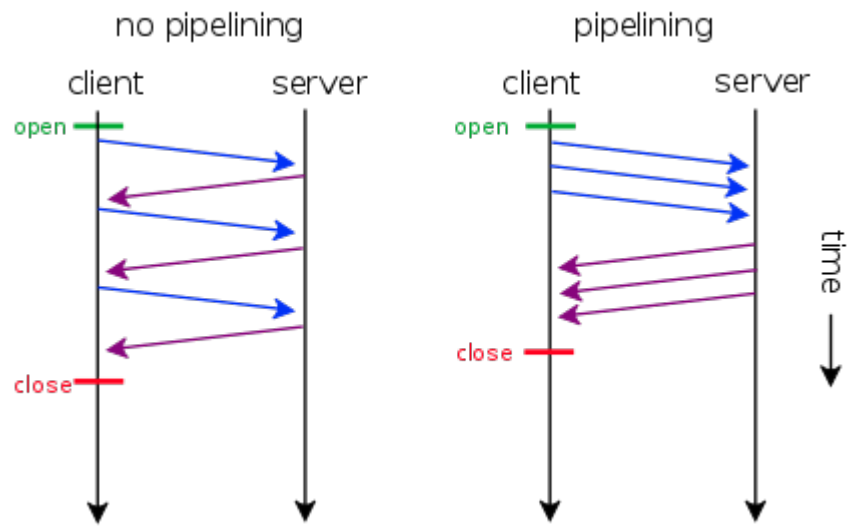
Connexions HTTP.

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, tout envoi de commande HTTP ouvrira une connexion, qui sera fermée lors de la réponse du serveur. 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. Le premier cas, chaque échange a sa propre connexion, donne ce qu'on appelle des **connexions non-persistantes**. Une connexion TCP est ouverte lors de l'envoi d'une commande, puis fermée une fois que le serveur a répondu à celle-ci. Le second cas, une connexion pour plusieurs échanges, 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 rafale.



Pipelining 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 sony 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 : commandé 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 un protocole dit sans état, à savoir que le serveur ne mémorise aucune information sur le client. Ainsi, 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 requête suivantes sous prétexte qu'elles ont déjà été servies. On dit que le protocole HTTP est un **protocole sans état**. Cette particularité permet de fortement simplifier le protocole, sans compter le gain en terme de performance. 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 et met à jour sur ordre du serveur. Lorsque vous consultez un serveur HTTP pour la première fois (lors d'une première connexion), celui-ci va renvoyer un cookie vierge avec la réponse. Lors des requêtes suivantes, le client enverra le contenu du cookie en même temps que ses requêtes. Dit autrement, les cookies sont des fichiers enregistrés par les sites web sur votre ordinateur. A chaque fois que vous connectez sur un serveur, celui-ci lira les cookies qu'il a déposé sur votre ordinateur et peut les mettre à jour. Ces cookies permettent au serveur de mémoriser des informations sur votre ordinateur, et non sur le serveur lui-même. Ces fichiers sont des fichiers texte, et ne sont donc pas des programmes exécutables (ce ne sont donc pas des virus).

Échange de cookies HTTP entre client et serveur.

Ces cookies peuvent contenir absolument tout et n'importe quoi, tant que le site aura jugé utile d'enregistrer ces informations dans le cookie. Ils peuvent y enregistrer vos MDP et login, ce qui vous permet de vous maintenir connecté et de ne pas avoir à les saisir à chaque connexion, par exemple. Les sites d'e-commerce sauvegardent aussi les paniers de produits réservés/sélectionnés avant qu'on passe leur commande. Comme autre exemple, certains services de publicité en ligne suivent les internautes 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 : chaque affichage d'une publicité sur un site enregistrera que le site a été visité dans le cookie, permettant au site publicitaire de savoir quels sont les sites visités par chaque utilisateur.

Quelques extensions de navigateur web permettent de supprimer ou de bloquer ces cookies traceurs. On pourrait notamment citer l'extension Privacy Badger pour Firefox et Chrome, ou encore Ghostery. 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 quand à l'utilisation de cette option. De plus, cette option doit être activée dans les options du navigateur : elle est désactivée par défaut.

Il existe 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, ceux-ci 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.

Il est demandé aux navigateurs de supporter a minima :

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