Systèmes et réseaux

Chapitre 7

Couche internet

Pablo Rauzy <pr@up8.edu>
pablo.rauzy.name/teaching/sr

Couche internet

La couche réseau

- La couche réseau est la troisième couche du modèle OSI.
- Dans le modèle TCP/IP, il s'agit de la couche internet.
- Son rôle est de faire le lien entre les réseaux locaux (ou étendus) afin de permettre la communication entre deux hôtes distants.
- Elle est donc en charge du *routage* des PDU qui transitent à son niveau.

Routage

- Le *routage* est le mécanisme qui sélectionne le chemin que vont prendre des données dans le réseau.
- Il existe différent type de routage :
  - *unicast* : les données sont envoyées à une seule destination spécifique,
  - *anycast* : les données sont envoyées à une seule destination, par exemple la plus proche.
  - *multicast* : les données sont envoyées à un ensemble donné de destinations,
  - *broadcast* : les données sont envoyées à tous le monde.
- Le routage est effectué par les *routeurs*.

- En ce qui concerne le routage, on peut distinguer deux catégories de machines :
  - les *hôtes* qui émettent et reçoivent des données,
  - les *routeurs* qui servent d'intermédiaires et sont en charge de l'acheminement des données jusqu'à leur destination finale.
- Quand un routeur est connecté à au moins deux réseaux, on parle alors de *passerelle* (*gateway*).
- Les passerelles sont généralement capables de "parler" plusieurs protocoles des deux ou trois premières couches du modèle OSI (et peuvent donc aussi être des switchs ou hubs).
- En pratique, même les simples routeurs sont des systèmes complexes et ne se limitent pas seulement au routage :
  - pare-feu, proxy, ordonnancement (qualité de service), DPI, ...

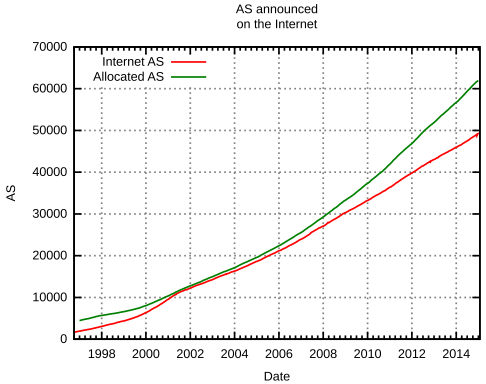
- Sur Internet, le routage se fait par *commutation de paquets* de manière *décentralisée*.
- Chaque routeur dispose d'information sur son voisinage dans sa *table de routage*.
- La table de routage contient trois types de routes :
  - les machines ou sous-réseaux directement connectées,
  - les routes statiques configurées en dur,
  - les routes dynamiques apprises via un *protocole de routage*.
- À cela s'ajoute les adresses du routeur lui même, et une *route par défaut*.
- Quand plusieurs routes sont possibles, la plus spécifique est empruntée.
- Pour voir les routes sur votre système : commande `netstat -r` ou `ip route`.

Backbone et Default-free zone

- L'architecture d'Internet s'est suffisamment décentralisée pour qu'il n'y ait plus réellement de *dorsale*.
- Cependant on utilise toujours le terme, parfois pour parler des *Tier 1 network*, parfois pour parler des routeurs dans la *default-free zone*.
- Un routeur est dit dans la default-free zone si il possède une table de routage complète et donc pas de route par défaut.
- En 2014, cela signifiait près de 500 000 routes vers près de 50 000 *systèmes autonomes*.

Systèmes autonomes

- On appelle *système autonome* (AS, *Autonomous System*) un ensemble de réseaux d'Internet dont le routage interne est cohérent.
- Par exemple, le réseau d'un fournisseur d'accès à Internet est un AS.
- Chaque AS dispose d'un numéro qui l'identifie uniquement.



Routage interne (exemple : OSPF)

- Au sein d'un AS, le routage est assuré par un protocole de *routage interne*, comme OSPF.
- OSPF signifie *Open Shortest Path First*, c'est un protocole de niveau "accès réseaux" dans le modèle TCP/IP.
- Le principe d'OSPF est de faire calculer à chaque routeur les plus courts chemins vers chaque sous-réseaux avec l'algorithme de Dijkstra.
- Le premier noeud de chaque chemin est inséré dans sa table de routage à une entrée commune à tous les chemins qui ont le même premier noeud.
- Pour avoir les informations nécessaires (le graphe du réseau) les routeurs s'échangent des messages :
  - *hello* quand il s'agit d'établir ses propres relations d'adjacences,
  - *link-state advertisements* (LSA) pour communiquer leur liste d'adjacence (elles sont relayées de proche en proche).

Routage externe (exemple : BGP)

- Entre AS, le routage est assuré par un protocole de *routage externe*, par exemple BGP.
- BGP signifie *Border Gateway Protocol*, c'est un protocole de niveau "application" dans le modèle TCP/IP.
- BGP a permis la décentralisation effective d'Internet (et donc la fin de la dorsale).
- Le principe de BGP est d'échanger entre passerelles des informations sur les réseaux connus et pour lesquels du transit est proposé (+ des informations pour éviter les boucles etc.).
- Pour cela sont établis des connexions point-à-point ou locales (dans un *Internet Exchange Point*), et si une telle connexion meurt, toutes les entrées apprises par elle dans la table de routage sont effacées.

Internet Exchange Point

- Un Internet eXchange Point (IX ou IXP) est une infrastructure physique (comme un gros switch) permettant le *peering* entre différents AS.
- Avec BGP, cela permet d'améliorer grandement la qualité du réseau :
  - le coût est réduit, car le trafic passant par un IX n'est potentiellement pas facturé à l'AS alors que les flux vers son fournisseur le sont,
  - la latence est réduite par l'existence d'interconnexions directes géographiquement proches,
  - la bande passante est améliorée pour les mêmes raisons.

Protocoles

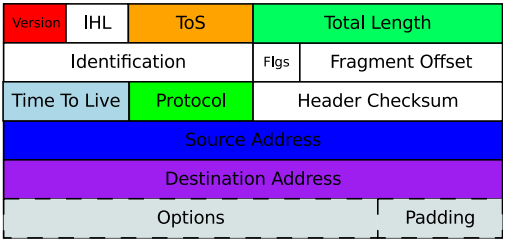
- Voyons quelques protocoles se situant $\sim$ au niveau de la couche réseau :
  - ARP,
  - IP (IPv4 et IPv6),
  - ICMP.

- Le protocole ARP (*Address Resolution Protocol*) se place entre les couches 2 et 3 du modèle OSI (ou entre les couches “accès réseau” et “internet” du modèle TCP/IP).
- Son but est d’associer les adresses MAC et les IP correspondantes.
- Quand une machine veut connaître l’adresse MAC d’une IP :
  - elle *broadcast* un message ARP disant qu’elle cherche à joindre cette IP, en renseignant son IP et son adresse MAC à elle ;
  - la machine concernée est la seule à répondre, pour cela elle ajoute l’association MAC et IP de la demandeuse dans sa table, et lui répond (en *unicast*) avec son adresse MAC.
- Les associations sont enregistrées dans un cache ARP pour ne pas avoir à faire cette requête à chaque envoie de paquet IP, mais ont une durée de vie limitée.
- Certains équipement (routeurs par exemple) broadcast leur IP et MAC de manière automatique lors de leur démarrage (on parle de *gratuitous ARP*), pour vérifier que leur adresse IP n’existe pas déjà et de se faire connaître immédiatement sur le réseau.

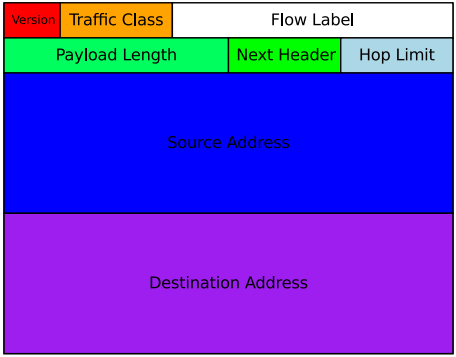
- Le protocole ARP (*Address Resolution Protocol*) se place entre les couches 2 et 3 du modèle OSI (ou entre les couches “accès réseau” et “internet” du modèle TCP/IP).
- Son but est d’associer les adresses MAC et les IP correspondantes.
- Quand une machine veut connaître l’adresse MAC d’une IP :
  - elle *broadcast* un message ARP disant qu’elle cherche à joindre cette IP, en renseignant son IP et son adresse MAC à elle ;
  - la machine concernée est la seule à répondre, pour cela elle ajoute l’association MAC et IP de la demandeuse dans sa table, et lui répond (en *unicast*) avec son adresse MAC.
- Les associations sont enregistrées dans un cache ARP pour ne pas avoir à faire cette requête à chaque envoie de paquet IP, mais ont une durée de vie limitée.
- Certains équipement (routeurs par exemple) broadcast leur IP et MAC de manière automatique lors de leur démarrage (on parle de *gratuitous ARP*), pour vérifier que leur adresse IP n’existe pas déjà et de se faire connaître immédiatement sur le réseau.
- Un petit mot sur l’*ARP poisoning*.

- IP (*Internet Protocol*) est une famille de protocoles du niveau OSI 3, ou niveau “internet” du modèle TCP/IP.
- Son rôle est d’acheminer des paquets des niveaux supérieurs (usuellement TCP ou UDP) de leur source à leur destination.
- Cet acheminement se fait sans qu’aucun chemin ne soit établi à l’avance (commutation par paquet et non par circuit) : il s’agit de protocoles *non orienté connexion*.
- En dehors d’assurer *au mieux* (*best effort*) cet acheminement des paquets, les protocoles IP ne fournissent aucune garantie quand à la corruption de données, l’ordre d’arrivée des paquets, la perte de paquet, la duplication de paquet.
- Les protocoles IP sont considérés *non fiables*, et la seule fiabilité offerte est une somme de contrôle sur les entêtes.
- Cela permet de réduire la charge des routeurs et d’augmenter leur vitesse de traitement.

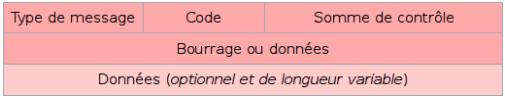
- IP version 4, décrite dans une RFC en 1981, est la première version a avoir été largement déployée.
- Elle est encore très largement majoritaire sur Internet en 2024.
- Une adresse IPv4 est codée sur 32 bits (exemple : `192.168.0.1`).
- La charge d'un paquet IP peut être fragmentée si elle rend le paquet plus grand que la MTU (*maximum transmission unit*, souvent $\sim$ 1500 à cause d'Ethernet).



- L'épuisement du nombre d'adresse IPv4 disponible (0 depuis le 3 février 2011) pousse à une transition vers IPv6, qui lui succède.
- Le fonctionnement d'IPv6 est similaire à celui d'IPv4.
- Les adresses sont codées sur 128 bits (`2620:0:862:ed1a::1`).
- `https://fr.wikipedia.org/wiki/IPv6`.



- ICMP (Internet Control Message Protocol) est un protocole de niveau OSI 3, mais au dessus d'IP en pratique.
- Pour IPv6, ICMPv6 est un protocole différent qui remplace ICMP (et ARP) d'IPv4.
- `https://fr.wikipedia.org/wiki/Internet_Control_Message_Protocol`.



Programmation

- Voyons quelques fonctions de l'API réseau :
  - `getaddrinfo` / `freeaddrinfo` / `gai_strerror` / `inet_ntop` ;
  - `socket` ;
  - `sendto` / `recvfrom` ;
  - `shutdown` / `close`.

getaddrinfo / freeaddrinfo / gai_strerror / inet_ntop

- Traduction d'adresses et services réseaux compatible IPv4 et IPv6 :
  - ```
    #include <sys/types.h>
    #include <sys/socket>.h>
    #include <netdb.h>

    int getaddrinfo(const char *restrict node,
                    const char *restrict service,
                    const struct addrinfo *restrict hints,
                    struct addrinfo **restrict res);

    void freeaddrinfo(struct addrinfo *res);

    const char *gai_strerror(int errcode);
    ```
- Conversion d'adresses IPv4 et IPv6 au format texte pour l'affichage :
  - ```
    #include <arpa/inet.h>
    const char *inet_ntop(int af, const void *restrict src,
                          char dst[restrict .size], socklen_t size);
    ```

- Créer un point de communication :
  - ```
    #include <sys/socket.h>
    int socket(int domain, int type, int protocol);
    ```
    ```

sendto / recvfrom

- Envoyer / recevoir un message via une socket :
  - ```
    #include <sys/socket.h>
    ssize_t sendto(int sockfd, const void buf[.len], size_t len, int flags,
                   const struct sockaddr *dest_addr, socklen_t addrlen);
    ```
  - ```
    #include <sys/socket.h>
    ssize_t recvfrom(int sockfd, void buf[restrict .len], size_t len,
                     int flags,
                     struct sockaddr *_Nullable restrict src_addr,
                     socklen_t *_Nullable restrict addrlen);
    ```

- Terminer une communication :
  - `#include <sys/socket.h>`
    `int shutdown(int sockfd, int how);`
  - `#include <unistd.h>`
    `int close(int fd);`

- En plus des pages de manuel, une bonne adresse :
  - `https://beej.us/guide/bgnet/`