Synthèse Réseau

Contents

Synthèse Réseau 1
Introduction au modèle OSI 2
- Couche applicative 4
  - Le DNS 4
  - Mail 5
  - Le web 6

Couche de transport et transport fiable 6
- L'UDP (User Datagram Protocol) 6
- Le TCP (Transmission Control Protocol) 7

Implémentation du transport fiable dans TCP 11
- TPDU TCP 12
- Temporisateur TCP 12
- Envois des acquits 13
- Fast retransmit 13
- Gestion des pertes avec TCP 13
- Algorithme de Nagle pour l'émission de TPDU 13
- Contrôle de flux et de la fenêtre 14
- Connexion et déconnexion TCP 14
- Contrôle de la congestion 16

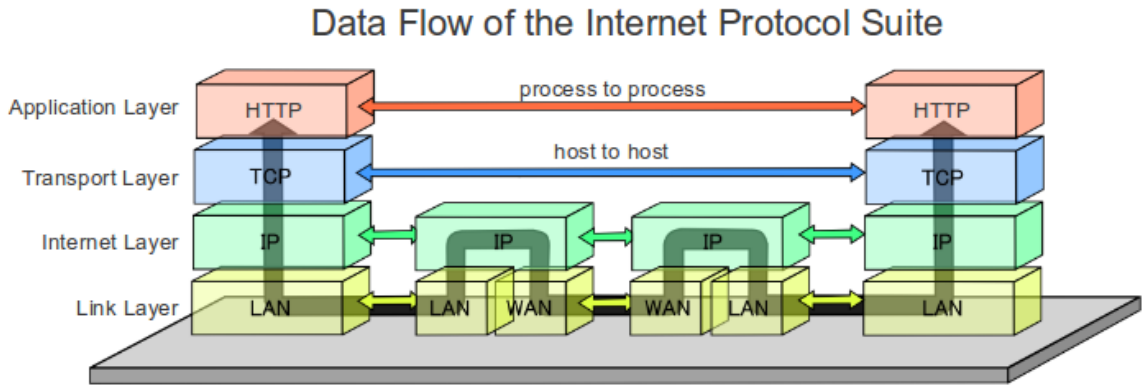
Couche internet et routage 16
- Identification des machines 17
- Sous-réseaux 17
- Routage 17

Protocole IPv4 et IPv6 23
- IPv4 vs IPv6 23
- CIDR vs classes 24
- Types d'adresses 24
- Paquet IPv4 24
  - Fragmentation IPv4 25
  - Eviter les boucles 25
- ICMPv4 25
- Paquet IPv6 25
  - Différence avec l'IPv4 26
- Types d'adresses IPv6 26
- ICMPv6 27
  - Neighbor Discovery Protocol (ND) 27
- Transition vers IPv6 27
  - Tunnel-brokers 27
  - 6to4 28
  - Déploiement dual-stack 29
  - 6rd 29
- Pallier les problèmes de l'IPv4 en attendant l'IPv6 29
  - Fonctionnement du NAT 29

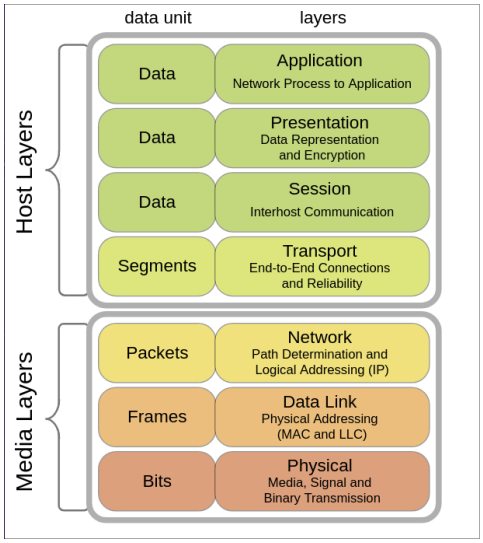
Routage à grande échelle sur Internet 30
- Routage intra-domaine 31

Comment faire sur de grands réseaux 32
Routage inter-domaine 32
Politiques de peering 32
Routage multicast 33
Types de multicast 33
Différentier les traffics multicast 34
Le protocole IGMP 34
Routage multicast 35
Routage multicast dans Internet 35
Couche d’accès réseau (les trames) 36
Délimitation de l'information 36
Indication de la longueur 36
Character stuffing 37
Bit stuffing 37
Détection d'erreurs 37
Accès au media 38
Eviter les collisions 38
Différentes technolgies d’accès réseau 39
Protocole ARP (IPv4) 40
Au sein d'un même sous réseau 40
Et lorsqu'il y a plusieurs sous-réseaux ? 40
Le protocole NDP (IPv6) 41
Ethernet 42
La trame Ethernet 42
CSMA/CD 42
Topologie 42
Multicast Ethernet IPv4 43
Multicast Ethernet IPv6 43
Interconnexions (switchs et hub) 43
Spanning Tree Protocol (STP) 43
PPP - Point to Point Protocol 44
Sous-protocoles de PPP 45
Sécurité réseau 46
Qu'est ce que la sécurité réseau et comment ? 46
Le chiffrement 46
KDC (Key Distribution Center) 47
Autorités de certifications 47
Contrôle d'accès (firewall et proxy) 48
IPSec (IP Security) 48
Le protocole AH 49
Protocole ESP 49
La sécurité des réseaux sans-fil 49
Les VPN 50

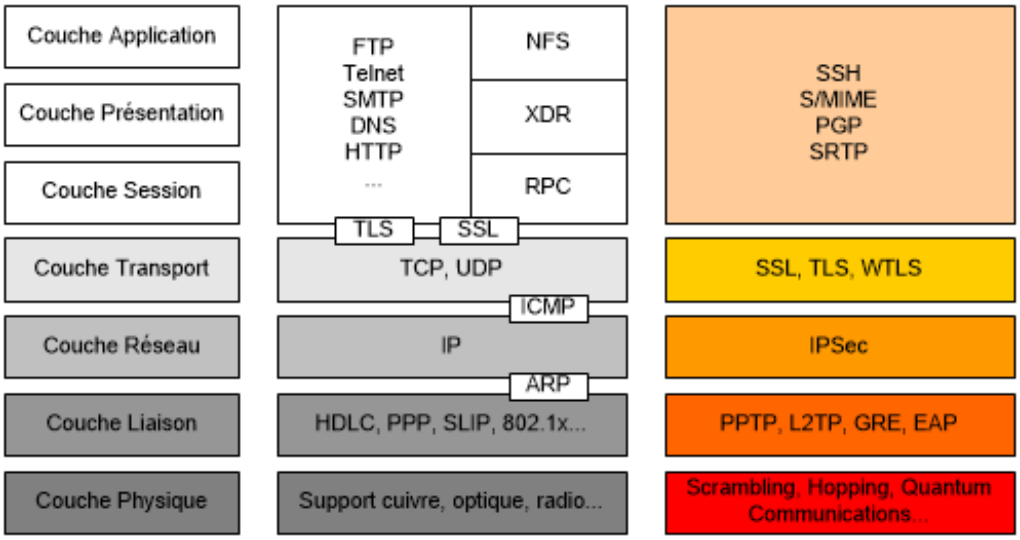
Introduction au modèle OSI



Le modèle OSI est une norme de communication réseau proposée par ISO. Elle met ainsi en relation plusieurs protocoles de communications différents (IP, HDLC, TCP, UDP, HTTP, etc).



Voici par exemple une liste de différents protocoles pour chaque couche du modèle.



Couche applicative

Si on fait abstraction de toutes les couches en dessous de la couche application, on trouve le protocole applicatif. Le protocole applicatif définit comment les données de l'application peuvent être demandées et envoyée (par exemple via HTTP pour des sites internet, IMAP pour recevoir des emails ou encore SMTP pour envoyer des emails).

Le protocole applicatif est le langage utilisé par l'application pour communiquer, il décrit donc la forme des messages et le sens des échanges (définition syntaxique et sémantique).

Pour s'identifier, les applications utilisent un port et une IP (IPv4 ou IPv6), l'IP indique la machine et le port défini l'application émettrice ou destinataire (exemple, 80 pour HTTP, 443 pour HTTPS, 22 pour SSH, 53 pour DNS, etc).

Le DNS

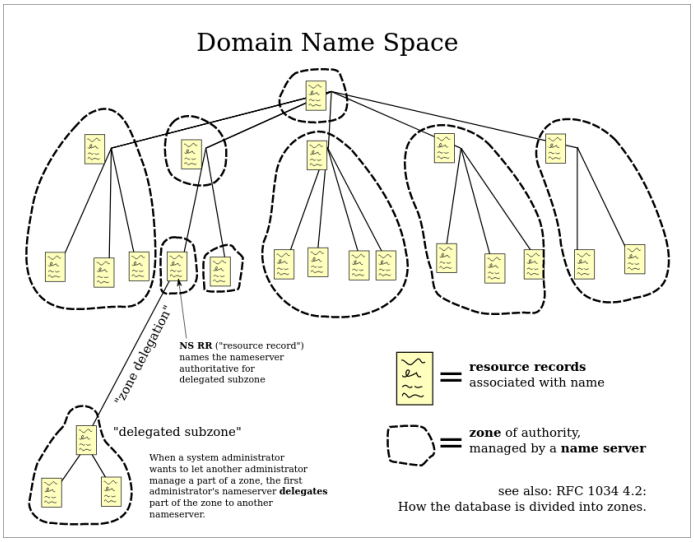
Chaque machine est identifiée au moyen d'une adresse IP (codée sur 32 bits IPv4 ou 128 bits avec l'IPv6).

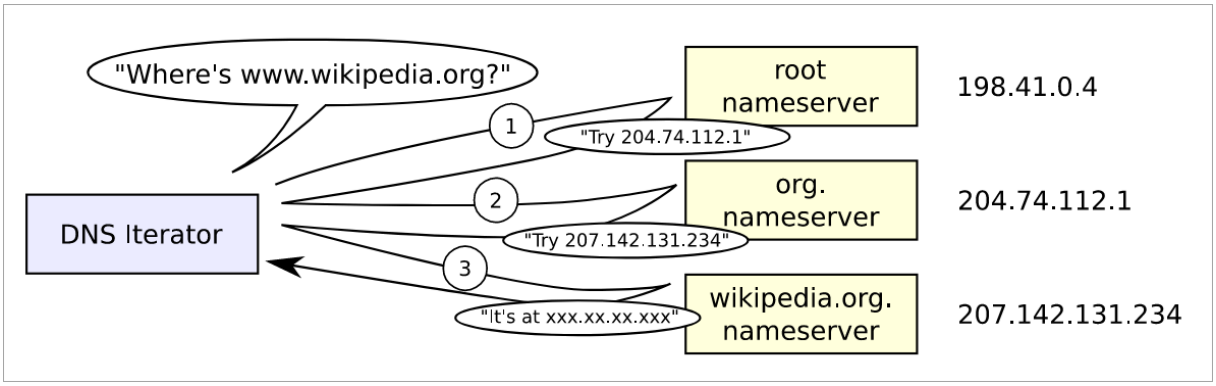
Le but du DNS (Domain Name System) est d'agir comme une sorte d'annuaire, ainsi à la place de devoir retenir des choses tel que `2001:41d0:404:200::597` il suffit de retenir `snowcode.ovh`. Il est donc possible de réserver un nom de domaine (généralement payant).

Une première manière de gérer cela serait d'avoir un fichier texte liant un nom et une adresse IP, par exemple avec `/etc/hosts` qui lie automatiquement `localhost` à l'adresse `127.0.0.1`.

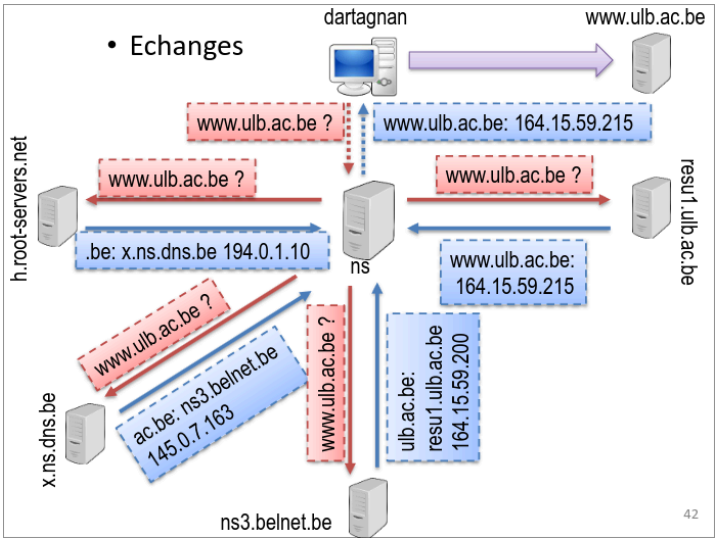
Le problème est qu'il serait impossible de synchroniser une base de donnée contenant tous les noms de domaines de tout le monde sur tous les appareils. On pourrait alors décider de créer une base de donnée centralisée, le problème est alors que tout le monde dépendrait d'un unique serveur pour les noms de domaines.

C'est pour cela qu'il y a plusieurs serveurs liés entre eux. Par exemple si on veut aller sur `swilabible.be`, on va d'abord demander au serveur mondial où se trouve le DNS de `be`, ce dernier peut alors ensuite renvoyer où se trouve `swilabible.be`. Il est donc possible d'avoir plus de niveaux d'imbrications.





Il est donc possible de créer son propre DNS pour gérer ses propres sous domaines.



Le serveur qui va effectuer cette recherche est généralement celui qui est proposé par le réseau local (sauf s'il a été spécifiquement été précisé dans la configuration du système) via le protocole DHCP.

Mail

Les mails utilisent plusieurs protocoles, le SMTP (via le port 25) permet d'envoyer des messages à des serveurs (tel que gmail.com, outlook.com, etc), la réception d'un message (ouvrir sa boite mail) se fait via les protocoles POP3 (si on veut télécharger tous les mails localement) ou IMAP (si on veut ne pas avoir à télécharger tout localement).

Les mails doivent toujours respecter la RFC 1855 afin de pouvoir être bien reçue et comprise par son destinataire.

Pour transmettre une pièce jointe (binaire) en texte, on peut utiliser le base64 qui va encoder le binaire avec des caractères alphanumériques et quelques caractères spéciaux.

Le problème est que les protocoles de mail sont très vieux, ainsi, il est tout à fait possible de prétendre être quelqu'un d'autre, aussi les mails ne sont pas chiffrés et beaucoup de fournisseurs d'accès internet bloquent le port 25.

Il est cependant indispensable de pouvoir supporter les mails, car les emails sont devenus incontournables au fil du temps.

Le web
Le web a été développé au CERN à Genève, derrière le "web" il y a plusieurs éléments (normes, protocoles, applications) :

- HTML (description de documents) ;
- Des serveurs web pour les délivrer (exemple apache ou nginx) ;
- Des clients pour les lire (Firefox, Chrome, etc) ;
- Le protocole HTTP qui permet à ces deux éléments de communiquer (versions allant de 1 à 3).
- Définition de l'URL (protocol://machine:port/chemin), par exemple (https://books.snowcode.ovh:8888/hello.html)

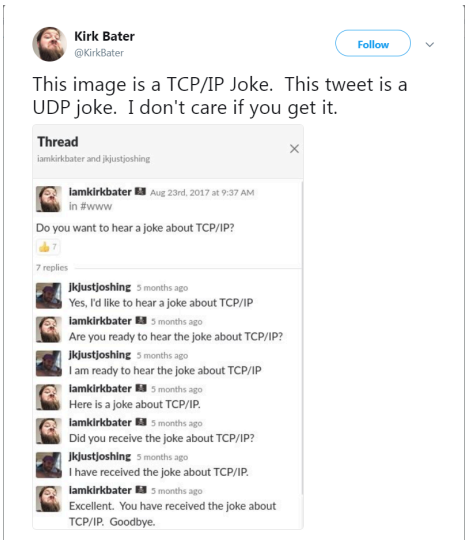
Chaque requête contient une commande (GET, HEAD, POST, PUT, DELETE), un entête (contenant des informations comme un token d'accès pour les cookies, d'où on vient, les métadonnées sur l'appareil, etc), et le corps qui est le contenu de la requête (page ou formulaire).

Ces entêtes peuvent être utilisés pour identifier et tracker des utilisateurs, car leur combinaison permet d'identifier des utilisateurs avec une certaine précision. Cela peut notamment être testé sur amiunique.org.

Couche de transport et transport fiable

La couche applicative repose sur la couche de transport. Cette dernière s'en fout du type de donnée utilisée, cette couche a seulement pour but de transférer les données.

Il existe deux protocoles, le **TCP** (Transmission Control Protocol) qui permet d'envoyer des informations de manière fiables (en vérifiant la bonne réception des "paquets" de données), et l'**UDP** (User Datagram protocol) est un protocole qui envoie les paquets sans se soucier de la bonne réception. Ce dernier, bien que moins fiable, est beaucoup plus rapide.



Il faut donc connaitre le port du programme à contacter, pour cela le système maintient un annuaire liant un numéro de port à une application.

L'UDP (User Datagram Protocol)

40	**320**	Source Port	Destination Port
44	**352**	Length	Checksum
48	**384+**	Data	

Avec l'UDP on va simplement transmettre les données sans se soucier de leur bonne réception. Ainsi pour chaque message (TPDU, Transport Protocol Data Unit) il faut connaitre le numéro de port source (et destination ainsi que la longueur du message et éventuellement un "checksum" permettant de vérifier l'intégrité des informations.

Le protocole UDP est très utilisé pour les applications qui ont besoin d'aller vite, même si cela veut dire de potentiellement perdre des informations. Par exemple pour les jeux massivement multijoueurs, les diffusions en direct de vidéo ou audio, etc.

Le TCP (Transmission Control Protocol)

Le problème avec l'UDP est qu'il n'y a aucune vérification de la bonne réception des paquets ou encore de leur ordre ou de leur intégrité.

Le but du protocole TCP est de garantir l'intégrité des données.

Transfert fiable

TCP est donc un protocole qui implémente le "transfert fiable", nous allons voir ici en quoi consiste le transfert fiable.

Le transfert fiable est une façon de transférer l'information entre un émetteur et un récepteur de telle sorte à pouvoir palier à des pertes, des duplications, des altérations ou un désordre parmi les informations.

Pour cela, chaque TPDU (Transport Protocol Data Unit) contient un "checksum" permettant de vérifier que l'information n'est pas corrompue ⟶ protection contre l'altération.

Et lors de chaque réception d'information, le récepteur doit confirmer la bonne réception, si l'émetteur ne recoit aucun acquis de bonne réception avant un certain temps (timer), il considère que l'information est perdue et la renvoi ⟶ protection contre la perte d'information.

Si l'acquis lui-même est perdu, l'émetteur va renvoyer l'information et le récepteur va réenvoyer son acquis, car ce dernier a déjà reçu l'information ⟶ protection contre la duplication et la perte d'acquis.

Chaque acquis et chaque envoi d'information est donc numéroté, il est ainsi possible de savoir pour chaque acquis à quoi il fait référence. Si un acquis est donc envoyé deux fois, l'émetteur pourra savoir à quelle information chaque acquis fait référence et agir en fonction. S'il envoie une information 1, reçois l'acquis pour 1, puis envois une information 2 et reçois de nouveau un acquis pour 1, il ne prendra pas compte du deuxième acquis ⟶ protection contre la duplication d'acquis.

Les acquis et les informations étant ainsi numérotées et allant dans un ordre de croissant. Et puis ce que l'émetteur attend toujours d'avoir reçu une confirmation de bonne réception de chaque partie de l'information, ce protocole assure donc que les informations sont reçues dans le bon ordre → protection contre le désordre.

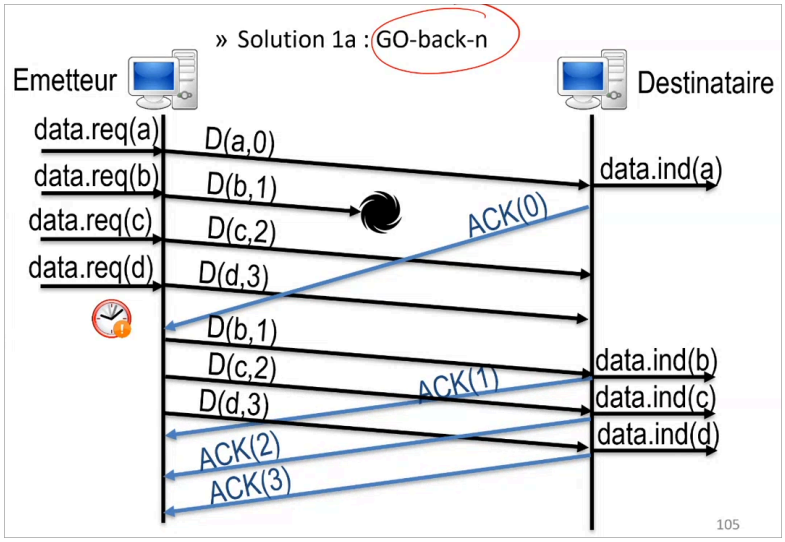
Fenêtre glissante

Les performances de TCP sont bien plus mauvaises que UDP car si le ping est élevé le round-trip time (temps allé-retour) va être très élevé aussi, cela sera donc très lent de tout transmettre. Pour résoudre ce problème, on peut alors utiliser un système de "fenêtre glissante", on va envoyer plus d'information avant d'attendre un acquis (donc moins d'acquis et pas d'envois de trop de données).

La fenêtre défini une série de numéros de séquences qui peuvent être envoyés sans devoir attendre un acquis. Une fois cette fenêtre épuisée, il faut attendre un acquis (pour toute la fenêtre) pour pouvoir recommencer.

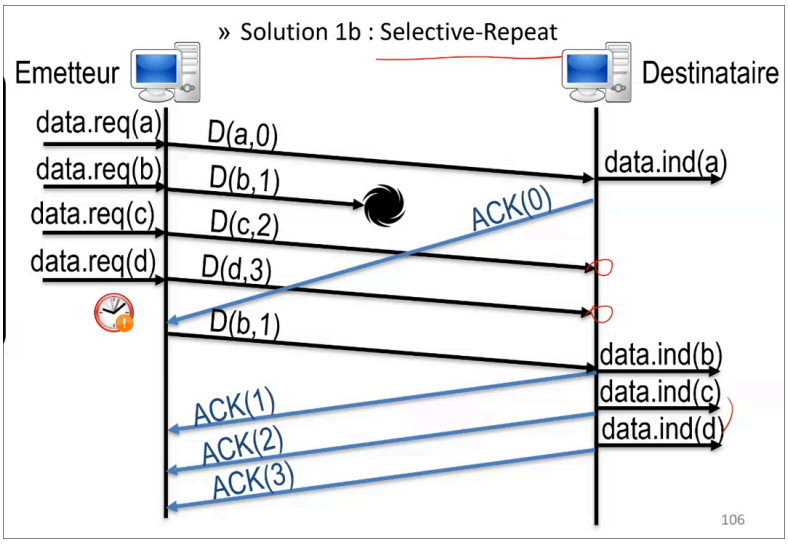
Perte d'information

Lorsqu'une perte d'information survient, il y a plusieurs manières de palier à une perte.

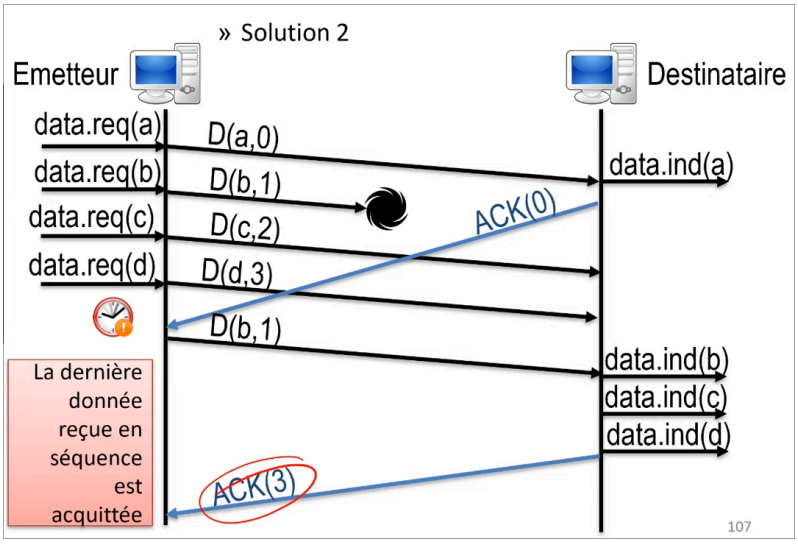


La première, c'est le **go-back-n** dans lequel le destinataire oublie tous les TPDU reçus hors séquence, ainsi s'il doit recevoir 0, 1, 2, 3, mais ne reçois pas 1, il ne va pas tenir compte de 2 et 3 et va dire à l'émetteur par un acquis : "J'ai seulement reçu le TPDU 0". L'émetteur devra alors re-envoyer les TPDU 1, 2 et 3 qui seront alors acquittés par le destinataire.

Cette méthode est avantageuse pour le destinataire, car il n'a pas besoin de retenir les TPDU hors-séquence, mais peu avantageuse pour l'émetteur qui doit tout réenvoyer.



La deuxième méthode est le **Selective Repeat** (plus courante aujourd'hui) qui consiste à garder en mémoire les données hors séquence, si on reprend l'exemple précédent, si on attend de recevoir 0, 1, 2 et 3 et que l'on ne reçoit pas 1, alors on acquitte 0 qui a bien été reçu, l'émetteur envois alors la donnée suivate, 1. Le destinataire va ensuite acquitter tous les autres paquets reçus (1, 2 et 3) qui ne seront donc pas ré-envoyés.



Pour ne pas avoir à acquitter tout un par un, on peut également acquitter la dernière information reçue en séquence (dans ce cas 3), ce qui équivaut à acquitter 1, 2 et 3 d'un coup, ce qui est donc plus efficace.

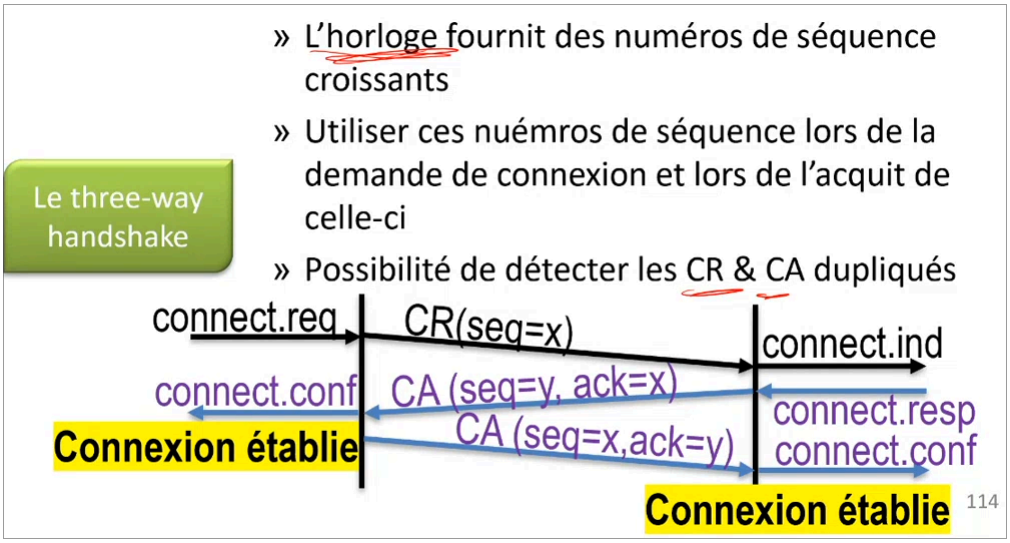
Capacité de traitement variable

Seulement, la capacité de traitement du destinataire peut varier, c'est pourquoi il va préciser dans ses acquis la taille actuelle de la fenêtre, plus la capacité de traitement du destinataire est grande, plus la fenêtre sera grande, et inversement.

À savoir qu'étant donné que les numéros de séquence sont réutilisés, il est possible d'avoir une duplication d'un acquis avec un certain numéro de séquence avec beaucoup de retard. Cela pourrait poser un problème si par hasard le numéro de séquence actuel est justement celui-là. C'est pourquoi la couche réseau (IP) s'occupe de faire un "timeout" sur les paquets, ainsi les paquets trop anciens sont juste oubliés, ce qui règle donc ce problème.

Connexion et déconnexion

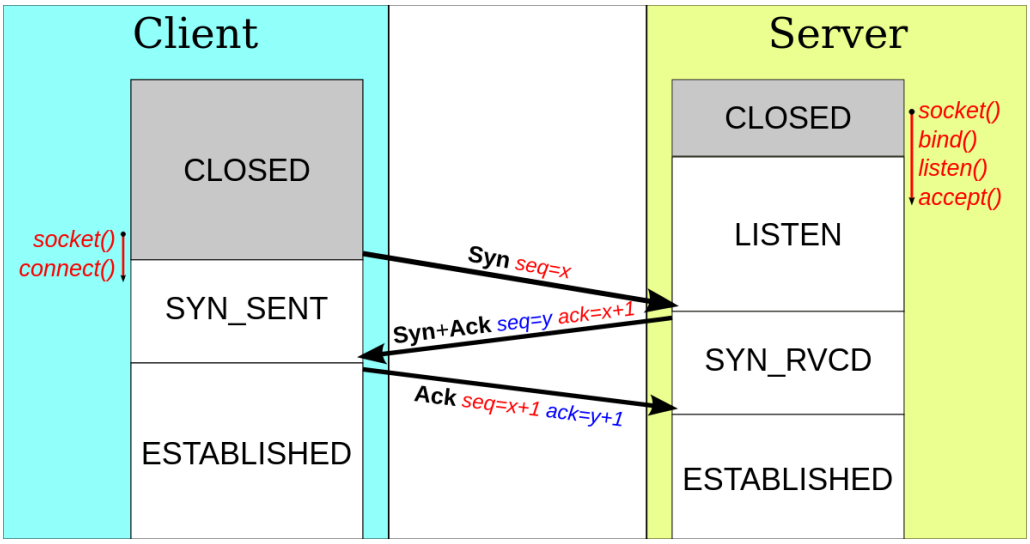
Pour pouvoir commencer à transférer des données, il faut d'abord établir une connexion pour partager des informations initiales. Pour ce faire, on utilise un **three-way handshake**.



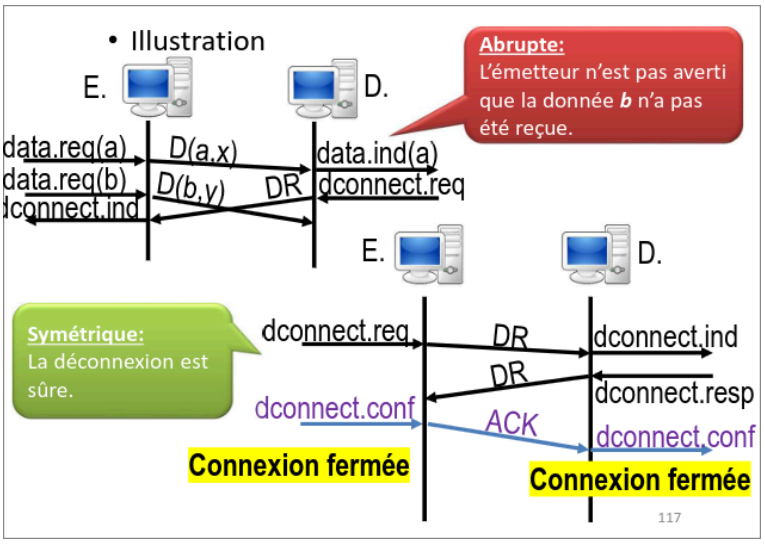
Le client va générer un numéro de séquence (x) et envoyer une demande de connexion au serveur avec ce dernier.

Le serveur va ensuite également générer un numéro de séquence (y) et acquitter la requête, la connexion est alors établie pour le client.

Enfin, le client va acquitter aussi, la connexion est alors établie pour le serveur.



Si une requête est dupliquée, le serveur va envoyer un acquis, mais le client répondra par un `REJECT` pour indiquer que la connexion est refusée, car il n'a pas fait de requête.



Pour ce qui est de la déconnexion, elle peut se faire soit de manière **abrupte**, c'est-à-dire que l'un des deux indique à l'autre "je me casse" et se déconnecte. Le problème, c'est que des données peuvent alors être perdues ou perdre l'information sur la déconnexion.

L'autre méthode est de se déconnecter de manière **symétrique**, autrement dit de manière similaire au three-way handshake. `A` envois à `B` une requête de déconnexion, `B` envois à `A` une requête de déconnexion, `A` acquitte la requête à `B` et se déconnecte (et B fait de même).

Communication bidirectionnelle

Souvent, il arrive que le client et le serveur doivent tous les deux transférer des données, ce qui complique donc un peu les choses.

Ainsi, on peut soit ouvrir deux connexions (une pour client ⟶ serveur et une pour serveur ⟶ client) mais cela ajoute donc beaucoup de trafic de contrôle et ralenti les choses.

Sinon, on peut utiliser le **piggyback** qui consiste à fusionner les TPDU de contrôle (acquis) et les TPDU de réponse en un seul TPDU ce qui diminue drastiquement donc la quantité de trafic de contrôle.

Implémentation de TCP

Voir sur la page suivante pour la description de l'implémentation du transfert fiable dans le protocole TCP.

Implémentation du transport fiable dans TCP

TCP est un protocole qui implémente le transfert fiable dont on a parlé juste avant.

Il comprend 3 phases,

- La phase de connexion qui utilise un three-way handshake
- Le transfert d'informations en utilisant des acquits comme vu précédemment
- La fermeture de connexion

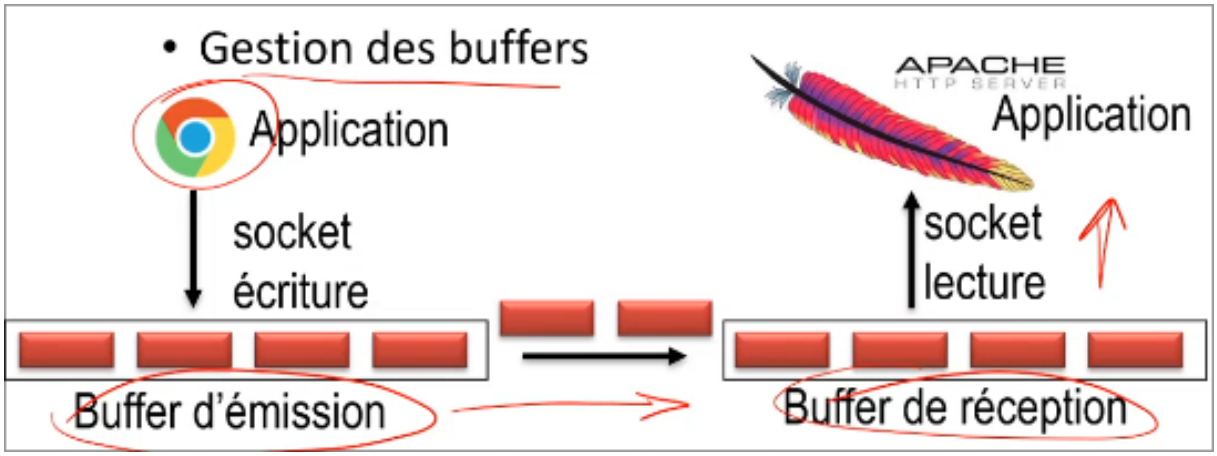
TCP fonctionne également en **unicast**, c'est-à-dire d'un destinataire à un autre et pas à un groupe de destinataire (pas multicast).

Une connexion dans TCP est identifié par quatre informations, le port source, le port de destination, l'IP source et l'IP destination.

TCP définit aussi un MSS (Maximum Segment Size) qui indique la taille maximum des données qui peuvent être dans un TPDU, cela dépends souvent de la couche réseau et de liaison utilisée, par exemple 1500 octets pour Ethernet.

Il y a par ailleurs une extension à TCP qui est utilisée dans la plupart des systèmes d'exploitations qui est le multipath TCP qui consiste à utiliser plusieurs accès réseaux simultanés pour transférer des données plus rapidement (par exemple utiliser la 4G et le Wifi en même temps).

Pour ne pas tout le temps envoyer des choses sur le réseau en permanence, TCP utilise des buffers. Ainsi, lorsque qu'une application veut envoyer des TPDU, il les place dans un buffer d'écriture, une fois plein, les informations sont envoyées. Lors de la réception, les données sont placées dans un buffer de destination, une fois toutes les données reçues, les données sont envoyées à l'application.



TPDU TCP

Un TPDU TCP est composé de plusieurs informations :

- Port source
- Port destination
- Numéro de séquence
- Numéro d'acquittement
- Taille de l'entête
- Des indicateurs (ACK pour acquis, SYN pour demander une connexion, FIN pour terminer une connexion, etc)
- La taille de la fenêtre glissante
- Le "checksum" pour vérifier les données
- Le pointeur de donnée urgente et les options, qui ne servent à rien ou sont facultatives
- Données

Ce système permet donc de faire de la communication bidirectionnelle étant donné qu'il est possible de mettre des données et un acquit dans un même TPDU. On considère en TCP que l'acquit est toujours le prochain numéro de séquence attendu.

Temporisateur TCP

TCP doit également définir un temporisateur, c'est-à-dire mettre un "timeout" au bout duquel, si aucun acquit a été reçu, le(s) TPDU sont considérés comme perdu et doivent être renvoyé.

Ce délai doit donc être plus grand que le RTT (Round-Trip Time) qui est le temps de faire un aller-retour entre un émetteur et une destination. Aussi, si le RTT est très variable, le délai du temporisateur sera plus grand.

La valeur du temporisateur est alors $ RTTmoyen + 4 * RTTvariation $.

Envois des acquits

Il y a plusieurs cas différents d'envois d'acquits en TCP, lors de la réception d'un nouveau TPDU :

- SI on reçoit un TPDU ayant le numéro de séquence attendu ET que toutes les données ont été acquittées jusque-là, ALORS on attend jusqu'à 500 ms ET si aucun TPDU n'arrive au bout de ce délai, ALORS on acquitte le TPDU
- SI on reçoit un TPDU ayant le numéro de séquence attendu MAIS que toutes les données n'ont pas été acquittées jusque-là, ALORS on envoie un acquit pour tous
- SI on reçoit un TPDU ayant un numéro de séquence plus grand que prévu, ALORS on re-duplique l'acquit précédent pour demander le TPDU manquant
- SI on reçoit un TPDU couvrant un trou dans la séquence, ALORS on envoie acquit pour demander le prochain TPDU de la séquence

Fast retransmit

Une amélioration de TCP est le **Fast Retransmit** qui permet de ne pas avoir à attendre le timeout du temporisateur pour renvoyer un ou des TPDU.

Pour ce faire, le destinataire va envoyer des acquits pour tous les TPDU de la séquence, même ceux qui sont perdus. Ensuite, le destinataire va envoyer trois acquits identiques pour le TPDU perdu.

La réception de ces trois acquits est vue comme une demande de ré-envois de ces données par l'émetteur qui n'a ainsi pas besoin d'utiliser son temporisateur dans ce cas. Cela permet donc d'aller beaucoup plus vite.

Gestion des pertes avec TCP

TCP retient uniquement les TPDU qui arrivent en séquence (donc avec les numéros de séquence attendus à chaque fois).

Mais TCP sauvegarde tout de même les TPDU qui arrive hors séquence afin de ne pas avoir à les redemander plus tard.

La fast-retransmit vu plus tôt permet à TCP d'aller plus vite pour demander les données perdues à l'émetteur.

Algorithme de Nagle pour l'émission de TPDU

Il serait fort peu pratique d'émettre des TPDU pour chaque petite donnée, car avoir beaucoup de petit TPDU causerait des problèmes de congestion du réseau.

L'idée est alors de combiner plusieurs TPDU dans un seul, plus gros TPDU. Le fonctionnement de cet algorithme est celui-ci :

1. Le premier octet est envoyé immédiatement
2. Tant que l'accusé de réception n'est pas reçu, on accumule toutes les données suivantes dans un seul TPDU (tampon ou buffer). Lorsque l'acquittent arrive, on envoie le TPDU.
3. On répète la deuxième étape.

Contrôle de flux et de la fenêtre

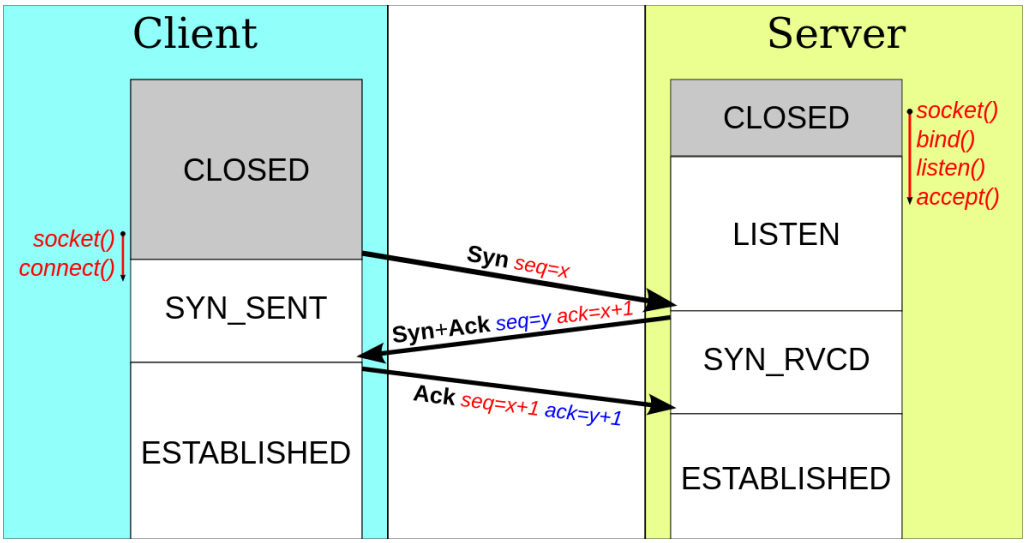
Le destinataire doit indiquer la taille de sa fenêtre (qui symbolise sa capacité de traitement) au fur et à mesure afin de ne pas être submergé de données.

L'émetteur de son côté est obligé d'attendre la réception d'un acquit lorsque la fenêtre est vide avant de recommencer à transmettre.

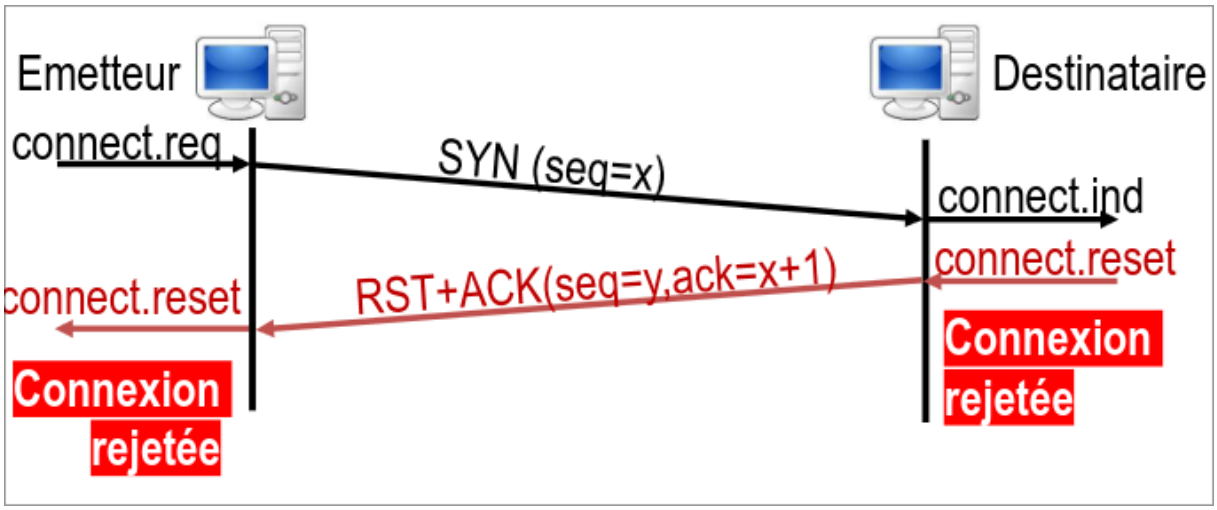
Connexion et déconnexion TCP

Une requête de connexion en TCP se fait avec le flag `SYN`. Le serveur peut ensuite soit accepter en utilisant les flags `SYN` et `ACK`, ou refuser avec `RST` et `ACK`. Si la connexion est acceptée par le serveur, le client envois un `ACK` pour signaler qu'il est prêt à commencer l'échange de données, ou un `RST` et `ACK` pour refuser et annuler la connexion.

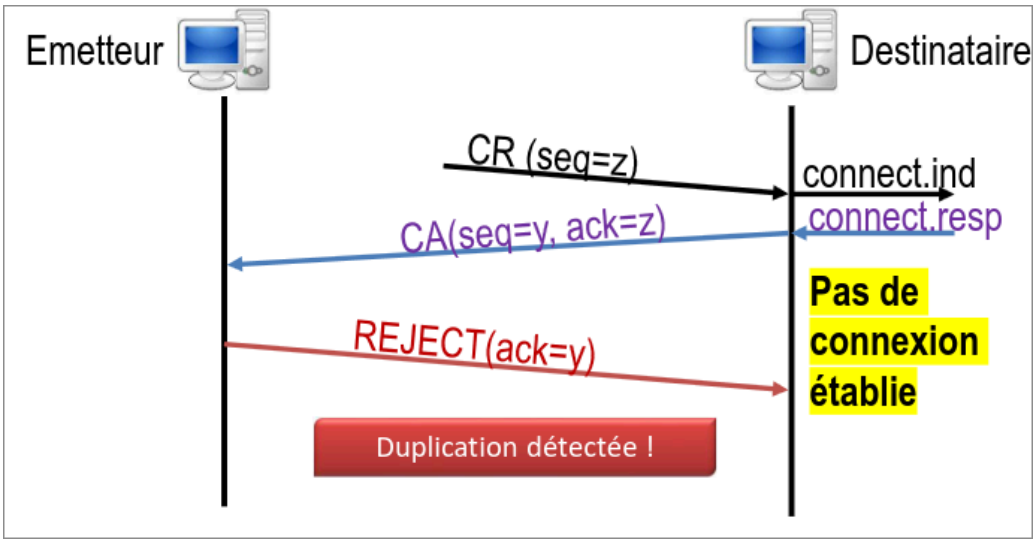
Voici ce qu'il se passe lorsque la connexion est acceptée par le serveur :



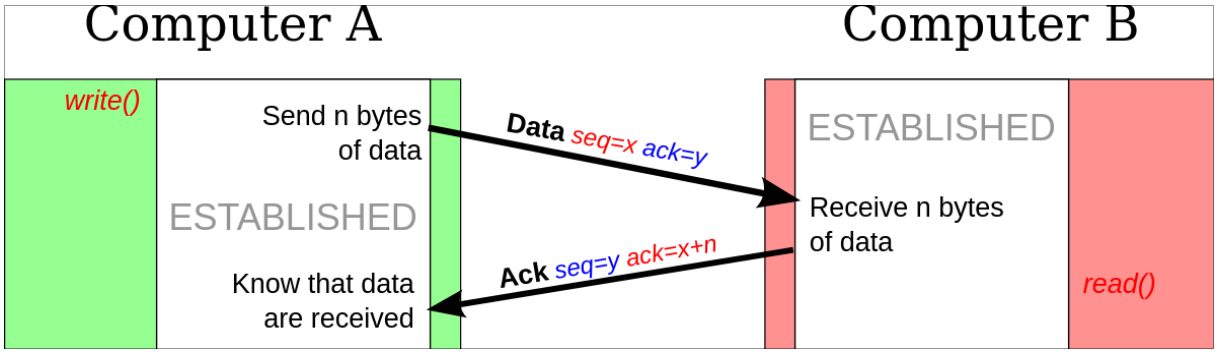
Voici ce qu'il se passe lorsque la connexion est refusée par le serveur :



Et voici ce qu'il se passe lorsque la connexion est invalide et est refusée par le client (compter que REJECT serait en TCP `SYN+ACK`) :

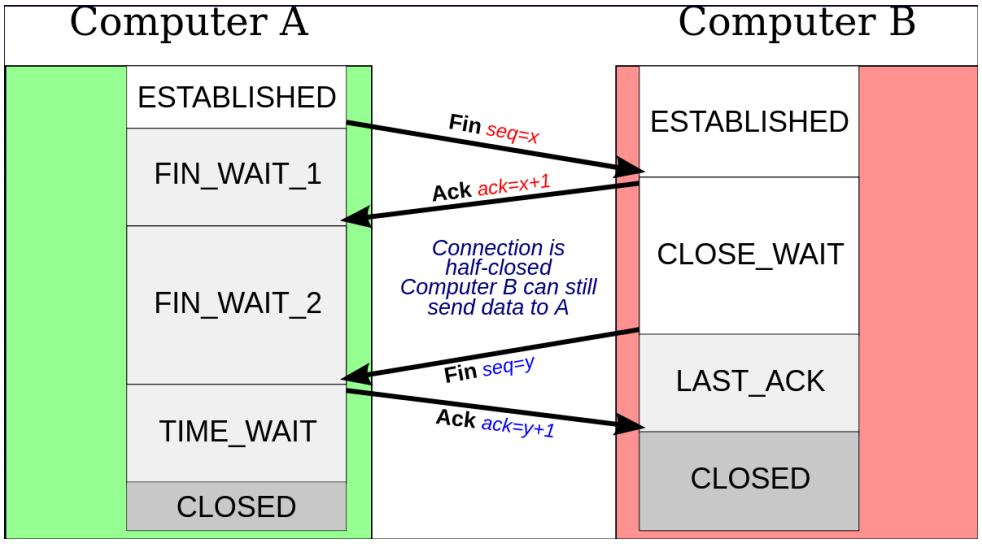


Une fois la connexion établie, un échange de donnée peut donc s'opérer :



Une fois l'échange de donnée terminé, la connexion peut être fermée correctement en utilisant une méthode similaire au three-way handshake utilisé pour la connexion. On envoie un TPDU avec un flag `FIN` pour demander la déconnexion, l'autre partie renvois donc un `ACK` transfert également quelques dernières données, puis envois un `FIN` à son tour qui a une réponse par un `ACK`. La connexion est alors fermée.

Il est aussi possible d'avoir une connexion abrupte, si un `FIN` est envoyé sans attendre d'acquit. Vous pouvez avoir plus de détail en lisant la partie plus tôt sur la déconnexion dans le transfert fiable.



Contrôle de la congestion

Tous les réseaux sur internet ne sont pas égaux, certains sont donc beaucoup plus lents que d'autres. Pour éviter de surcharger ces réseaux avec beaucoup de données, il est donc important de mettre en place un système permettant de limiter cette congestion.

Pour cela, l'émetteur va retenir une "fenêtre de congestion" (qui n'a rien à voir avec la fenêtre glissante).

Cette fenêtre de congestion indique la quantité de donnée qui peut être transmise, elle est mesurée en MSS (maximum segment size, c'est-à-dire la taille maximale d'un TPDU). Au départ, elle est à 1, et à chaque `ACK` reçu, elle augmente de 1. Cette donnée va donc grandir de manière exponentielle. Cette phase est appelée le **slow-start**.

Ensuite, une fois que la fenêtre atteint un certain maximum prédéfini, elle va ensuite grandir de manière linéaire en augmentant de 1 à chaque RTT (Round-Trip Time, donc il faudra attendre que toutes les données envoyées soient acquittées). Cette phase est appelée le **AIMD** (Adaptative Increase Multiplicative Decrease).

Ce qui est fait lorsque des données sont perdues dépends de comment on sait que les données sont perdues :

- Soit, on reçoit un **Fast-Retransmit** (3 acquits dupliqués), la congestion est alors définie comme légère et on définit le maximum de la fenêtre à la valeur actuelle de la fenêtre divisée par 2. Et on définit la valeur de la fenêtre au seuil. On reprend alors en mode linéaire (AIMD).
- Soit, il y a un **timeout du temporisateur**, la congestion est alors définie comme forte et on définit le maximum de la fenêtre à la valeur actuelle de la fenêtre divisée par 2. Et on met la valeur de la fenêtre à un et on recommence en slow-start (exponentiel).

Couche internet et routage

La couche internet qui soutient la couche de transport sert à faire acheminer les informations d'une machine source vers une machine destination. Cela se fait cependant sans garantie de fiabilité, c'est pour cela que le protocole TCP est nécessaire.

Certains systèmes, principalement ceux qui transitaient sur le réseau téléphonique, nécessitait une phase d'ouverture de connexion. Ce n'est cependant plus obligatoire aujourd'hui.

Identification des machines

Les machines sont identifiées par des adresses sur 32 bits (IPv4) ou 128 bits (IPv6).

Ces adresses sont écrites généralement sous forme décimale pointée. On regroupe donc les adresses par octets (8 bits) que l'on représente sous forme décimale (pour l'IPv4).

Par exemple :

```
11000001 10111110 01000000 01111100
193     .190     .64      .124
```

En IPv6, les adresses sont codées sur 128 bits, sont représentées en hexadécimal par blocs de 16 bits séparés par `:`. On peut également abréger les `0` consécutifs en utilisant `::`

Ainsi `2001:0bc8:38eb:fe10:0000:0000:0000:0011` devient simplement

`2001:0bc8:38eb:fe10::11`

Sous-réseaux

Pour pouvoir se connecter directement à une autre machine, il faut que cette dernière se situe dans le même **sous-réseau**. Chaque sous-réseau est lui-même identifié par une adresse IPv4 particulière.

Pour savoir si machines sont directement connectées sur un sous-réseau donné. Il faut appliquer un **masque de sous-réseau** sur l'**IP du réseau**.

```
IP RES : 11000000.10101000.00000001.00000010 - 192.168.1.2
MASQUE : 11111111.11111111.11111111.00000000 - 255.255.255.0
         ----------------------------------
PREFIX : 11000000.10101000.00000001.00000000 - 192.168.1.0
```

Cela signifie qu'il y a 256 adresses possibles (2(32 - nombre de 1 dans le masque)), qui auront toutes un certain préfixe défini plus tôt. Par exemple, 192.168.1.1, 192.168.1.5, 192.168.1.255 sont toutes des adresses faisant partie d'un seul et même sous-réseau 192.168.1.2.

Note : le sous-réseau peut plus simplement être indiqué via la notation <IP RÉSEAU>/<NOMBRE DE 1 DU MASQUE>. Par exemple, plus tôt, on avait un sous-réseau 192.168.1.2/24. Il est aussi bon de noter qu'il y a également une adresse de **broadcast** qui permet de communiquer des paquets à tout le monde dans le réseau.

Routage

La plupart du temps, on communique avec des machines qui sont en dehors de notre réseau direct. Il faut donc connecter les routeurs entre eux et utiliser des algorithmes pour pouvoir acheminer les informations là où il faut.

Routeur

Pour transférer des paquets (unité d'information dans la couche internet), ces derniers transitent par des **routeurs**. Ces derniers sont des relais au niveau de la couche réseau et ont pour but de trouver le meilleur chemin pour faire transiter l'information.

Ainsi, il peut interconnecter des réseaux de natures différentes, le routeur (box) chez vous peut connecter votre réseau sur le réseau téléphonique (xDSL) ou de télédistribution (coaxial).

Le routeur a aussi plusieurs interfaces réseau avec lesquelles il communique, tel que le Wifi, une connexion au réseau de l'ISP (Internet Service Provider tel que Proximus), etc.

Modèles de routage

Il existe deux modèles de routage de l'information :

	Datagrammes	Circuits virtuels
Configuration du connexion	Pas obligatoire	Obligatoire
Adressage	Le paquet contient les adresses source et de destination complètes	Le paquet contient l'identifiant du circuit virtuel.
Informations d'état	Nul autre que la table de routeur contenant le réseau de destination	Chaque numéro de circuit virtuel entré dans la table de configuration, utilisé pour le routage.
Routage	Paquets acheminés indépendamment	Route établie à la configuration, tous les paquets suivent la même route.
Effet d'une panne de routeur	Uniquement sur les paquets perdus lors d'un crash	Tous les circuits virtuels passant par un routeur défaillant sont interrompus.
Contrôle de congestion	Difficile car tous les paquets routés indépendamment.	Simple en pré-affectant suffisamment de buffers à chaque circuit virtuel lors de la configuration, puisque le nombre maximal de circuits est fixe.

Le mode **circuit virtuel** qui va sur base d'une information donnée par l'émetteur déterminer le chemin entre la source et la destination (donc avec phase d'ouverture), ensuite va transférer les paquets sur cette route, une fois terminé une des entités annonce une déconnexion et interrompt le circuit.

Ce mode, bien que simple à comprendre (c'est simplement un circuit), est très peu pratique pour de grands réseaux tels qu'internet, car chaque routeur devrait connaitre tous les autres routeurs et si un routeur ne fonctionne plus, cela pourrait mettre en péril une grande partie du réseau.

C'est pourquoi la méthode utilisée par internet est le mode **datagramme** où chaque paquet mentionne l'adresse de destination, sur base de cette information les routeurs orientent les paquets vers la destination avec une **table de routage**. Les paquets sont donc transférés individuellement, il n'y a pas besoin de phase de connexion/configuration et le réseau peut grandir beaucoup plus facilement. Il faut toute fois noter qu'il est possible que plusieurs paquets ayant la même source et la même destination peuvent ne pas prendre le même chemin.

Routage statique

Les routes vers les différentes destinations sont définies manuellement par l'administrateur·ice.

Cette méthode est très utilisée, car conceptuellement simple, mais ingérable pour les réseaux de taille importante et variable.

Routage dynamique global

Le routage dynamique global consiste à faire en sorte que chaque routeur dispose d'une carte complète du réseau et déterminer manuellement les chemins les plus courts vers les destinations.

Algorithme état liaisons

Au début, les routeurs échanges des informations entre eux pour pouvoir aider à construire une carte du réseau.

Chaque routeur connait au départ les routes auquel il est relié et les couts (les couts sont calculés sur base de la vitesse de la connexion) associés.

Les routeurs publient ces informations aux autres routeurs afin de construire la carte du réseau. Chaque route est échangée dans un LSP (Link State Packet) qui comprend l'identification du routeur qui annonce ainsi que la destination et son cout associé.

Chaque LSP est à un numéro de séquence associé qui est incrémenté par la source à chaque nouvel envoi. Ainsi, les routeurs peuvent se souvenir des derniers numéros de séquence pour chaque LSP afin de ne pas renvoyer les mêmes LSP en boucle dans le réseau.

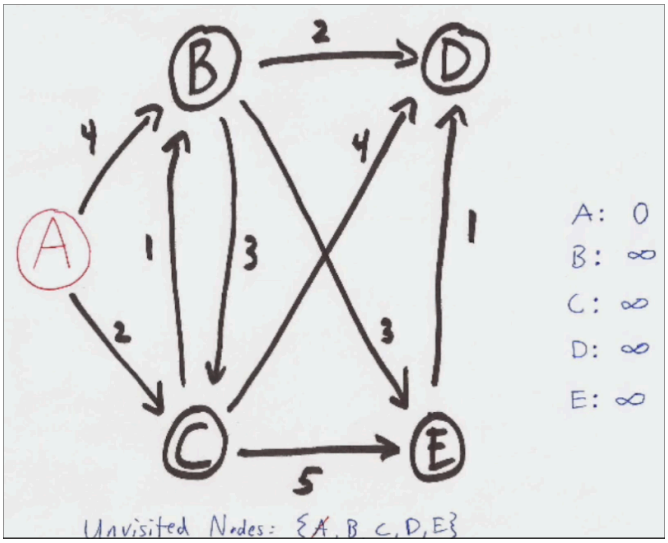
Une fois que la carte du réseau est établie sur base de ces envois, pour chaque requête, on établit le chemin le plus court entre le routeur actuel et la destination. Pour ce faire, on utilise l'**algorithme de Dijkstra**.

L'algorithme de Dijkstra

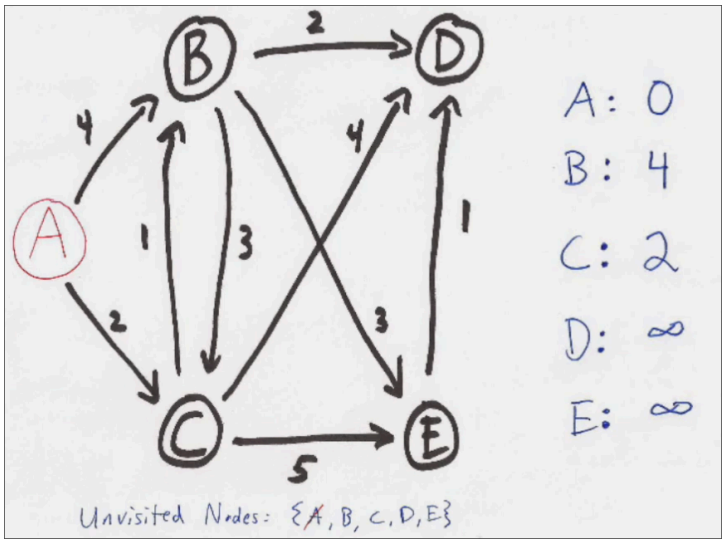
> *Les images et les explications de cet algorithme viennent de cette vidéo.*

Au départ, on indique que le cout pour atteindre le routeur actuel est 0 et ceux pour atteindre les autres routeurs est infinie, car on n'a pas encore calculé le cout.

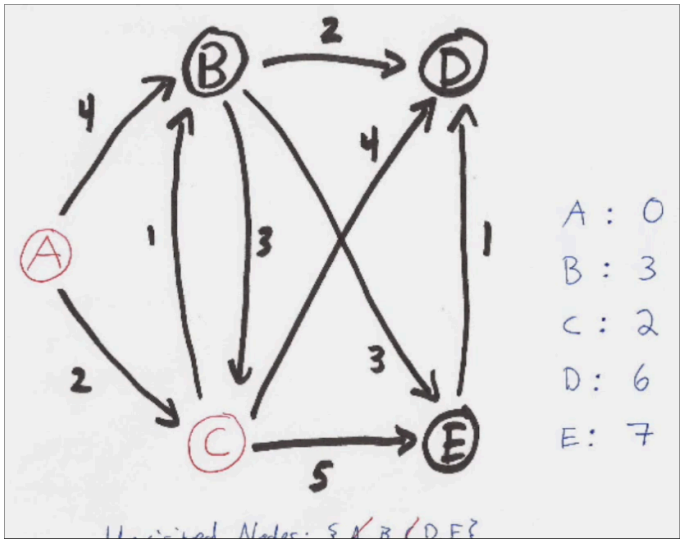
On garde également une liste des routeurs à "visister".



Ensuite, on peut compléter le tableau des couts en indiquant les couts pour atteindre les routeurs voisins du routeur actuel.



On peut ensuite faire la même chose depuis le routeur ayant le cout le plus bas, dans ce cas-ci, le routeur C.

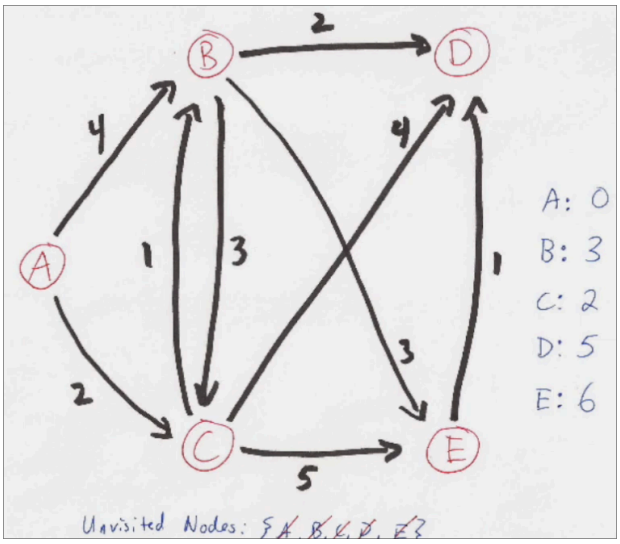


Depuis C, on va ainsi regarder pour chaque routeur voisin de C, quel est le cout pour y accéder. Si le cout total (donc le cout pour atteindre C

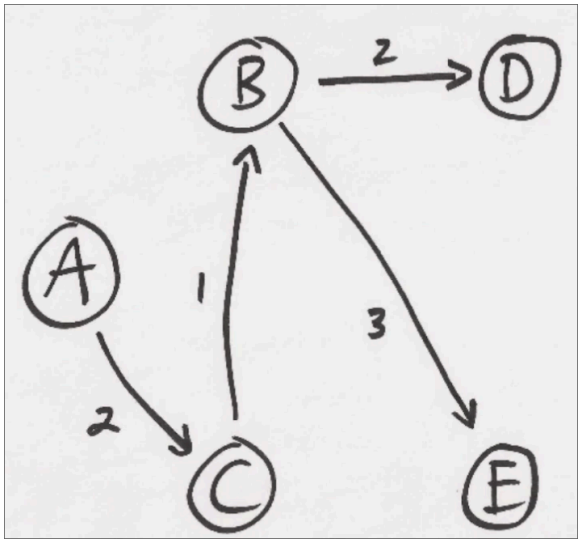
1. le cout pour atteindre le routeur en question) est plus petit que le

cout noté précédemment, on met ainsi le cout à jour.

On continue alors le processus jusqu'à avoir fait cette vérification depuis tous les routeurs à visiter.



Une fois cela fait pour tous les routeurs, on peut alors établir quel est le chemin le plus court pour atteindre chaque routeur depuis le routeur courant (A) :



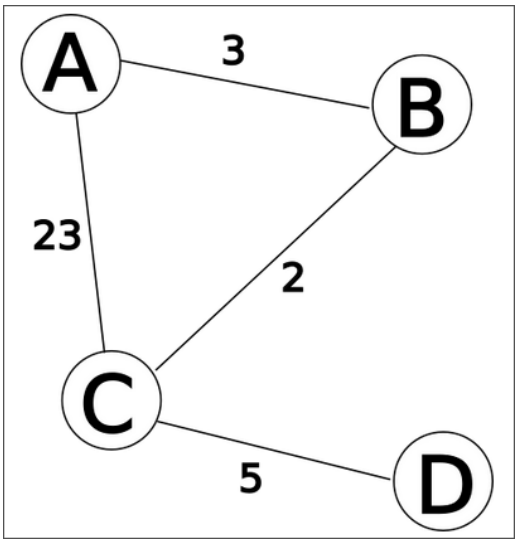
Sur base de cela, nous pouvons donc créer la table de routage. Dans le cas de A, c'est assez simple, car tous les paquets seront envoyés à C.

Routage dynamique décentralisé

À la place d'avoir une carte complète du réseau, on garde simplement une matrice des distances entre les différents routeurs voisins. Il est ainsi possible de savoir, pour chaque routeur cible, quel routeur doit être utilisé (car plus rapide). Ceci a un grand avantage pour de grands réseaux parce qu'il n'est pas nécessaire de connaitre la carte complète du réseau pour pouvoir envoyer des paquets.

Algorithme vecteur de distance

Imaginons le réseau de routeurs suivant :



Au départ, chaque routeur établit une matrice indiquant le cout pour aller à chacun de leur voisin :

T=0

from A	via A	via B	via C	via D
to A				
to B		3		
to C			23	
to D				

from B	via A	via B	via C	via D
to A	3			
to B				
to C			2	
to D				

from C	via A	via B	via C	via D
to A	23			
to B		2		
to C				
to D				5

from D	via A	via B	via C	via D
to A				
to B				
to C			5	
to D				

Ensuite, tous les routeurs envoient ces informations sur le réseau à leur voisin. Ainsi A envois sa matrice à B et C, B l'envois à A et C, C l'envoi à A, B et D et D l'envoi à C.

Sur base de cette information, chaque routeur peut recalculer les couts. Ainsi A va recalculer le cout d'aller à C sur base de la matrice de B et va remarquer qu'aller à C en passant par B est beaucoup plus rapide (3 + 2 = 5 contre 23).

Ils vont ainsi mettre à jour leurs matrices :

T=1

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D			28	

from B	via A	via B	via C	via D
to A	3		25	
to B				
to C	26		2	
to D			7	

from C	via A	via B	via C	via D
to A	23	5		
to B	26	2		
to C				
to D				5

from D	via A	via B	via C	via D
to A			28	
to B			7	
to C			5	
to D				

Ensuite, le processus va se répéter jusqu'à ce que la carte de tout le monde soit complète. Ainsi A va par exemple apprendre le cout pour aller à D est seulement de 10 en passant par B (3+7 = 10).

T=3

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D		10	28	

from B	via A	via B	via C	via D
to A	3		7	
to B				
to C	8		2	
to D	13		7	

from C	via A	via B	via C	via D
to A	23	5		15
to B	26	2		12
to C				
to D	33	9		5

from D	via A	via B	via C	via D
to A			10	
to B			7	
to C			5	
to D				

Une fois la table de tout le monde complétée, on peut ainsi définir la table de routage assez simplement en regardant par où il faut passer pour atteindre chaque routeur afin d'avoir la distance la plus courte.

Ainsi, la table de A indique par exemple que tous les paquets seront envoyés à B.

Lorsqu'un ou plusieurs liens avec un routeur est down, les routeurs qui le remarquent indique que le lien vers ces routeurs a un cout infini. Il propage alors cette information dans le réseau pour établir à nouveau la matrice et la table de routage de tous les autres routeurs. Cela se fait de manière assez lente ce qui fait que pendant un certain temps certaines destinations pourrait être inaccessibles.

Protocole IPv4 et IPv6

IP est le protocole de base d'internet et globalement le seul protocole de la couche internet. Il a cependant deux versions majeures, la version 4 et la version 6.

IPv4 vs IPv6

La version 4 d'IP est celle qui est la plus répandue sur Internet, mais qui est toutefois limitée par le nombre d'adresses possibles, car les adresses sont codées sur 32 bits contre 128 bits pour l'IPv6.

Cela signifie que l'IPv4 à moins de cinq milliards d'adresses possibles. IPv6 a comparativement plus de 7x1028 fois plus d'adresses que l'IPv4.

Aujourd'hui toutes les adresses IPv4 ont été assignées, nous allons voir quels stratagèmes ont été utilisés pour faire en sorte que ce ne soit pas un trop gros problème. Mais il est bon de noter que le futur se trouve dans l'IPv6 et qu'il est donc très important de supporter l'IPv6.

CIDR vs classes

Historiquement, les adresses étaient définies suivant des "classes" allant de A à E. La classe A est équivalente à un masque `/8`, la classe B à un masque `/16`, la classe C à un masque `/24`, la classe D pour le multicast et la classe E pour une utilisation future.

Aujourd'hui, on utilise plus tôt le CIDR (Classless InterDomain Routing) qui consiste à préciser le masque avec un `/<nombre>` à la fin d'une IP (comme on a vu plus tôt). Le CIDR a l'avantage de permettre une configuration des réseaux plus précise et d'être plus simple à comprendre.

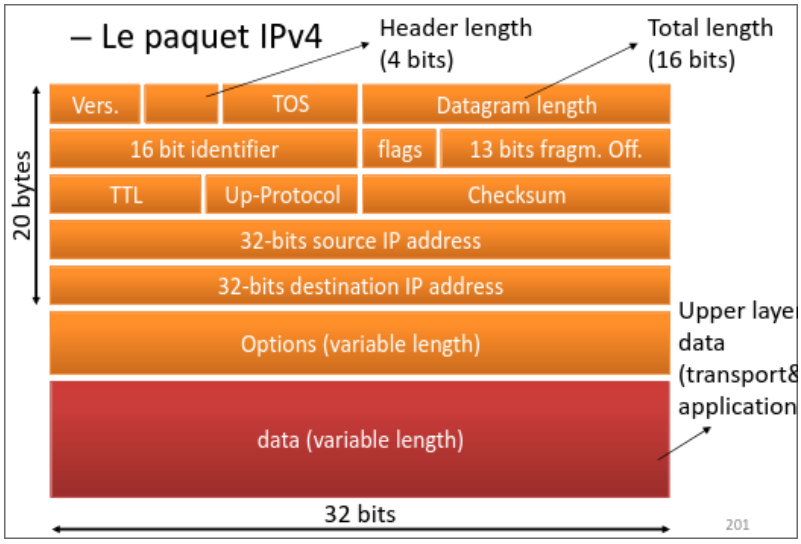
Types d'adresses

Il existe plusieurs types d'adresses :

- Adresses **publiques** qui sont celles utilisées sur Internet et sont achetées à ou allouée par l'IANA (Internet Assigned Number Authority).
- Adresse **loopback** est l'adresse désignant la machine courante, cela permet ainsi d'accéder à un serveur local. Cette adresse est `127.0.0.1` (IPv4) ou `::1` (IPv6) ou `localhost` (qui redirige vers l'adresse IPv4 ou IPv6 dans le fichier d'hôtes).
- Adresses **privées** utilisées sur des réseaux locaux (LAN), elles ne sont pas propagées sur internet et en sont complètement isolées.
- Adresse de **réseau** comme vu plus tôt (préfixe + le reste des bits à 0)
- Adresse de **broadcast** comme vu plus tôt également (préfixe + le reste des bits à 1)

Paquet IPv4

Un paquet IPv4 contient diverses informations :



- La **version** (IPv4 dans ce cas)
- La **longueur d'entête**
- Le "**type of service**" qui indique la priorité du paquet, cette valeur est généralement ignorée
- Le **datagram length** qui indique la longueur totale du paquet sur 16 bits

- L'**identifiant** du paquet
- Des "**flags**" indiquant si le paquet est fragmenté ou non
- Le **fragmentation offset** qui indique le déplacement par rapport au paquet initial
- Le **TTL** (time to live) qui est le temps de survie d'un paquet dans le réseau
- Le **UP-protocol** qui indique le destinataire des données (6 pour TCP, 17 pour UDP)
- Le **Checksum** qui permet de détecter une erreur sur l'entête du paquet, redondant par rapport à TCP
- L'**adresse IP source**
- L'**adresse IP destination**
- Les **options** à ajouter à l'information
- Les données de la couche transport

Fragmentation IPv4

Comme le MSS (Maximum Segment Size) indiquait la taille maximum d'un TPDU de la couche transport, le MTU (Maximum Transert Unit) indique la taille maximale supportée d'un paquet IP, cette taille est imposée par le réseau, donc plusieurs réseaux peuvent avoir des tailles maximales différentes.

Donc si un paquet de longueur 1500 arrive dans un réseau avec un MTU de 600, il va falloir diviser le paquet courant en autres paquets afin de pouvoir le transmettre.

Ainsi le flag permet de savoir si le paquet est fragmenté, l'identifiant permet d'identifier les paquets fragments ensemble. Le fragmentation offset donne l'ordre des fragments. Le dernier fragment a le flag à zéro ce qui indique qu'il n'y a plus de fragments qui suivent.

Eviter les boucles

Pour éviter que des paquets ne bouclent dans le réseau, on utilise la valeur TTL qui indique le temps de vie du paquet en routeur parcouru. Ainsi, la valeur est décrémentée par chaque routeur et si un paquet arrive avec un TTL de un ou inférieur, le paquet est jeté.

ICMPv4

ICMP (Internet Control Message Protocol) permet de "diagnostiquer" certains problèmes réseau via PING et TRACEROUTE.

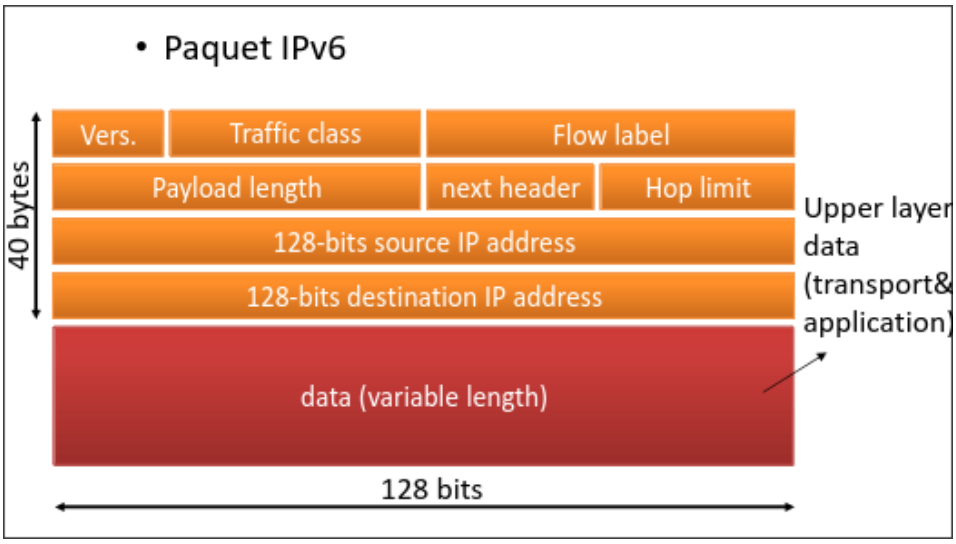
Ping permet de savoir si une machine est capable de recevoir ou répondre au niveau IP, très utile pour vérifier les connexions réseaux, mais souvent limité par les firewalls pour empêcher de pouvoir "tester" les machines du réseau.

Traceroute permet de déterminer le chemin d'une source vers une destination. Cela envoie des paquets IP avec un TTL croissant en partant de 1, qui va donc être rejeté par le routeur qui va ainsi renvoyer un message ICMP `time-exceeded` avec un identifiant du routeur, ce qui permet ainsi de connaitre tous les intermédiaires.

Il est parfois même possible, en utilisant certains sites internet, de retrouver les localisations des routeurs par lequel la requête est passée.

Paquet IPv6

IPv6 comme dit précédemment a pour but d'augmenter le nombre d'adresses disponible, améliorer les performances, intégrer de la sécurité, supporter les nouvelles applications en temps réel, facilité la configuration.



Un paquet IPv6 est composé de :

- La **version** (ici IPv6)
- Le **Traffic Class** permettant de marquer les paquets pour obtenir une *qualité de service particulière*
- **Flow label** permettant d'étiqueter un paquet afin de grouper des paquets faisant partie d'un "flow" commun tel qu'un live vidéo par exemple
- Le **payload length** indiquant la taille des données
- Le **next header** qui mentionne une option à traiter ou pour indiquer la couche supérieure à recevoir des données (UDP ou TCP)
- Le **hop limit** qui indique le total de routeurs que le paquet peut traverser, lorsque cette valeur arrive à 0 le paquet est jeté. Similaire au TTL en IPv4
- L'**adresse IP source**
- L'**adresse IP destination**
- Les données de la couche de transport

Différence avec l'IPv4

Il y a donc certaines différences entre les paquets IPv6 et les paquets IPv4 :

- Le checksum n'existe plus, car redondant par rapport à la couche de transport ou d'accès réseau. Cela permet d'améliorer les performances, parce que les routeurs n'ont plus besoin de vérifier cette valeur
- Disparition des options de taille variable, ces derniers peuvent être placés dans un entête particulière
- Disparition de la fragmentation au niveau des routeurs, IPv6 oblige un MTU de 1280 octets, les paquets peuvent donc être fragmentés. Cela est un grand avantage pour les routeurs qui n'ont plus besoin de fragmenter les paquets, c'est ainsi un gain de performance.

Types d'adresses IPv6

- Adresses **locale-lien** est attachée à l'interface et a une portée limitée au LAN, et ne traverse pas les routeurs. Ce sont des adresses de type `FE80::/10`
- Adresses **locale-unique** sont des adresses attachées à une interface qui peuvent traverser des routeurs mais non utilisable sur internet. Elles sont de type `FC00::/7`
- Adresses **globale-unique** sont des adresses globales, uniques sur Internet et attribuée par le service d'accès Internet. Elles sont de type `2000::/3`
- Adresses **multicast** désignant un groupe de receveurs. Elles sont de type `FF00::/8`

- Adresses **anycast** permettant d'interagir avec n'importe quelle adresse d'un groupe donné (le plus proche selon la politique de routage)

En IPv6, une machine a donc plusieurs adresses, par exemple une adresse globale-unique, une adresse locale-lien et une adresse IPv4 globale.

ICMPv6

ICMPv6 est une évolution d'ICMP pour l'IPv6. Ce protocole permet plusieurs choses :

- Elle permet une gestion des groupes multicast (IGMP en IPv4),
- Retrouver une adresse physique (MAC) en fonction d'une IP (ARP en IPv4),
- Neighbor discovery qui permet de déterminer les adresses local-lien des voisins, routeurs, etc afin de savoir si un voisin est accessible ou non

Elle permet aussi la signalisation des erreurs (destination unreachable, packet too big, time exceeded, parameter problem) ou d'autres messages d'information (ping, gestion de groupes multicast, neighbor discovery).

Neighbor Discovery Protocol (ND)

Le ND est un protocole utilisé pour découvrir les voisins et réaliser de l'auto-configuration des interfaces réseaux (construire l'adresse locale-lien de façon à ce qu'elle soit unique sur le LAN).

Contrairement à DHCP (Dynamic Host Configuration Protocol), qui sert aussi à configurer les hôtes, les adresses ne sont pas distribuées, mais construites. Il est cependant aussi possible de distribuer des adresses en IPv6 en utilisant DHCPv6.

ND prévoit 5 types de messages :

- Sollicitation de routeur, vérification d'un routeur
- Réponse de routeur, un routeur annonce sa présence
- Sollicitation de voisin, vérification d'un voisin
- Réponse de voisin, un voisin indique sa présence
- Redirection, redirige vers un autre routeur

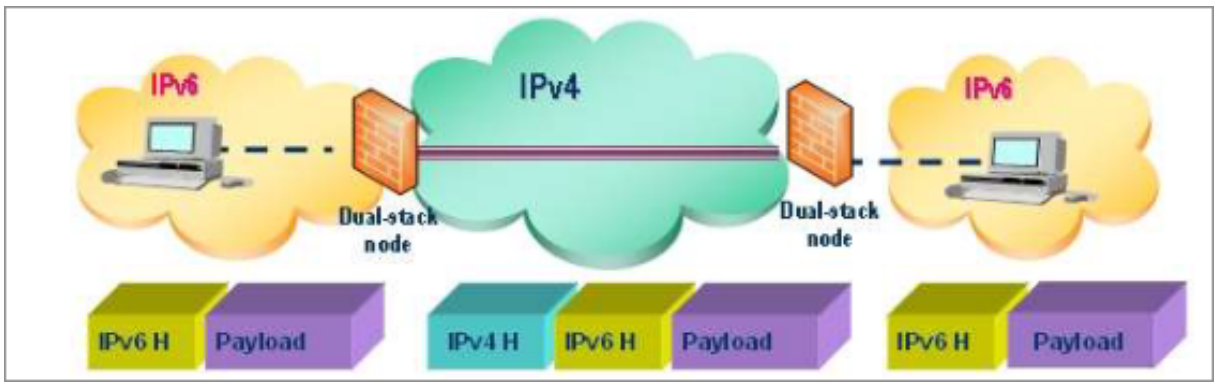
Neighbor solicitation/adverisement permet d'obtenir l'adresse physique (MAC) d'un voisin et de vérifier s'il est accessible.

Transition vers IPv6

La transition vers la version 6 du protocole IP doit être progressive, car il est impossible d'imposer un changement brutal sur plusieurs milliards d'appareils d'un coup. De plus, les couts pour la transition peuvent être importants parce qu'il faut un logiciel et/ou matériel adapté.

Il existe plusieurs dispositifs pour faire une transition vers l'IPv6 graduelle

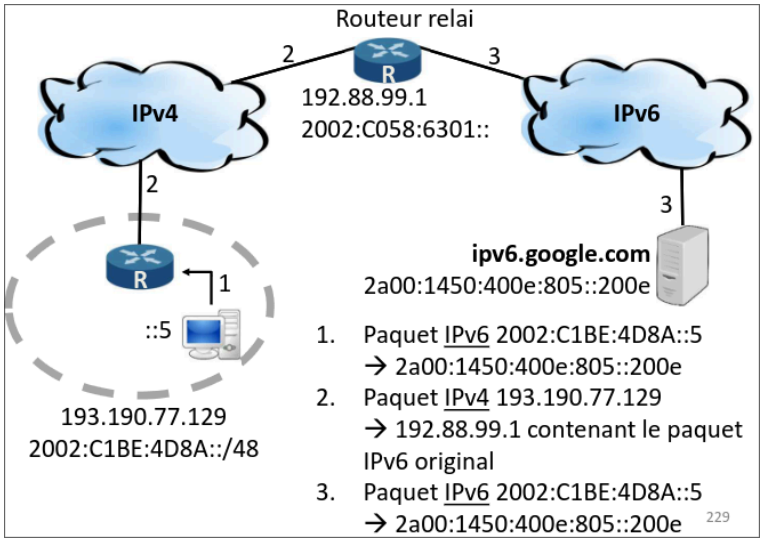
Tunnel-brokers



L'idée ici est de créer un tunnel entre le réseau du client et du fournisseur de service IPv6 (par exemple un site internet). Cela consiste à mettre des paquets IPv6 dans des paquets IPv4.

Un client va donc s'abonner aux services d'un tunnel-broker (par exemple Sixxs), et va envoyer ses paquets IPv6 dans des paquets IPv4 à ce tunnel-broker. Ce dernier va ensuite récupérer les paquets IPv4 pour les envoyer dans le réseau IPv6.

6to4



Une autre idée est d'utiliser des "tunnels automatiques" tel que des routeurs 6to4 qui vont faire le pont entre l'Internet IPv6 et l'Internet IPv4.

Ainsi les paquets IPv6 vont être encapsulés dans des paquets IPv4 et envoyé au routeur relais le plus proche. Ce dernier va ensuite envoyer le paquet IPv6 dans le réseau IPv6 et faire la même chose dans le sens inverse.

Tous les routeurs relais sont identifiés par l'adresse Anycast `192.88.99.1`. Ici le tunnel se fait automatiquement, il n'y a donc pas besoin de s'abonner à un service particulier par exemple.

L'IPv6 d'origine renseignée dans le paquet IPv6 commence toujours par `2002:` pour l'IPv6 et est suivi de l'adresse IPv4 publique. Cela permet ainsi, en utilisant une adresse IPv4 publique, de créer une adresse IPv6 unique.

Le 6to4 a cependant quelques problèmes, par exemple, il est assez difficile de dimensionner les routeurs 6to4 et il est difficile d'assurer une bonne connectivité. Surtout que si un ISP crée un routeur 6to4, il sera également obligé de gérer les clients d'autres ISPs.

Déploiement dual-stack

Le déploiement dual-stack est la méthode de transition la plus "pure" vers l'IPv6 puis ce qu'elle consiste en un fournisseur d'accès Internet qui transforme son infrastructure de manière à supporter nativement à la fois IPv4 et IPv6 (d'où le nom "dual-stack").

Ainsi, le réseau du client, ainsi que le réseau de l'ISP supporte tous les deux l'IPv6 et l'IPv4.

Le problème avec le dual-stack c'est que cela demande souvent une refonte importante de tout le réseau, ce qui n'est parfois pas possible.

6rd

6 Rapid Deployment est une technologie développée par Free. Elle consiste à implémenter une sorte de 6to4 à l'intérieur du réseau de l'ISP.

Ainsi les clients ont des routeurs 6rd qui vont placer les paquets IPv6 dans des paquets IPv4 et va l'envoyer à un routeur 6rd qui va ensuite propager les paquets IPv6 sur le réseau IPv6.

Les machines sont identifiées par des adresses IPv6 locale au réseau de l'ISP, il n'y a donc pas besoin de préfixe comme ce serait le cas pour 6to4 et il n'est ainsi pas nécessaire de modifier tout le réseau existant.

Le 6rd a permis à Free de déployer l'IPv6 à 1.5 million de clients en seulement 5 semaines.

Pallier les problèmes de l'IPv4 en attendant l'IPv6

En attendant que l'IPv6 soit vraiment répandue, il est possible de tout de même palier aux limites d'adressage de l'IPv4.

L'utilisation du CIDR permet par exemple d'affiner les espaces IP adressés et ainsi éviter d'assigner trop d'adresses à une entité.

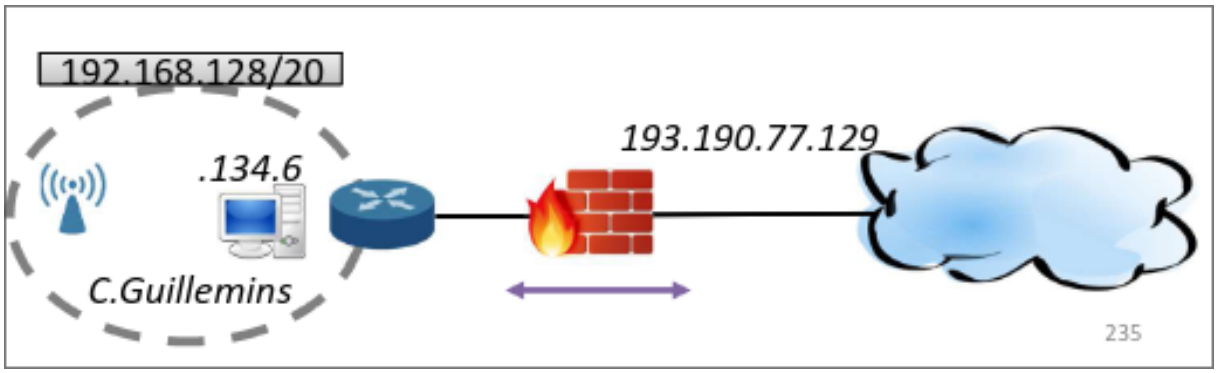
Le **NAT** (Network Address Translator) permet de partager une adresse IP publique entre plusieurs centaines de machines. C'est ce qui est généralement utilisé pour tous les réseaux domestiques.

Le DHCP (Dynamic Host Configuration Protocol) permet de distribuer les adresses au moment de la connexion et ainsi d'en avoir moins pour un certain groupe d'utilisateur.

Fonctionnement du NAT

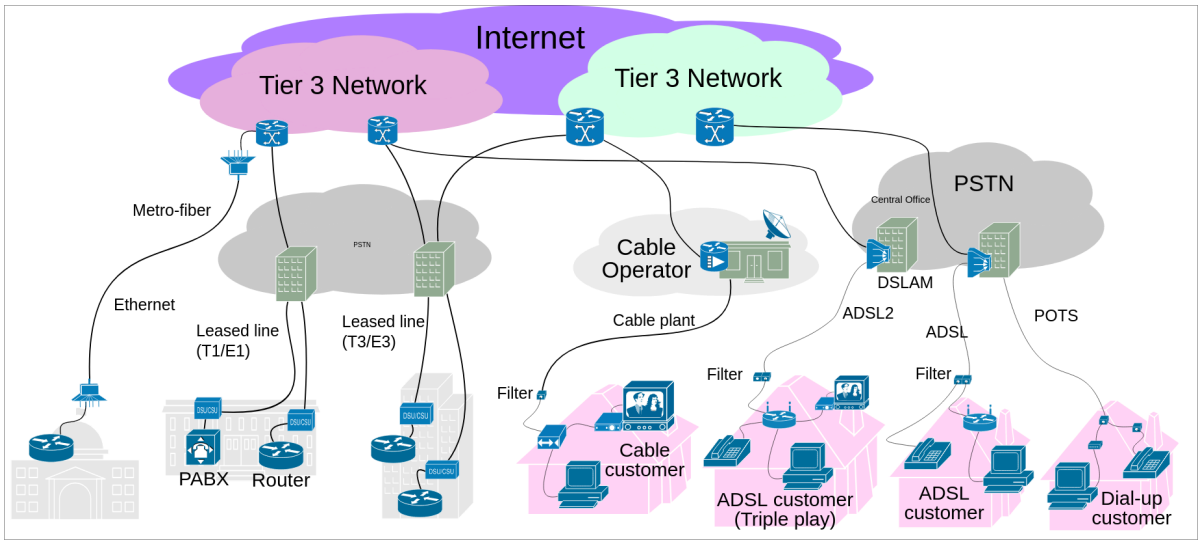
Le NAT fonctionne en modifiant le port et l'IP source des paquets. Ainsi, lorsqu'un paquet veut sortir du réseau local, le NAT va remplacer l'IP locale par l'IP publique, et le port source par un autre. Quand la réponse du serveur arrive, le NAT va regarder quel IP locale et quel port source correspond au port source transformé, va de nouveau modifier le paquet et l'envoyer à la bonne machine du réseau.

Le routeur peut également parfois faire office de firewall.



Routage à grande échelle sur Internet

Internet est un réseau de réseau. Les routeurs permettent de connecter des réseaux ensemble. Evidemment à une échelle telle qu'internet, chaque routeur ne peux pas connaitre tous les autres car il y en aurait beaucoup trop. Surtout qu'en plus il faudrait que chaque routeur sache l'état de tous les autres routeurs du réseau.



Pour résoudre ce problème, on va utiliser plusieurs niveaux différents de routeurs. Les routeurs domestiques font tout transiter vers un routeur plus global du fournisseur d'accès internet.

Ce dernier peut alors communiquer avec les autres routeurs du réseau de l'opérateur, un routeur de l'opérateur connait donc tous les autres routeurs de l'opérateurs, et lorsqu'il doit passer sur un réseau n'appartenant pas à l'opérateur, il doit alors passer par un routeur global. Les relations entre les routeurs d'un même opérateur (**routage intra-domaine**) utilisent les protocoles RIP, IGP ou OSFP (par état lien ou vecteur).

Il y a généralement un routeur global par opérateur (**Autonomous System**), et communiquent avec les autres routeurs de haut-niveau en utilisant le protocole BGP (Border Gateway Protocol). Les autres routeurs de haut niveau ne conaissent pas les routeurs de chaque fournisseur d'accès, et connaissent uniquement les autres routeurs de haut niveau. Les relations entre les routeurs

d'opérateurs différents se font exclusivement via le protocole **BGP (Border Gateway Protocol)**, on parles alors de **routage inter-domaine**.

Chaque opérateur (Autonomous System) est assigné un numéro unique attribué par l'**IANA** ou un de ses délégué (**RIPE** pour l'Europe et l'Asie) et un ensemble de sous-réseaux qu'il peut utiliser. Voici par exemple les informations sur l'AS "BELNET" (AS2611) qui est le FAI d'Helmo. Nous nous trouvons actuellement dans le sous réseau "193.190.0.0/15". Ces informations viennent du site AsnLookup.

193.190.77.129

AS Handle	AS2611
ASN Name	BELNET
Organization Name	BELNET
Organization ID	ORG-BA5-RIPE
Country	Belgium
Regional Registry	RIPE
IPv4 CIDRs	• 5.149.143.0/24 • 134.58.0.0/16 • 139.191.112.0/20 • 146.175.0.0/16 • 185.226.167.0/24 • 192.156.132.0/24 • 193.53.113.0/24 • 193.53.124.0/24 • 193.190.0.0/15 • 77.246.241.0/24 • 134.184.0.0/16 • 143.129.0.0/16 • 157.193.0.0/16 • 192.31.23.0/24 • 193.9.8.0/22 • 193.53.114.0/23 • 193.58.148.0/23 • 109.69.223.0/24 • 138.48.0.0/16 • 143.169.0.0/16 • 164.15.0.0/16 • 192.135.167.0/24 • 193.53.3.0/24 • 193.53.116.0/22 • 193.58.158.0/23 • 130.104.0.0/16 • 139.165.0.0/16 • 144.248.0.0/16 • 185.182.132.0/22 • 192.135.168.0/24 • 193.53.34.0/24 • 193.53.120.0/22 • 193.58.172.0/24
IPv6 CIDRs	• 2001:6a8::/32 • 2a07:9880::/29 • 2a00:b2c0:fade::/48 • 2a10:8280::/32 • 2a02:2c40::/32 • 2a04:b5c0::/29

Chaque routeur est donc associé à un ensemble de préfixes d'addresses (IP d'un réseau + masque), ainsi lorsqu'un paquet arrive à un routeur, ce dernier regarde d'abord si l'addresse de destination fait partie de son réseau, si ce n'est pas le cas le routeur regarde de quel réseau il fait partie et l'envois vers le routeur correspondant au réseau dans sa table de routage. Si aucun réseau ne correspond, le paquet est perdu.

> *Vous pouvez en savoir plus sur le routage en général sur internet en regardant cette vidéo de Computerphile.*

Routage intra-domaine

Le **protocole RIP** fonctionne avec des vecteurs de distance qui sont envoyés toutes les 30 secondes avec un TTL de 1 en multicast (donc ne peut uniquement passer aux routeurs voisins direct). Un vecteur est également envoyé lors d'un changement mais toujours avec un délais de 5 secondes avant la dernier message afin d'éviter de surcharger le réseau (**flapping**). RIP a deux types de messages : les requêtes (pour demander les informations de routage) et les réponses (pour envoyer des informations). Une destination définie dans le protocole RIP est une IP ainsi que le cout pour y accéder.

Le **protocole OSPF** est une implémentation de l'État-Lien avec l'algorithme de Dijkstra vu plus tot. Le routeur envois un message `HELLO` à ses voisins qui échangent ensuite des LSP toutes les 30 minutes et en cas de modification. Un routeur peut intéroger un voisin en envoyant un Link-State Request (LSR).

Comment faire sur de grands réseaux

On peut voir qu'Internet est déjà divisé en plusieurs "niveaux", par exemple celui de l'inter-domaine, de l'intra-domaine, etc. Mais que faire lorsque l'intra-domaine est très grand ?

La solution est de diviser le réseau en "zones" et avoir une zone appellée **backbone** qui va reprendre au minimum un routeur de chaque zone pour les lier tous. Ainsi les routeurs ont juste besoin de connaitre les autres routeurs de leur zone, et les routeurs de backbone ont juste besoin de connaitre les routeurs de leur zone + les autres routeurs backbone.

Voici un exemple de réseau backbone de haut niveau de 1991 aux États Unis.

image("https://upload.wikimedia.org/wikipedia/commons/a/ad/NSFNET-backbone-T1.png")

Routage inter-domaine

Le routage inter-domaine, comme dit précédemment est le routage qui est fait entre plusieurs AS. Chaque AS ne connait pas la carte du réseau d'autres AS pour des raisons politiques.

Le routage inter-domaine utilise le protocole **BGP (Border Gateway Protocol)** qui est un protocole utilisant le port TCP 179 et est basé sur les vecteurs de distance que nous avons vu dans le chapitre précédent.

Avec BGP le problème de **comptage à l'infini** (c'est à dire l'appartition de boucles de paquets, principalement lorsqu'un routeur tombe en panne) est réglé car **BGP liste tous les AS traversés, il est donc possible de détecter des boucles**.

Un routeur BGP communique **uniquement avec ses voisins** (c'est ce que l'on appelle de la connexion point-à-point). Cette connexion avec ses voisin est appellé une **session BGP**. Les routeurs s'échangent ainsi des informatoins concernant des destinations (les préfixes IP) en fonction de leur configuration.

Sur base de leur connaissance du réseau à l'aide des vecteurs de distance, les routeurs établissent quelle zone utiliser pour atteindre chaque destination.

Les routes annoncées aux autres routeurs se font sur base d'une sélection, toutes les routes n'ont pas forcément besoin d'être annoncée comme nous l'avons vu dans la section sur les politiques de peering.

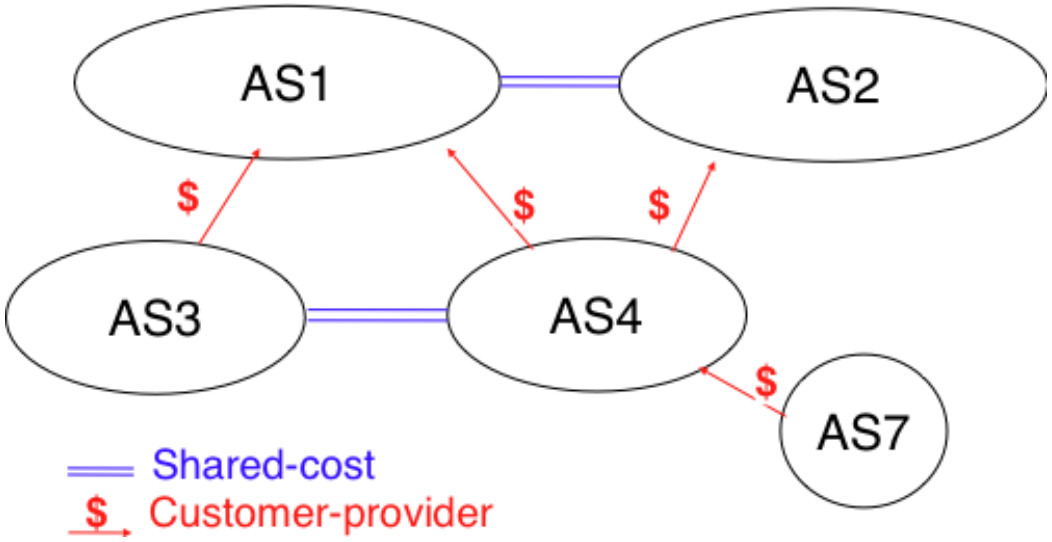
Lorsqu'un routeur BGP reçoit une route, il la prends comme une promesse. Si AS1 partage une route dont la destination se trouve dans l'AS2, il indique donc à tout le monde qu'il promet de contacter l'AS2 pour contacter la/les destination(s) en question.

Il est parfois possible que plusieurs routent mènent à la même destination, BGP ne spécifie pas quoi faire dans ce cas, il est donc du ressort de l'administrateur·ice de s'occuper de configurer le routeur pour résoudre ce problème.

Les routeurs BGP vont ainsi s'envoyer des messages `OPEN` pour établir une connection entre eux, des messages `KEEPALIVE` pour maintenir la connection (qui sont répondu aux messages `OPEN`), des messages `UPDATE` pour annoncer une route/destination ajoutée ou supprimée, et des messages `NOTIFICATION` pour annoncer des problèmes (par exemple une session interrompue).

Politiques de peering

Il existe deux politiques de routage inter-domaine principales :



- Le **customer-provider peering** consiste à simplement payer pour être connecté à un AS. Par exemple dans le schéma AS3 et AS4 payent pour les services d'AS1.
- Le **shared-cost peering** consiste à partager les couts. Par exemple AS3 et AS4 peuvent faire un shared-cost peering pour pouvoir payer moins cher les services d'AS1 en se répartissant les paquets.

Dans le shared-cost peering les AS vont accepter de se partager les informations, ainsi AS1 et AS2 vont se partager leur routes ainsi que celles de leur client.

Alors que dans une relation customer-provider, l'AS ne partage que ses propres routes et pas celle de son fournisseur.

Routage multicast

Ceci est la fin du chapitre du cours de réseau sur Internet. On va ici parler du fonctionnement du multicast IPv4 ainsi que du multicast sur Internet.

L'idée du multicast est simplement de partager une information avec un groupe de receveurs et non pas une seule. Cette idée n'a pas et n'est pas toujours présentes dans tout les réseaux.

Il existe donc plusieurs méthodes pour implémenter le multicast

Types de multicast

Le **one-to-all multicast** est un type de multicast qui fonctionne lorsque le réseau ne supporte pas le multicast. Il consiste simplement à utiliser une connexion séparée pour chaque destinataire. Il y a donc autant de connexions que de destinataires.

L'efficacité est mauvaise mais son implémentation est simple.

L'**application-level multicast** fonctionne également lorsque le réseau ne supporte pas nativement le mutlicast et consiste à transmettre l'information à un destinataire qui va ensuite se charger de la partager au suivants.

L'efficacité est meilleur que le one-to-all mais est compliquée à mettre en place car elle demande une certaine restructuration de l'application.

L'**explicit multicast** fonctionne uniquement si la couche réseau le permet. Il consiste à n'envoyer qu'un seul paquet qui sera répliqué par le routeur et acheminé à tous les destinataires.

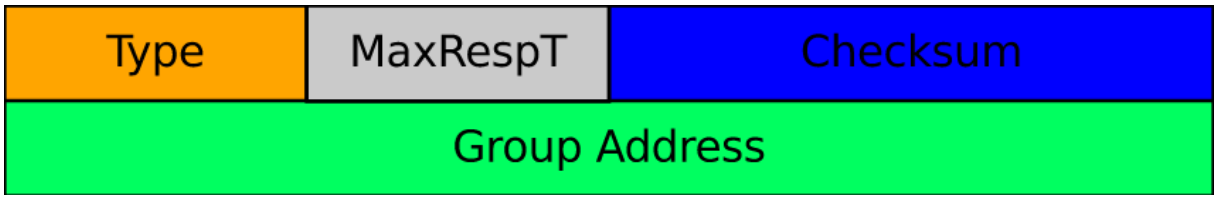
Cette solution est la plus efficace, son seul désavantage est que le réseau doit supporter nativement le multicast pour fonctionner.

Différentier les traffics multicast

Pour pouvoir différentier plusieurs émetteurs et également faire en sorte qu'uniquement les machines intéressées par le multicast ne le reçoit, on pourrait penser que l'on peut "simplement" indiquer toutes les adresses des destinataires dans le paquet. Cependant cela n'est pas pratique, surtout si il y a beaucoup de destinataire. D'autant plus que la source ne connait pas forcément les adresses de ses destinataires.

La solution utilisée est donc plus tot d'utiliser un identifiant appellée une **adresse IP multicast**

Le protocole IGMP



La solution de l'explicit multicast pose cependant certain challenges, comme celui de choisir l'adresse, démarrer ou arrêter le groupe, ajouter une nouvelle machine, vérifier qui peut rejoindre le groupe, etc.

Pour cela on utilise le protocole **IGMPv2 (Internet Group Management Protocol)** qui s'opère entre l'ordinateur et le routeur qui lui est directement connecté. Il offre un moyen pour un hôte d'avertir le routeur qu'il souhaite recevoir un trafic multicast donné et rejoindre le groupe.

Il est donc uniquement utilisé pour signaler au routeur qu'un hote et ou n'est plus intéressé (AKA qu'il est ou non dans le groupe multicast).

Il utilise trois types de messages, les **membership query** envoyés par un routeur à tous les hôtes pour déterminer quels groupes doivent être joints; les **membership report** envoyés par un hôte pour indiquer qu'il s'abonne à un groupe multicast et le **membership leave** qui est envoyé par un hôte pour indiquer qu'il quitte le groupe multicast.

Un message IGMP est composé d'un type, de l'addresse multicast de groupe (qui va de 224.0.0.0 à 239.255.255.255), un champ **Max Response Time** qui indique le temps d'attente maximum pour la réponse et le champ **checksum** qui pemret de contrôler la cohérence des informations.

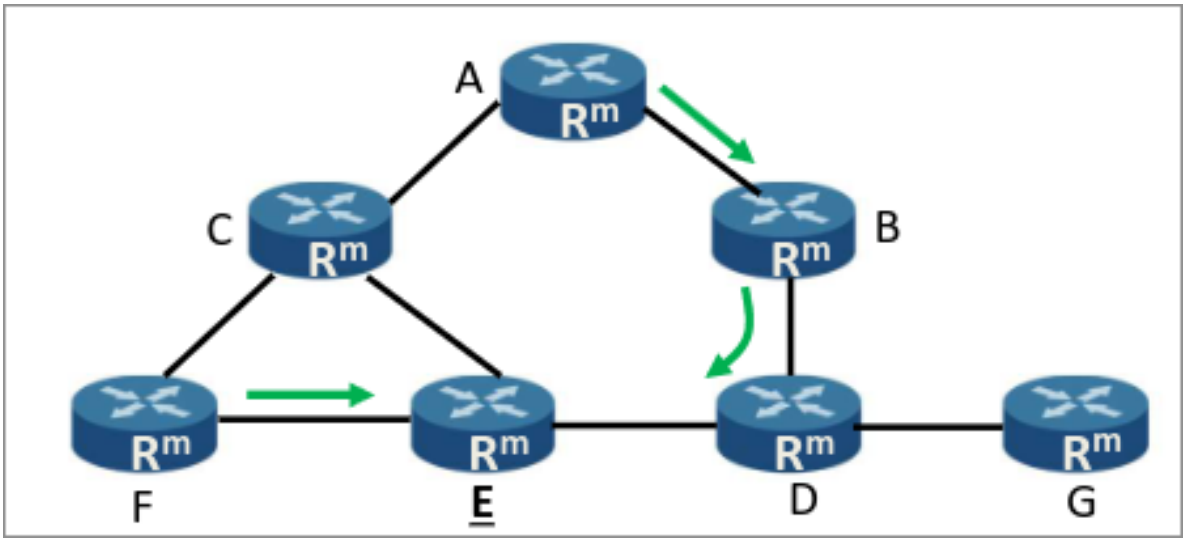
Dans ce modèle, n'importe qui peut s'abonner à un groupe multicast, personne ne connait donc l'ensemble des destinataire et tous les membres peuvent envoyer des informations multicast. Le contrôle d'accès et des informations est donc réalisé par l'application elle-même.

Routage multicast

Si il n'y a qu'un seul routeur, les choses sont simple car le routeur n'a qu'a garder en mémoire la liste des membres du groupe multicast et de retrasférer les informations de cette façon. Cependant les choses deviennent plus complexe lorsqu'il y a plusieurs routeurs.

Une première solution serait que tous les routeurs possèdent une carte complète du réseau (**arbre partagé**) et donc alors de trouver un moyen de relier tous les routeurs intéressés avec un cout minimum. Cela est cependant assez complexe à mettre en place et il n'y a pas de solution dans un temps fini.

Une deuxième solution pourrait être de définir un point de rendez vous. Ce système consiste simplement à choisir un routeur central (par exemple **E** dans le schéma) et à rediriger les messages join vers ce routeur.



Enfin une dernière solution serait d'avoir **un arbre pour chaque membre** en utilisant un système **RPF (Reverse Path Forwarding)**, c'est à dire que lorsqu'un routeur reçoit un paquet multicast qu'il n'a jamais vu, il le retransmet à tout le monde. Le routeur garde en mémoire qui lui a envoyé ce premier paquet et ne retransmettera que les paquets de ce routeur là, les autres seront ignoré. Le routeur peut également indiquer à ses voisins qu'il n'est pas intéressé par le traffic multicast (message **prune**).

L'avantage est ici que le routeur n'a pas besoin d'avoir une carte complète du réseau, simplement de retransmettre le paquet aux suivants.

Routage multicast dans Internet

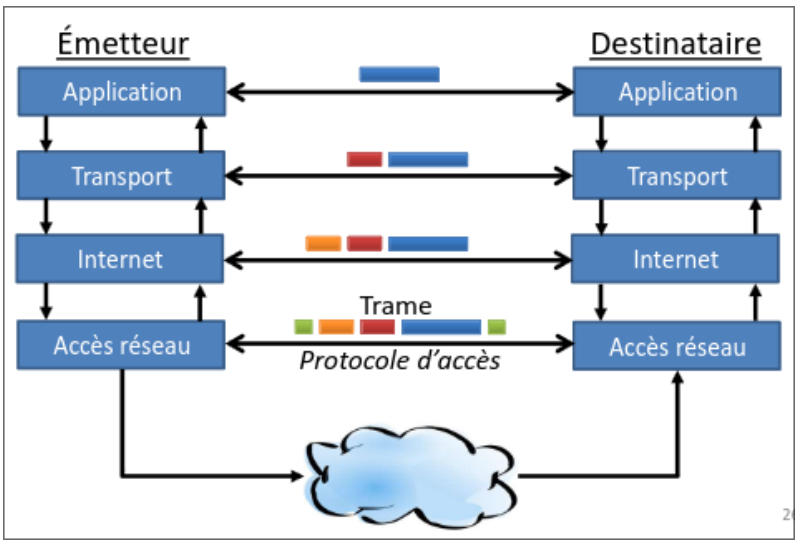
Comme vu précédemment le multicast explicite nécessite que le réseau supporte le multicast, c'est-à-dire que les routeurs soit des routeurs multicast. Donc si un routeur multicast est entouré par des routeurs non-multicast il ne servira à rien. Il est cependant possible d'établir des tunnels (un peu comme pour l'IPv6) qui permettent de transmettre du multicast via de l'unicast.

Différents algorithmes sont prévu pour faire du routage multicast sur Internet, le premier est le **DVMRP (Distance Vector Multicast Routing Protocol)** qui est un algorithme qui implémente un arbre pour chaque membre avec RPF et le pruning et peut également déterminer lorsqu'un traffic ne doit plus lui être propagé en fonction des routeurs qui dépendent de lui.

Un autre algorithme est le **PIM (Protocol Independent Multicast)** qui a deux modes différents, le mode "dense" qui est utilisé lorsqu'il y a beaucoup de routeurs multicast à un endroit, et le mode "sparse" lorsqu'il y a peu de routeurs multicast à un endroit.

Le mode "dense" utilise du RPF (donc un arbre pour chaque membre) alors que mode "sparse" utilisent un système de points de rendez-vous.

Couche d'accès réseau (les trames)



Nous allons maintenant voir la couche de plus bas niveau du réseau qui est celle de l'accès réseau.

La couche d'accès réseau concerne l'échange d'information entre deux machines directement connectées. L'utilité de cette couche est simplement de partager les informations sur la ligne et le format de l'information dépends de la technologie sous-jacente (ethernet, etc)

Délimitation de l'information

Pour pouvoir transmettre les données sur une ligne il est important de délimiter l'information affin d'en connaître sa taille et ainsi déterminer quand la **trame** commence et quand elle se termine.

> *Pour en savoir plus sur la délimitation de l'information, voir cet article.*

Indication de la longueur

Une première méthode serait simplement de garder en mémoire la taille de la trame. Ainsi si on sait par exemple qu'une trame fait 8 bits, on reçoit les 8 bits d'information, ensuite on sait que la trame est finie.

Le problème arrive si une erreur se produit et que la longueur est impactée. Par exemple si la longueur passe de 8 à 10, une partie de la trame suivante va être prise dans la trame actuelle.

Une manière de résoudre cela est d'utiliser des marqueurs de début et de fin en plus. Cela permet ainsi de vérifier que l'on ne prends pas une trame suivante comme donnée.

Character stuffing
Pour cela il y a deux méthodes, le **character stuffing** qui insère un certain charactère comme début et fin. On va ainsi ajouter un octet particulier pour signifier le début ou la fin d'une trame. Ces octets sont présent dans la table ASCII sous le nom de `DLE` (Data Link Escape, qui indique un délimiteur), `STX` (Start-of-Text, qui indique un début de trame), `ETX` (End-of-Text Character, qui indique la fin d'une trame).

Cependant cela pose un problème dans le cas où les données contiennent l'un de ces octets. Pour résoudre ce problème on va utilsier un troisième octet spécial de la table ASCII qui est `ESC` (Escape Character), qui sera placé devant les octets problématique dans les données.

A la reception des données, l'octet après `ESC` sera ignoré. Et bien sûr on peut utiliser ESC pour ignorer ESC lui même.

Dans l'image suivante, le `FLAG` représente `DLE STX` (en début de trame) ou `DLE ETX` (en fin de trame)

Bit stuffing
Une autre méthode est le **bit stuffing** qui fonctionne de la même manière que le character stuffing, mais aux niveaux des bits. Ici `11111` est interpreté comme un début ou fin de trame.

Pour empécher que cela cause des problèmes si la séquence se trouve dans les données, on va la suffixer par un `0` de transparence. Ensuite à la réception toutes les séquences de `111110` seront remplacée par `11111`.

Détection d'erreurs
En plus des marqueurs de début et fin de trame et des données on va également indiquer une donnée supplémentaire pour savoir si l'information a été modifiée ou non, ainsi le destinataire peut vérifier l'intégrité de la donnée reçue.

La performance de détection des erreurs dépends de la méthode utilisée.

- La **parité simple** consiste à simplement garder un bit de parité qui indique si il y a un nombre pair ou impair de bit dans la séquence. Cela signifie cependant que cette méthode ne peut détecter qu'un nombre impair d'erreurs.
- La **parité à deux dimensions** consiste à considérer les données comme un tableau et de garder un bit de partié pour chaque ligne et colonne. Cette méthode a l'avantage de permettre de pouvoir corriger une erreur et en détecter 2.
- Le **CRC (Cyclic Reddancy Check)** consiste à prendre un "code générateur", connu d'avance par les deux parties. Puis d'attacher une valeur aux donnée de sorte que les données soit exactement divisible par le code générateur. Ainsi à la reception, le destinataire a simplement besoin de vérifier que le reste de la division avec le code générateur est bien 0.

Accès au media

L'élément physique qui interconnecte les hôtes peut être soit **partagé**, dans quel cas tous les hôtes partagent le même média et doivent donc mettre en place des règles précises afin d'y accéder dans entrer en collision. C'est par exemple le cas du réseau sans fil, ou du cable coaxial.

Sinon ils peuvent être aussi dédié, où chaque hôte est relié séparément au réseau (par exemple via des cables RJ-45).

Il existe deux types de liaisons, les liaisons point-à-point (un émetteur et un destinataire, par exemple en xDSL) et les lignes boradcast (plusieurs émetteurs et plusieurs destinataires, par exemple en réseau local)

Eviter les collisions

Pour éviter que plusieurs machines parlent en même temps sur la même ligne (écrasant ou en corrompant ainsi les données), il existe 3 grand types de protocoles.

- Les protocoles de **partitionnement du canal** où chaque noeud reçoit exactement une part équitable.
- Les protocoles à **accès aléatoire** où il est possible d'envoyer de l'information n'importe quand et où il y a une possibilité de collision entre deux hôtes qui émettent en même temps.
- Les protocoles **taking-turns** où chaque hôte peut émettre à un omment déterminé et pendant un temps déterminé.

Protocole de partionnement du canal

Ces protocoles divisent la ligne pour que chaque noeud puisse en avoir une partie. Cette division est soite une **TDM (Time Division Multiplexing)**, c'est-à-dire une division où chaque noeud a un espace temps qui lui est réservé.

L'autre division est la division par fréquence ou **FDM (Frequency Division Multiplexing)** qui consiste à diviser le canal en fréquence. Chaque noeud peut alors communiquer continuellement sur sa fréquence sur la ligne.

Cette méthode a l'avantage d'être équitable et de n'avoir aucune collision. Cependant lorsqu'un noeud n'émet rien, la bande passante est perdue et transmet avec un débit moyen de R/N bps (c'est-à-dire le débit de la ligne divisé par le nombre de noeuds).

Protocole à accès aléatoire

Les différents noeuds de la ligne vont "apprendre la politesse", c'est-à-dire qu'un noeud vas donner à tout le monde l'occasion de s'exprimer, ne vas pas parler sans y avoir été invité, va éviter de

monopoliser la conversation, demander avant de faire quelque chose et ne pas interrompre quelqu'un qui parle.

Un exemple de tel mécanisme est celui mis en place par le satellite Slotted-ALOHA permettant de connecter plusieurs îles entre-elles. Pour empécher les collisions, le satellite envois un signal pour indiquer le début d'un "slot", c'est-à-dire un espace temps où il est permit d'émettre, cet espace temps à une durée déterminée.

Cela signifie donc que pour chaque trame (slot), elle peut soit être valide, vide ou en collision. Cela permet donc de limiter le nombre de collision plus facilement.

Un autre exemple de mécanisme comme celui-ci est le CSMA/CD (on y reviendra plus tard) qui va écouter avant de transmettre, donc si quelqu'un transmet déjà, on va attendre qu'iel ait fini.

Protocole taking-turns
Un noeud est désigné comme le noeud maître, celui-ci offre à chacun la possibilité de dialoguer en offrant un certain slot de temps à tout le monde.

Si un noeud n'a rien n'a transmettre on passe au suivant. Cela permet ainsi de disposer de toute la bande passante pour faire la transmission.

Un exemple d'implémentation d'un tel système est d'utiliser un **jeton**, le noeud maître va émettre un jeton, et l'envois à un noeud. Si un noeud n'a rien à transmettre il le réenvois, sinon il envois ses données suivit du jeton. Le jeton fonctionne ainsi un peu comme un baton de parole.

Si ce baton de parole est perdu, le noeud maître peut alors le réenvoyer.

Voici une représentation d'un réseau broadcast de 3 noeuds, un noeud maître et un jeton :

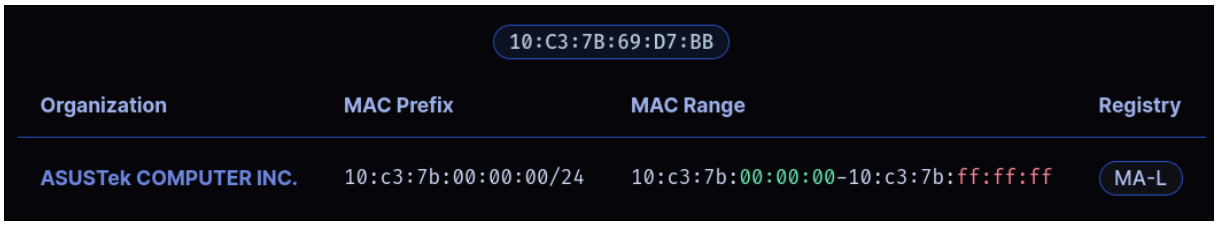
Différentes technolgies d'accès réseau

Dans la section précédente, nous avons vu la base de ce qu'est la couche d'accès réseau, ce qu'est une trame et comment les trames sont gérées. Maintenant nous allons voir comment cela s'intègre avec différentes technologies, dont Ethernet.

Les réseaux locaux ou **LAN (Local Area Network)** sont des réseaux qui interconnecte tous les équipements d'un campus, d'une entreprise, d'une maison, ou autre établissement local. Certains réseaux comportent un ou plusieurs routerus qui sont connectés à Internet. Il existe plusieurs norme de transfert, Ethernet, Wifi ou Tokenring.

Dans les chapitres précédents on a vu que l'on identifie des machines par adresse IP, cependant comment faire au niveau de la couche d'accès ? Cela se fait avec des **adresses physiques (MAC)**. Les addresses MAC sont unique et sont codées sur 48 bits dont une partie fixée par le vendeur et une autre fixée par l'IEEE.

Par exemple on peut voir sur le site oui.is que le préfixe `10:c3:7b` correspond à ASUS et les derniers nombres sont un identifiant unique donné par le vendeur.



Ainsi lorsqu'un noeud reçoit des informations, il vérifie que les trames correspondent à son adresse MAC, sinon il les ignores.

Protocole ARP (IPv4)

Il y a cependant un petit soucis, c'est que sur Internet, on utilise des adresses IP (et le port pour la couche de transport) et non pas des trames pour identifier des services. Donc comment faire pour savoir quel est l'adresse MAC (adresse physique) destination d'une certaine adresse ? Par exemple si une machine 1 souhaite connaître l'adresse physique de `192.168.0.5`.

Au sein d'un même sous réseau

Tout d'abord la machine va envoyer un message en broadcast (c'est-à-dire à tout le monde) demandant "qui est 192.168.0.5 ?"

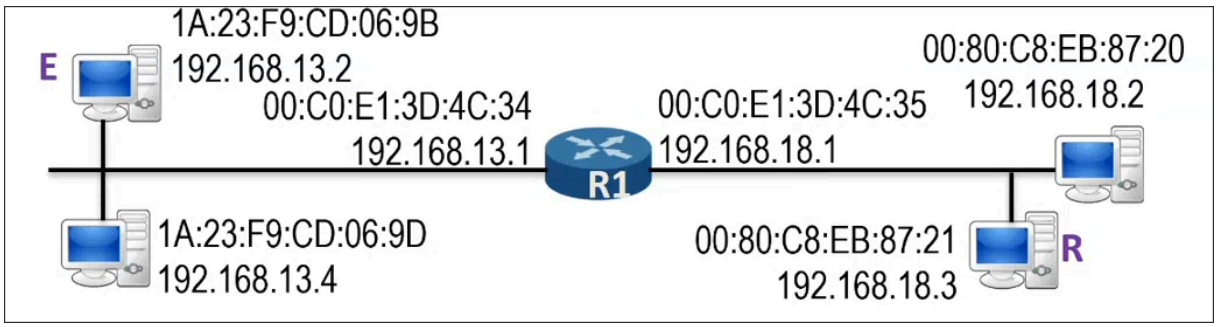
Ensuite la machine qui se reconnait va répondre "Je suis 192.168.0.5 et mon adresse physique est `AA:BB:AA:55:55:55`"

Ensuite le PC 1 peut retenir l'information (dans sa table ARP, disponible en faisant `arp -a` sous Linux ou Windows) afin de ne pas avoir à le redemander plus tard. Elle peut ensuite tranquillement communiquer avec le PC 5 en encapsulant son paquet IP dans sa trame en précisant l'adresse MAC du PC 5.

Ce système est appellé **ARP (Address Resolution Protocol)** et est exclusivement utilisé au sein d'un sous-réseau.

Et lorsqu'il y a plusieurs sous-réseaux ?

Lorsqu'il y a plusieurs sous-réseau avec un routeur entre les deux, le système fonctionne de manière similaire. Si la machine E souhaite communiquer avec la machine R par exemple :



Elle va d'abord demander avec ARP l'adresse physique de R1 sur base de son adresse IP. Ensuite il pourra lui communiquer les données à envoyer à R.

R1 va alors récupérer l'addresse IP précisée dans le paquet qui est dans sa trame. Ensuite il va demander en broadcast l'adresse physique de R et sur base de cette adresse, il va pouvoir lui transmettre les données qui lui sont destinées.

En somme c'est la même chose qu'avant, sauf qu'on le fait deux fois, une première fois entre l'émetteur et le routeur, et une deuxième fois entre le routeur et la destination.

Le protocole NDP (IPv6)

L'ARP que nous avons vu plus tot est un protocole qui a été créé pour l'IPv4, pour l'IPv6 quelque chose d'autre a été imaginé. En effet le système d'ARP bien que fonctionnel est assez complexe à maintenir car il faut à chaque fois émettre la demande à tout le monde pour savoir qui correspond. En IPv6 on utilise le **Neighbor Discovery Protocol (NDP)**.

Lorsqu'une machine se connecte sur le réseau elle va envoyer un message de **Router Discovery** en multicast à destination des routeurs. Le routeur va ensuite répondre avec un message de **Router Advertisement** indiquant son adresse physique (qui est également envoyé périodiquement sur le réseau).

Ensuite pour connaître un autre noeud sur le réseau, on envois un message de **Neighbor Discovery** de manière similaire à ARP sauf que c'est du multicast et non du broadcast. La machine va ensuite répondre avec un message de **Neighbor Advertisement**. Il existe encore un message supplémentaire qui est celui du **Redirect** qui est envoyé par un routeur pour indiquer qu'il faut préférer un autre routeur.

Ce système est plus performant qu'ARP car il utilise du multicast et non du broadcast. Cela signifie par exemple que seul les routeurs recevront les messages de *Router Discovery* et que seul les autres machines reçevront les messages de *Neighbor Discovery*

Pour en savoir plus sur le NDP n'hésitez pas à voir cette vidéo.

Ethernet

Ethernet est une technologie qui a plus de 50 ans mais qui est l'une des plus populaire dans le monde nottament grâce à sa grande simplicité et au fait qu'il est bon marché.

Ethernet est un service sans connexion (pas de séquence de connexion à faire pour l'utiliser) et non fiable (des données peuvent être perdues).

Ethernet permet également de faire des réseaux en étoile, il s'agit de connecter les appareils à un switch qui s'occupe alors de retransmettre les informations (chaque appareil a donc une connexion dédiée vers le switch).

La trame Ethernet

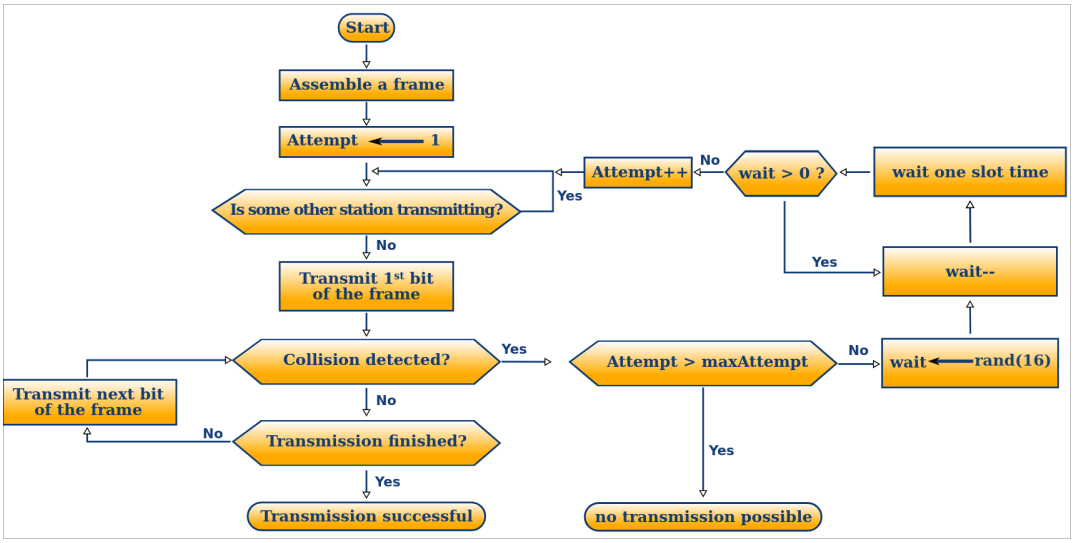
Une trame Ethernet est composée principalement de :

- Un **préambule** qui indique des informations relatives à la synchronisation de l'heure
- Un **délimiteur de début de trame** et un **délimiteur de fin de trame** (comme on a vu précédemment pour délimiter une trame)
- L'adresse MAC **source** et **destination**
- Le **type** (indiquant le type de donnée contenue, par exemple : IPv4, IPv6, etc)
- Les **données** (c'est-à-dire le paquet IP)
- Un **code CRC** (ne portant pas sur le préambule) permettant de vérifier l'intégrité de la trame.

CSMA/CD

Ethernet est une ligne broadcast, cela signifie qu'il faut un système pour contrôler l'accès. Dans le cas d'Ethernet, ce système est le CSMA/CD. Avec ce système avant de transmettre une information on écoute le canal et si une transmission est en cours on attent.

Une fois que le canal est libre on transmet les informations tout en écoutant pour voir si on détecte des collisions, si oui, on envois un message de colllision et on attends une durée aléatoire. Et au bout d'un certain nombre de tentative, la transmission est abandonnée.



Ce mécanisme peut également utiliser l'**exponential back-off** qui consiste à attendre plus longtemps en fonction du nombre de collisions.

Topologie

Un réseau Ethernet se fait généralement avec des connexions point à point entre le relai (switch) et les hôtes (les machines) via des connecteurs RJ45.

Le switch peut être intelligent ou *manageable* ce qui signifie que l'on peut le configurer plus finement. Il peut également être compatible SDN (Software Defined Network).

Un **SDN** est un moyen de découpler l'équipement de la configuration et ainsi avoir une configuration générique qui fonctionne quelque soit l'équipement et permet de faire tout un tas de configuration (équilibrage des charges, optimisation, qualité de service, routage intelligent, etc)

Multicast Ethernet IPv4
Pour pouvoir transmettre en multicast via Ethernet, il faut pouvoir créer une adresse MAC multicast. Celle-ci se compose des 23 derniers bits de l'adresse IP, un identificateur fixe indiquant qu'il s'agit d'une adresse multicast dans les 24 premiers bits (`01-00-5E`).

Par exemple pour transformer `224.0.9.45` en adresse MAC multicast, on prends les 23 derniers bits (ce qui donne en hexadécimal `00-09-2D`) et on ajoute l'identifiant devant, ce qui donne alors l'addresse multicast `01:00:5E:00:09:2D`.

Le problème avec l'IPv4 ici c'est que l'on perds les premiers bits de l'IP, ce qui signifie qu'il peut y avoir une même adresse MAC pour plusieurs adresses IP multicast.

Multicast Ethernet IPv6
Le mécanisme est semblable à celui pour l'IPv4, on copie ici les 32 derniers bits préfixé ensuite par l'identificateur `33-33`.

Ainsi l'adresse `ff02:0:0:0:1:fff9:2973` devient `33:33:ff:f9:29:73`.

Interconnexions (switchs et hub)
Un hub est un relai au niveau physique, c'est un modèle assez rare, risqué et peu pratique car il ne fait simplement que simplement reproduire tous les messages sur tous ses ports sans vérification, ce qui peut donc mener à des collisions.

Aujourd'hui on utilise plus tôt des **switch** qui va analyser une trame et la propager uniquement à la destination. Le switch maintient une table indiquant pour chaque port, les adresses MAC qui y sont accessible.

Tout comme n'importe quel appareil sur le réseau il implémente CSMA/CD pour contrer les collisions. Puis ce qu'il peut faire plus de traitement il peut même être configuré ou interconnecter des technologies différentes en traduisant les différents protocoles.

Un type de configuration particulier qui peut être fait sur un switch sont des **VLAN (Virual Local Area Network)** qui sert à découper le switch pour créer des réseaux différents sans avoir besoin d'acheter plusieurs switch. Deplus dans un VLAN un même port peut être dans plusieurs VLAN simultanément en étiquettant les trames.

Spanning Tree Protocol (STP)
Les switch peuvent être connecté entre eux pour former des réseaux plus grand, cela pose cependant un problème car comment faire pour éviter que des paquets ne bouclent entre plusieurs switch ?

Pour ce faire on met en place un système appellé "**Spanning Tree Protocol (ou STP)**", ou en français "arbre de recouvrement minimum".

Ce système va utiliser des appareils appeller des **ponts (network bridges)** qui vont permettre de lier plusieurs switch entre eux. Ainsi chaque pont va être identifié par un numéro, le numéro le plus petit sera considéré comme la **racine** et chaque pont va émettre des **BPDU (Bridge Protocol Data**

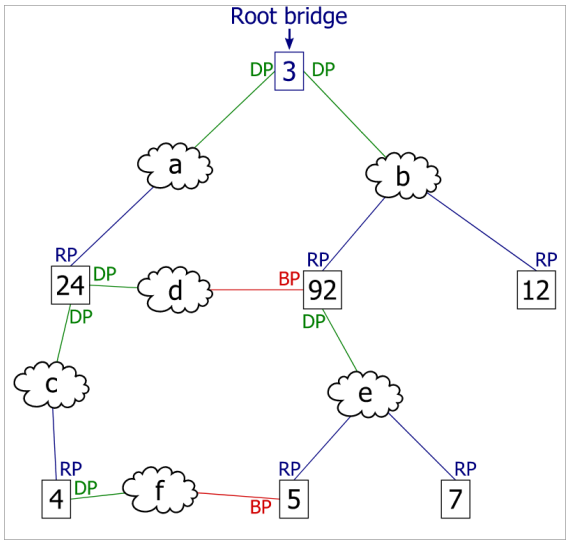
Unit) vers les autres ponts. Chaque pont va donc envoyer des BPDU contenant leur identifiant, l'identifiant de la racine et leur cout pour atteindre la racine.

Par exemple :

1. Lorsque l'on connecte un switch (identifiant 3), il commence par se considérer par la racine, il va donc envoyer un message indiquant "Je suis 3, mon cout pour atteindre la racine est 0 et la racine est 3"
2. Ensuite lorsqu'il reçoit un autre message, par exemple un message disant "Je suis 2, mon cout pour atteindre la racine est 1, la racine est 0", il se rend compte qu'il n'est pas la racine, il va donc se mettre à jour en conséquence et va envoyer "Je suis 3, le cout pour atteindre la racine est 2 et la racine est 0"
3. Le switch va à chaque fois se souvenir de quel port lui donne le meilleur BPDU et va adapter le cout indiquer de son BPDU en conséquence.
4. La racine envois des BPDU quoi qu'il arrive, les autres switch eux n'en envois que pour mettre à jour les informations. Une fois que la racine est le seul émetteur, cela veut dire que la phase de synchronisation est finie
5. Chaque switch va alors définir un port racine (qui est le port qui reçoit les BPDU de la racine), des ports désignés (si les messages sont moins bon, cout supérieur) à ceux envoyés par le switch afin de propager l'info de ceux en bas de l'arbre et des ports bloqués si le message est melilleur que celui du switch actuel (les informations seront donc ignorées).

En bloquant ainsi des ports, on peut donc éviter de créer des boucles dans le réseau. En cas de panne, le pont qui détecte la panne peut alors émettre un BPDU mettant à jour le cout et ainsi recalculer un nouveau chemin.

A la fin d'une séquence STP, on se retrouve donc avec un réseau tel que celui-ci :



Les lettres dans des nuages sont des switch tandis que les numéros dans des carrés sont des ponts. `DP` signifie "Designated Port", `BP` signifie "Blocked Port" et `RP` signifie "Root Port".

PPP - Point to Point Protocol

Le PPP est un protocole utilisé entre un émetteur et un destinataire, il est donc une sorte d'alternative à Ethernet à utiliser lorsqu'il n'y a que deux noeuds. Par exemple entre un client d'un ISP et l'ISP ou entre deux routeurs de haut niveau.

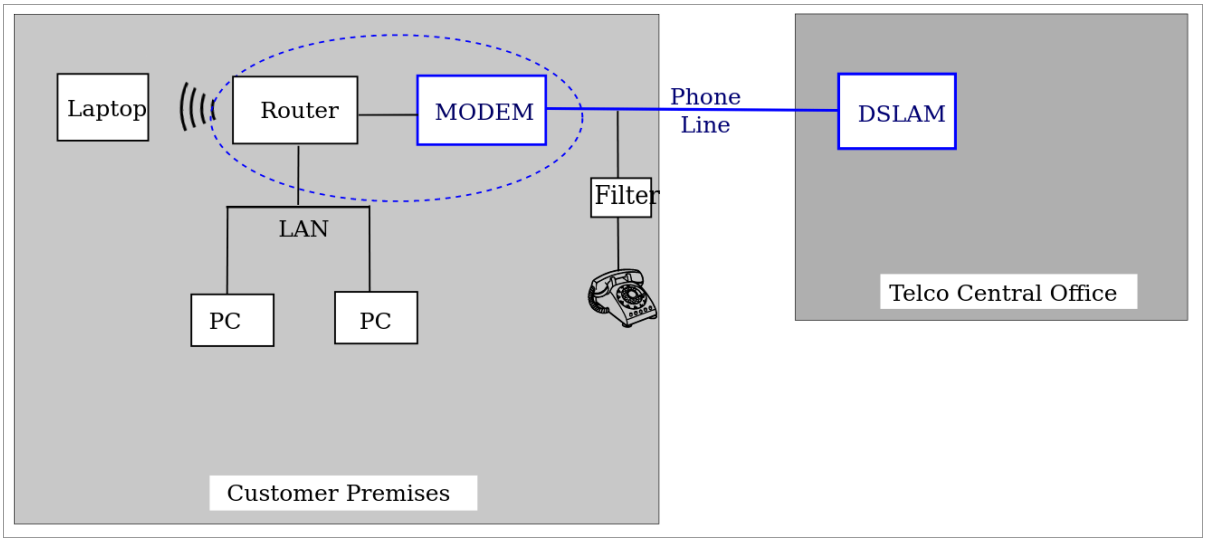
C'est nottament très utilisé pour les connection xDSL via PPPoE (PPP over Ethernet) ou PPPoA (PPP over Asynchronous Transfer Mode).

Une trame PPP se compose d'un **flag** au début et à la fin (pour délimiter la trame via du character stuffing comme vu dans la section précédente), un champ "**address**" et un champ "**control**" qui peuvent être utilisés pour signifier la source et la destination mais puis ce qu'il n'y a que deux noeuds la destination (address) est toujours mise à l'adresse broadcast `11111111` et `00000011` pour l'adressed de contrôle. La trame comprends également le **protocole** (IP, etc), les données contenue dans la trame et un **checksum CRC** pour détecter les erreurs.

Donc par exemple dans une connexion entre un PC chez un particulier et le routeur de l'ISP via une connection xDSL. La connexion du routeur du particulier va être passée par un modem pour pouvoir fonctionner sur la ligne téléphonique. Les données vont arrivée à un point "DSALM" qui est un multiplexeur, c'est-à-dire qu'il va combiner tous les signaux pour les mettres sur une seule ligne qui va ensuite être dirigée soit vers un autre multiplexeur, soit vers le routeur.

Les trames qui sont envoyée depuis le routeur à l'autre routeur, elles sont en PPP et ont donc toute pour destination "l'autre bout de la ligne" ce qui fait qu'il est impossible de se connecter à votre voisin de rue sans passer par le routeur par exemple. C'est ce qui fait qu'une rue ne représente pas un LAN.

Par ailleurs, petit détail, dans le cas du DSL il y a également un filtre qui permet d'éviter les collisions entre les données réseau et les données téléphoniques analogiques.



Sous-protocoles de PPP

Un premier sous-protocole de PPP est le **LCP (Link-Control Protocol)** qui s'occupe d'établir la liaison et négocier les options, authentifier les utilisateurs si nécessaire, maintenir l'activité et fermer la connexion.

Afin d'authentifier les utilisateur·ice·s on utilise MS-CHAPv2, cela permet ainsi que seul les clients de l'ISP puissent se connecter sur la ligne.

Un autre protocole est le **NCP (Network Control Protocol)** qui est une famille de protocoles qui permettent de configurer les couches supérieures (configuration IP, DNS, etc)

Sécurité réseau

La sécurité est une préoccupation assez récente, beaucoup d'applications n'étaient pas sécurisées avant (POP3, Telnet, FTP, IMAP, etc). L'ajout de la sécurité a été fait par après car il était impossible de faire migrer tout le monde vers d'autres solutions. Mais il est important que la sécurité soit déjà présente au coeur même de chaque application.

Surtout que la mise en réseau n'arrange rien car il y a beaucoup plus de possibilités d'attaques, surtout qu'une très grande proportion du traffic internet concerne des tentatives d'attaque.

Qu'est ce que la sécurité réseau et comment ?

La sécurité réseau recouvre un ensemble de domaine tel que la **confidentialité des informations** (grâce à du chiffrement, de façon à ce que l'information soit uniquement compréhensible par l'émetteur et le récepteur), l'**authentification des entités** (vérification de l'identité, encore une fois grâce à du chiffrement), l'**intégrité des messages** (détection de la modification d'un message, ne pas pouvoir le renier, encore une fois grâce aux hash cryptographiques), et enfin la **disponibilité et le contrôle d'accès** (protéger contre les (D)DoS, et contrôler l'accès aux ressources de l'application).

- Il faut donc utiliser le **chiffrement PARTOUT** pour permettre de garantir toutes ces conditions.
- Il faut également utiliser un **pare-feu** qui va limiter et contrôler les accès à une machine ou à un réseau avec un ensemble de règles (en fonction des ports, du type de traffic, des IP, etc).
- Il faut aussi **opter pour une solution centralisée** pour mieux pouvoir monitorer les serveurs.
- Il faut pouvoir être **pret à réagir**, pouvoir monitorer les attaques et en cas extrême de pouvoir restaurer les backups.
- Il faut aussi **rester informer** sur les nouvelles à propos de la sécurité afin de pouvoir réagir rapidement et préventivement
- Enfin, il faut **mettre à jour les applications** afin d'éviter les bugs et bénéficier des améliorations de sécurité. Il faut cependant faire attention à ce que les mises à jour n'impacte pas les données de l'application, et n'intègre pas de nouvelles vulnérabilités.

Le chiffrement

Il est indispensable d'utiliser du chiffrement asymétrique car tout autre méthode pourrait être interceptée et rejouée.

On va alors utiliser un système de "challenge cryptographique", une fois que l'utilisateur·ice veut se connecter, le serveur va lui demander de signer/chiffrer un texte aléatoire avec sa clé privée. Le serveur demande alors quel est la clé publique, et l'utilisateur·ice envois sa clé publique.

Ce système permet de faire un échange sécurisé dans la majorité des cas, sauf dans le cas d'un **man-in-the-middle attack**.

Ici, Eve intercepte les données et retourne sa clé publique à la place de celle d'Alice, elle est donc authentifiée et peut intercepter le traffic sans qu'Alice ne s'en rendent compte.

Contre cela on utilise donc des systèmes de tiers de confiance. Il y a deux systèmes de tiers de confiance principaux les **KDC (Key Distribution Center)** (utilisées pour les systèmes à clé secrète, c'est-à-dire de la cryptographie symmétrique ou la même clé peut être utilisé pour chiffrer et déchiffrer) et les **autorités de certifications** (qui est utilisée dans les systèmes à clé publique-privée, c'est-à-dire lorsqu'il y a une clé différente pour chiffrer et déchiffrer).

KDC (Key Distribution Center)
Dans le système du KDC, on a un serveur au quel chaque membre souhaitant communiquer donne une clé secrète.

Lorsque deux machines veulent communiquer, l'une demande au serveur en chiffrant sa demande, de se connecter à l'autre.

Ensuite le serveur génère une clé de session et la même clé de session chiffrée avec la clé secrète du destinataire.

Ensuite on envois cette clé chiffrée au destinatire afin que tout le monde aie connaissance de la clé de session. La communication est à présent établie.

Autorités de certifications
Le KDC possède un gros problème car il est central, il connait les clés secrète de tout le monde (donnée très sensible) et la clé ne peut pas à l'origine être envoyée au serveur de manière sécurisée.

Alors pour sécuriser sur Internet le protocole le plus utilisé est **SSL/TLS (Secure Server Socket et Transport Layer Security)**. TLS représente les dernières versions de SSL.

Dans ce système on possède une **autorité de certification** qui va permettre de vérifier l'identité d'une personne en la signant avec son propre certificat.

Dans ce système l'administrateur·ice système va générer une clé privée et une clé publique. Iel va ensuite passer cette clé publique à l'autorité de certification qui va faire certaines verifications dont au minimum vérifier par un challenge que le client possède bien la clé privée qui va avec la clé publique.

Si tout est correct, l'autorité de certification va signer la clé publique en disant "Je confirme que helmo.be a bien la clé suivante".

Ensuite lorsque le navigateur va charger la page, il va vérifier si le certificat du site est bien signé par l'une des autorités de certifications qu'il connait, si oui il vérifie que le certificat n'est pas expiré. Si tout est bon, la page peut alors être affichée.

Les navigateurs ont donc une liste d'autorités de certifications auquel ils font confiance, mais il faut aussi savoir qu'il peut y avoir une hierarchie d'autorité de certification.

Les navigateurs ont ainsi seulement besoin de connaître les autorités racine, pour pouvoir vérifier les certificats qui ont été autorisé par les autorités approuvées par les autorités racines.

Les vérifications qui sont faites par les autorités de certifications varient, certains vont vérifier les données légales de l'entité qui demande la certification, d'autres vont simplement vérifier qu'elle a bien la clé privée (c'est nottament le cas de *Let's Encrypt* qui est une autorité de certification gratuite et qui ne fait aucune vérification légale).

Un site sécurisé par Let's Encrypt ne signifie pas pour autant qu'il n'est pas fiable bien que beaucoup de sites de phishing l'utilise pour ne pas avoir à révèler leur identité. Let's Encrypt est surtout majoritairement utilisé pour des fins parfaitement légitimes.

Fonctionnement de SSL/TLS
La plus pars des sites Internet sont sécurisé à l'aide de l'un de ces protocoles. Lorsque Alice va sur le site de Lea, Lea va envoyer son certificat, Alice va ensuite vérifier qu'il est valide en le vérifiant, ensuite Alice va générer une clé de session qu'elle va communiquer à Lea en le chiffrant avec sa clé publique.

Lea et Alice ont donc maitenant la clé de session qui leur permette de communiquer. On utilise alors la clé de session secrète car elle est plus rapide à utiliser pour chiffrer les données que de tout faire avec la clé publique.

A savoir que ce n'est pas par ce que le site est marqué comme sécurisé que ce n'est pas une arnaque. Si vous allez sur un site de fishing marqué comme sécurisé cela veut seulement dire que personne d'autre que vous et l'anarqueur·euse n'aura accès à vos données.

Contrôle d'accès (firewall et proxy)

Il est très important de chiffrer le canal, mais il faut aussi faire attention que n'importe qui ne puisse pas y entrer. Cela se fait via des mécanismes de firewall ou de proxy. Mais il faut évidemment que l'application soit elle-même bien configurée pour empécher n'importe qui d'accéder au panel admin.

Pour ce qui est du **pare-feu (firewall)**, cela consiste à isoler des éléments du reste du réseau. Par exemple en mettant en place un DMZ (demilitarized zone) qui consiste à faire en sorte que seule une machine du réseau soit accessible sur le réseau, ou encore en mettant en place un ensemble de règles sur le traffic entrant et/ou sortant.

Par exemple on peut définir que le traffic sortant est illimité mais que seul le port 80, 443 et 22 d'une machine particulière sont accessible depuis internet (traffic entrant).

Mais il est aussi possible de filtrer avec plus de finesse par exemple en utilisant des outils tel que Fail2Ban (Stateful inspection) qui va filtrer les paquets pour interdire l'accès à des adresses IP ayant fait trop de tentatives.

Ou encore CrowdSec qui consiste à créer une base de donnée d'adresses IP d'attaquant partagée afin de pouvoir plus facilement détecter les adresses IP malicieuses.

Ou encore des NGFW (Next Generation Firewall) qui permet d'identifier les applications, le type de site, le type de traffic, etc. Ces derniers fonctionnent en catégorisant un maximum de site internet et en créant des règles sur base de ces catégories.

Enfin pour pouvoir centraliser tout le traffic à un endroit on peut utiliser un proxy qui va traiter les requètes en amont. C'est par exemple le cas de Cloudflare qui va protéger la véritable adresse IP des sites internet et qui va nottament protéger les sites contre les attaques DoS et DDoS.

Une bonne règle générale avec les firewall est de faire les choses sur base du motto "*Tout ce qui n'est pas explicitement autorisé est interdit*".

IPSec (IP Security)

Le concept de l'IPSec est de sécuriser la couche réseau elle-même plus tot que de sécuriser la couche application.

Cela consiste à chiffrer les données des paquets d'une manière ou d'une autre (clé publique/privée, clé secrète ou clé de session). Bien-sûr tout le paquet n'est pas chiffré, seul les données le sont, sinon les routeurs ne saurait pas à qui donner le paquet.

Un autre avantage de l'IPSec est qu'il est impossible de se faire passer pour quelqu'un d'autre (IP Spoofing, c'est-à-dire définir comme adresse source d'un paquet, une adresse différente de la sienne pour se faire passer pour quelqu'un d'autre, ce qui est la raison pour laquel l'IP n'est pas une bonne source d'authentification).

L'IPSec utilise deux protocoles particuliers le premier est **le protocole AH (Authentication Header)** qui permet d'authentifier la source et garanti l'intégrité des données, mais cela *sans chiffrement*.

Ensuite il y a le protocole **ESP (Encapsulated Security Payload)** qui fournit l'authentification, l'intégrité *et* la confidentialité via le chiffrement.

Contrairement à la couche réseau IP traditionelle qui n'a pas de connexions, il faut faire une sorte de connexion en IPSec. Cette connexion est appellée **association sécurisée (SA)** et est nuidirectionnelle. Cette connexion identifie le protocole à utiliser (AH ou ESP), l'adresse IP source (et uniquement la source puis ce que c'est unidirectionnel), et un identifiant de la connexion sur 32 bits appellé *Security Parameter Index (SPI)*. Tous les paquets d'une même association sécurisée auront le même SPI.

Le protocole AH
Comme dit plus tot ce protocole fournit une authentification et un contrôle des données mais sans confidentialité.

Il va insèrer une entête intermédiaire entre l'entête IP et les données du paquet. Et va utiliser un numéro de protocole particulier (51) pour indiquer que AH est utilisé.

L'entête AH contient, un champ "next header" pour indiquer la couche de transport à utiliser, le champ SPI pour identifier la connexion, le numéro de séquence pour éviter les attaques par rejeu (un attanquant qui intercepterait des paquets chiffrer pour les réenvoyer) ainsi qu'un chmap authentification qui assure l'intégrité des données.

Protocole ESP
Le protocole assure l'intégrité, l'authentification et la confidentialité des données.

On commence par établir une SA (connexion initiale) pour que la source puisse envoyer des messages de manière sécurisée à la destination.

Les données sont alors encapsulées dans une structure ESP où tout le contenu est chiffré à l'exception de l'entête qui contient les informations pour la destination. Les données sot alors chiffrées avec un algorithme choisis entre les différentes entités.

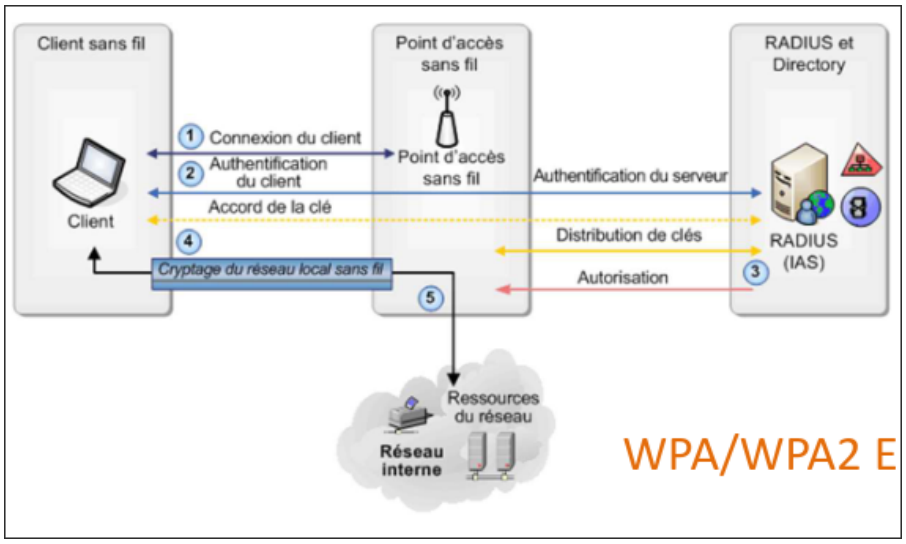
Encore une fois la structure contient un champ "next header" qui indique la couche de transport à utiliser.

L'IPSec ne peut pas être utilisé sur Internet comme protocole de couche réseau car les routeurs Internet ne le supporte pas. Donc l'IPSec est utilisé soit sur Internet en étant encapsulé dans des paquets IP, soit sur des grands réseaux locaux ou la sécuirté est critique.

La sécurité des réseaux sans-fil
La manière de sécuriser un réseau sans fil va varier en fonction de l'usage.

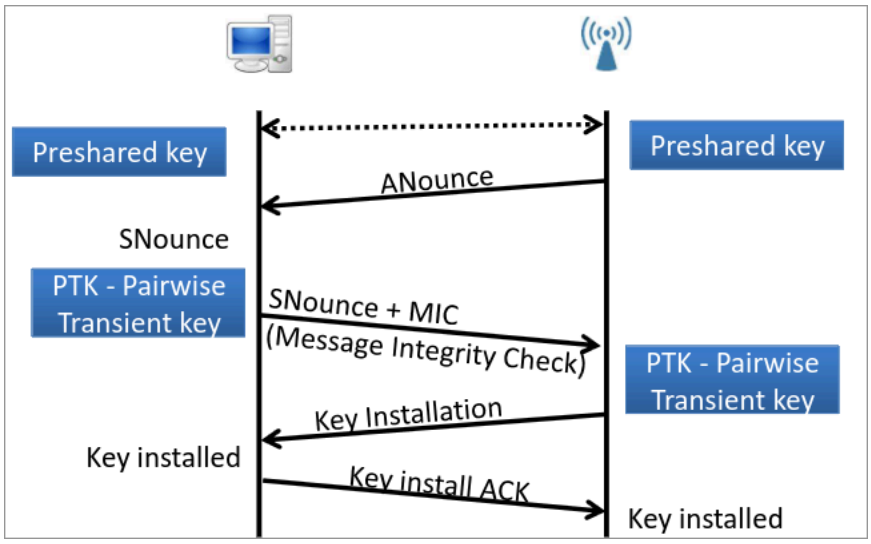
Il existe plusieurs normes mais seule le WPA2 et le WPA3 ont une sécurisé suffisante. Le WEP et le WPA sont tous les deux peu sûr.

Pour une **entreprise**, il est préférable de mettre en place un système d'authentification par login et mot de passe différent pour chaque personne. Cela est par exemple le cas d'HELMo avec le réseau *HELMo-Wifi* ou *eduroam*.



Pour le **public**, il est préférable d'utiliser un portail captif imposant un système d'enregistrement ou d'identification.

Et enfin pour un usage **personnel** il est préférable de faire les choses simple avec un mot de passe partagé entre tous·tes les utilisateur·ice·s. Il est aussi possible de sécuriser d'avantage en cachant le réseau (en empéchant qu'il soit automatiquement détecté), mais cela est peu pratique.



Les VPN

Un VPN **Virtual Private Network** permet d'intégrer dans le réseau interne une machine connnectée à l'extérieur du réseau via l'établissement d'un tunnel.

Il existe différentes sortes de VPN, certaines se base sur **PPTP (Point-to-Point Tunneling Protocol)** qui utilise MS-CHAPv2 pour l'authentification, ce système est peu sûr.

Il existe une autre sorte de VPN se basant sur **IPSec** qui est très sûr mais complexe à mettre en place.

Enfin il y a des VPN (et c'est probablement la majorité d'entre eux) qui se base sur **SSL/TLS** par exemple c'est le cas de OpenVPN qui est utilisé à HELMo. Il a l'avantage d'être sécurisé et beaucoup plus simple à mettre en place.

Pour en savoir plus sur les VPN, vous pouvez regarder cette vidéo

Cela pose cependant certains problèmes de sécurités car un VPN intègre une machine externe au réseau interne d'une entreprise. Une entreprise peut donc être attaquée par une machine extérieur et cela augmente donc le risque d'attaques. Il faut mettre en place un système de protection pour limiter le nombre de machines accessibles par le VPN, bien protéger le réseau avec un firewall et garder des logs afin de pouvoir analyser de potentielles attaques.