Programmation Réseaux

Année 2024-2025

Anne Micheli

Organisation

Évaluations :

- partiel le vendredi 7 mars
- projet donné vers la 7ème semaine de cours
- examen final

Organisation

Le 1er TP des JI / Info 1 est aujourd'hui après le cours. Les autres TP commencent la semaine prochaine :

- JI / Info 1 - vendredi 16h15 - Aldric Degorre
- Info 2 - jeudi 10h45 - Aldric Degorre
- Info 3 - jeudi 16h15 - Peter Habermehl
- BI / Info 4 - lundi 14h15 - Fabien de Montgolfier
- Info 5 - lundi 10h45 - Anne Micheli
- MI - vendredi 8h30 - Astyax Nourel

Organisation

Le 1er TP des JI / Info 1 est aujourd'hui après le cours. Les autres TP commencent la semaine prochaine :

- JI / Info 1 - vendredi 16h15 - Aldric Degorre
- Info 2 - jeudi 10h45 - Aldric Degorre
- Info 3 - jeudi 16h15 - Peter Habermehl
- BI / Info 4 - lundi 14h15 - Fabien de Montgolfier
- Info 5 - lundi 10h45 - Anne Micheli
- MI - vendredi 8h30 - Astyax Nourel

Info 2 et Info 3 : TP des jeudis 1er et 8 mai reportés aux lundis précédents à 10h45 en 2031 $\rightarrow$ voir l'emploi du temps

Organisation

Le 1er TP des JI / Info 1 est aujourd'hui après le cours. Les autres TP commencent la semaine prochaine :

- JI / Info 1 - vendredi 16h15 - Aldric Degorre
- Info 2 - jeudi 10h45 - Aldric Degorre
- Info 3 - jeudi 16h15 - Peter Habermehl
- BI / Info 4 - lundi 14h15 - Fabien de Montgolfier
- Info 5 - lundi 10h45 - Anne Micheli
- MI - vendredi 8h30 - Astyax Nourel

Moodle : documentation de cours, énoncés de TP et projet, annonces...
Nous pourrons utiliser Moodle également pour vous écrire
$\Rightarrow$ **inscrivez-vous** sur Moodle dans votre **groupe de TP**.

Objectifs du cours

C'est un cours de programmation réseau et non de réseau.

- Comprendre les mécanismes de la communication « haut-niveau » entre des machines
- Faire communiquer des machines
- Formater des données
- Programmer en C

I- Introduction au réseau

Réseau

Réseau

Un **réseau** est formé d'entités (ordinateur, routeur, LAN) reliées entre elles par cable, fibre, wifi, ...

Réseau

Réseau

Un **réseau** est formé d'entités (ordinateur, routeur, LAN) reliées entre elles par cable, fibre, wifi, ...

Routeur

Un **routeur** permet de transmettre des paquets d'une interface à l'autre.

Réseau

Réseau

Un **réseau** est formé d'entités (ordinateur, routeur, LAN) reliées entre elles par cable, fibre, wifi, ...

Routeur

Un **routeur** permet de transmettre des paquets d'une interface à l'autre.

Interface

Une **interface** réseau assure la connexion entre un **hôte** (une machine) et un réseau donné.

Réseau

Réseau

Un **réseau** est formé d'entités (ordinateur, routeur, LAN) reliées entre elles par cable, fibre, wifi, ...

Routeur

Un **routeur** permet de transmettre des paquets d'une interface à l'autre.

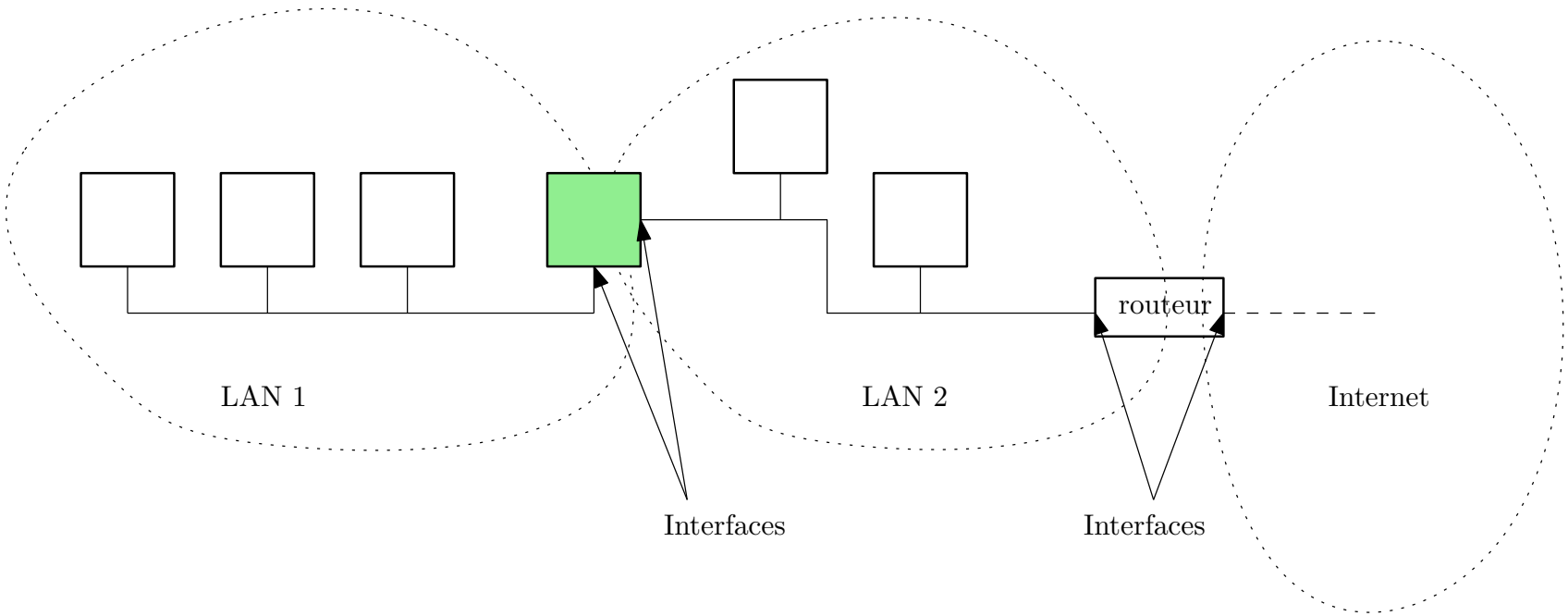
Interface

Une **interface** réseau assure la connexion entre un **hôte** (une machine) et un réseau donné.

LAN

Local Area Network : réseau local

Réseau



Adresses

Il y a deux types d'adresse :

Adresses

Il y a deux types d'adresse :

- **adresse MAC** (Media Access Control) : adresse physique d'une entité stockée en général dans la carte réseau. Utilisée pour l'adressage sur un même réseau local. L'adresse est unique et pérenne.

Adresses

Il y a deux types d'adresse :

- **adresse MAC** (Media Access Control) : adresse physique d'une entité stockée en général dans la carte réseau. Utilisée pour l'adressage sur un même réseau local. L'adresse est unique et pérenne.
- **adresse IP** (Internet Protocol) : adresse donnée temporairement à une entité sur un réseau donné. Une entité peut avoir plusieurs adresses IP, éventuellement sur des réseaux différents.

Adresse IPv4

L'adresse est codée sur **32 bits**.

Adresse IPv4

L'adresse est codée sur **32 bits**.

Représentation : **4 entiers** non signés codés chacun **sur 8 bits** donc chaque entier a une valeur entre 0 et 255. Les entiers sont séparés par un point.

Adresse IPv4

L'adresse est codée sur **32 bits**.

Représentation : **4 entiers** non signés codés chacun **sur 8 bits** donc chaque entier a une valeur entre 0 et 255. Les entiers sont séparés par un point.

Exemple : 10.0.175.225

Adresse IPv4

L'adresse est codée sur **32 bits**.

Représentation : **4 entiers** non signés codés chacun **sur 8 bits** donc chaque entier a une valeur entre 0 et 255. Les entiers sont séparés par un point.

Exemple : 10.0.175.225

L'adresse ***loopback*** : **127.0.0.1**

Adresse loopback

L'adresse **loopback** est l'adresse locale de l'entité. Les paquets ne sont pas envoyés sur le réseau mais vers une application locale à l'entité.

Adresse IPv4

L'adresse est codée sur **32 bits**.

Représentation : **4 entiers** non signés codés chacun **sur 8 bits** donc chaque entier a une valeur entre 0 et 255. Les entiers sont séparés par un point.

Exemple : 10.0.175.225

L'adresse ***loopback*** : **127.0.0.1**

Adresse loopback

L'adresse **loopback** est l'adresse locale de l'entité. Les paquets ne sont pas envoyés sur le réseau mais vers une application locale à l'entité.

$2^{32} = 4\ 294\ 967\ 296$ adresses IP distinctes $\Rightarrow$ pas assez d'adresses $\Rightarrow$ nécessité d'utiliser un autre ensemble d'adresses

Adresse IPv6

L'adresse est codée sur **128 bits**.

Adresse IPv6

L'adresse est codée sur **128 bits**.

Représentation : **8 entiers hexadécimales** codés chacun **sur 16 bits**.
Les entiers sont séparés par « : ».

Exemple : `1b01:fa06:0842:1510:fc:a41b:189a:31af`

Adresse IPv6

L'adresse est codée sur **128 bits**.

Représentation : **8 entiers hexadécimales** codés chacun **sur 16 bits**.
Les entiers sont séparés par « : ».

Exemple : `1b01:fa06:0842:1510:fc:a41b:189a:31af`

Règles d'écriture :

- on peut omettre les 0 de gauche de chacun des 8 entiers

`1b01:fa06:0842:0:0:0:189a:31af` ↔ `1b01:fa06:842:0:0:0:189a:31af`

- on peut remplacer *une seule fois* une suite de 0 consécutifs séparés par « : » par « :: »

`1b01:fa06:842:0:0:0:189a:31af` ↔ `1b01:fa06:842::189a:31af`

Adresse IPv6

L'adresse est codée sur **128 bits**.

Représentation : **8 entiers hexadécimales** codés chacun **sur 16 bits**.
Les entiers sont séparés par « : ».

Exemple : `1b01:fa06:0842:1510:fc:a41b:189a:31af`

Règles d'écriture :

- on peut omettre les 0 de gauche de chacun des 8 entiers

`1b01:fa06:0842:0:0:0:189a:31af` ↔ `1b01:fa06:842:0:0:0:189a:31af`

- on peut remplacer *une seule fois* une suite de 0 consécutifs séparés par « : » par « :: »

`1b01:fa06:842:0:0:0:189a:31af` ↔ `1b01:fa06:842::189a:31af`

L'adresse ***loopback*** : **::1**

Adressage IPv4

Certaines adresses sont réservées à un usage spécifique.

Adresses de préfixe :

- **127.0.0** : locale à l'hôte
- **10** ou **192.168** ou **172.16** à **172.31** : privée
- **224** à **239** : multidiffusion
- **255.255.255.255** : diffusion

Adressage IPv4

Certaines adresses sont réservées à un usage spécifique.

Adresses de préfixe :

- **127.0.0** : locale à l'hôte
- **10** ou **192.168** ou **172.16** à **172.31** : privée
- **224** à **239** : multidiffusion
- **255.255.255.255** : diffusion

notation CIDR (Classless Inter-Domain Routing) :

- permet de définir un ensemble d'adresses
- l'ensemble d'adresses 172.16 à 172.31 peut se noter **172.16/12**. Cela signifie que les 12 premiers bits sont fixes et les 20 suivants libres. Il y a donc 2^{20} adresses possibles avec un préfixe de longueur 12 bits.

NAT (network address translation)

IPv4 réserve certaines adresses pour des *usages privés* :

- **10.0.0.0/8** : 10.0.0.0 → 10.255.255.255
- **172.16.0.0/12** : 172.16.0.0 → 172.31.255.255
- **192.168.0.0/16** : 192.168.0.0 → 192.168.255.255

Pour communiquer avec l'extérieur, une entité avec une IP privée devra faire passer ses messages via un **NAT**.

Adressage IPv6

adresse IPv6 locale au lien : adresses **fe80 : :/10**

Adressage IPv6

adresse IPv6 locale au lien : adresses **fe80 : :/10**

adresses IPv6 globales : **2000 : :/3**

3 bits	45 bits	16 bits	64 bits
001	préfixe pour le routage global	sous-réseau	identification de l'hôte

Adressage IPv6

adresse IPv6 locale au lien : adresses **fe80 : :/10**

adresses IPv6 globales : **2000 : :/3**

3 bits	45 bits	16 bits	64 bits
001	préfixe pour le routage global	sous-réseau	identification de l'hôte

- pas de correspondance automatique entre adresses IPv4 et IPv6
- les réseaux IPv4 et IPv6 cohabitent
- pour connaître la configuration réseau de votre machine : commande **ip address show** ou **ifconfig** (obsolète)

Résolution de noms

Pour l'humain, difficile de retenir une adresse IPv4 ou IPv6.

$\rightarrow$ utilisation de noms pour désigner une adresse IP

Exemples : $\underbrace{\textbf{lucy}}_{\textbf{machine}}.\underbrace{\textbf{informatique.univ-paris-diderot.fr}}_{\text{domaine}}$

localhost désigne en général l'interface locale à l'hôte

Commande pour connaître le nom de l'hôte : **hostname** et le domaine : **hostname -d**

Résolution de noms

Pour l'humain, difficile de retenir une adresse IPv4 ou IPv6.

$\rightarrow$ utilisation de noms pour désigner une adresse IP

Exemples : $\underbrace{\textbf{www}}_{\textbf{machine}}.\underbrace{\textbf{informatique.univ-paris-diderot.fr}}_{\textbf{domaine}}$

Structure hiérarchique :

- la machine **www**
- dans le sous-domaine **informatique**
- dans le sous-domaine **univ-paris-diderot**
- dans le domaine **fr**

localhost désigne en général l'interface locale à l'hôte

Commande pour connaître le nom de l'hôte : **hostname** et le domaine : **hostname -d**

Résolution de noms

serveur DNS (Domain Name System)

Un **serveur DNS** permet d'associer les adresses de noms avec leurs adresses IP.

- annuaire distribué (il y a donc plusieurs serveurs DNS)
- Le DNS contient des informations liées à un nom de domaine :
  - l'adresse IPV4 (**A** record)
  - l'adresse IPv6 (**AAAA** record)
  - les serveurs de courrier électronique pour le domaine (**MX** record)
  - les serveurs DNS du domaine (**NS** record)
- commandes : **host** et **dig**
- le fichier `/etc/resolv.conf` contient les IP des serveurs DNS interrogé par le système

Protocole

protocole

Sur le réseau, les entités communiquent en suivant des règles rassemblées sous la forme d'un **protocole**.

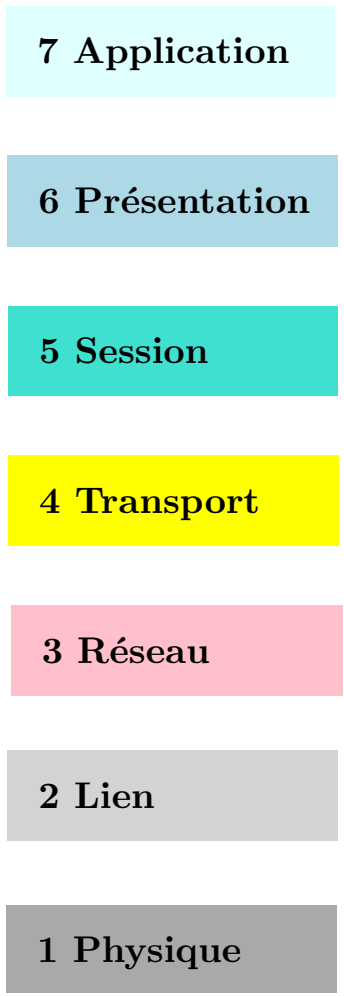
Il y a différentes sortes de protocoles. Des protocoles en charge :

- des formats des messages envoyés ou reçus par l'utilisateur
- des formats des données envoyées sur le réseau
- des messages + adresses source et de destination
- des données reçues et de leur cohérence

$\Rightarrow$ séparation en couches pour ces différents traitements.

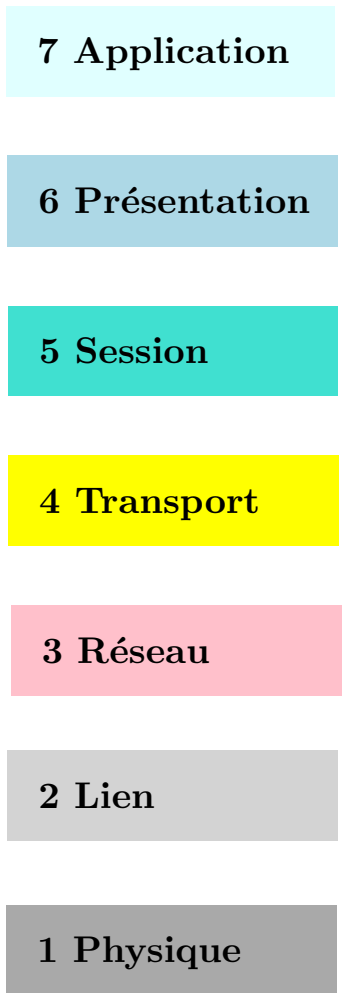
Les modèles en couches (layer)

Le modèle OSI (Open System Interconnection)



Les modèles en couches (layer)

Le modèle OSI (Open System Interconnection)



interface utilisateur/applications réseaux (navigateur, service mail, ...)

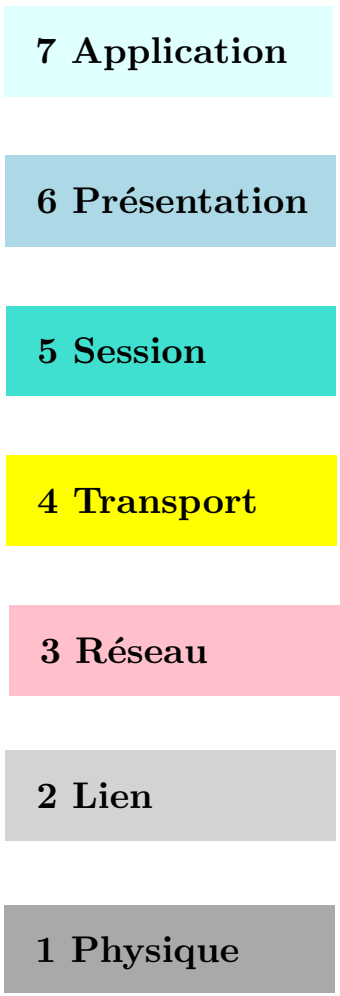
Les modèles en couches (layer)

Le modèle OSI (Open System Interconnection)

7 Application	interface utilisateur/applications réseaux (navigateur, service mail, ...)
6 Présentation	codage, cryptage des données (***data***)
5 Session	
4 Transport	
3 Réseau	
2 Lien	
1 Physique	

Les modèles en couches (layer)

Le modèle OSI (Open System Interconnection)



interface utilisateur/applications réseaux (navigateur, service mail, ...)

codage, cryptage des données (***data***)

connexion entre les deux hôtes (socket)

Les modèles en couches (layer)

Le modèle OSI (Open System Interconnection)

7 Application	interface utilisateur/applications réseaux (navigateur, service mail, ...)
6 Présentation	codage, cryptage des données (***data***)
5 Session	connexion entre les deux hôtes (socket)
4 Transport	couper les données en ***segments*** de données, les réassembler dans l'ordre (protocoles TCP, UDP)
3 Réseau	
2 Lien	
1 Physique	

Les modèles en couches (layer)

Le modèle OSI (Open System Interconnection)

7 Application	interface utilisateur/applications réseaux (navigateur, service mail, ...)
6 Présentation	codage, cryptage des données (***data***)
5 Session	connexion entre les deux hôtes (socket)
4 Transport	couper les données en ***segments*** de données, les réassembler dans l'ordre (protocoles TCP, UDP)
3 Réseau	transmette les segments de données sous forme de ***paquets*** - routage des paquets (protocole IP)
2 Lien	
1 Physique	

Les modèles en couches (layer)

Le modèle OSI (Open System Interconnection)

7 Application	interface utilisateur/applications réseaux (navigateur, service mail, ...)
6 Présentation	codage, cryptage des données (***data***)
5 Session	connexion entre les deux hôtes (socket)
4 Transport	couper les données en ***segments*** de données, les réassembler dans l'ordre (protocoles TCP, UDP)
3 Réseau	transmette les segments de données sous forme de ***paquets*** - routage des paquets (protocole IP)
2 Lien	encapsuler les paquets dans des ***trames (frame)*** contenant l'adresse MAC du prochain noeud - détecter les erreurs de transmission
1 Physique	

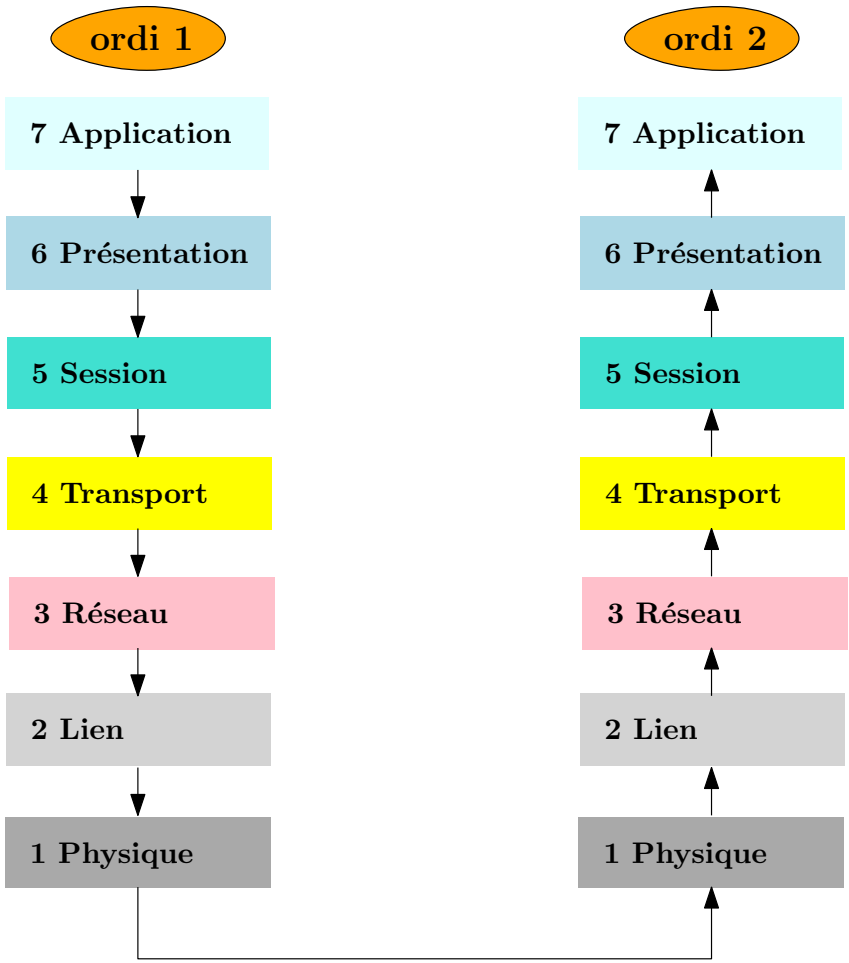
Les modèles en couches (layer)

Le modèle OSI (Open System Interconnection)

7 Application	interface utilisateur/applications réseaux (navigateur, service mail, ...)
6 Présentation	codage, cryptage des données (***data***)
5 Session	connexion entre les deux hôtes (socket)
4 Transport	couper les données en ***segments*** de données, les réassembler dans l'ordre (protocoles TCP, UDP)
3 Réseau	transmette les segments de données sous forme de ***paquets*** - routage des paquets (protocole IP)
2 Lien	encapsuler les paquets dans des ***trames (frame)*** contenant l'adresse MAC du prochain noeud - détecter les erreurs de transmission
1 Physique	transmission physique sur cable ethernet, Wifi, fibre optique

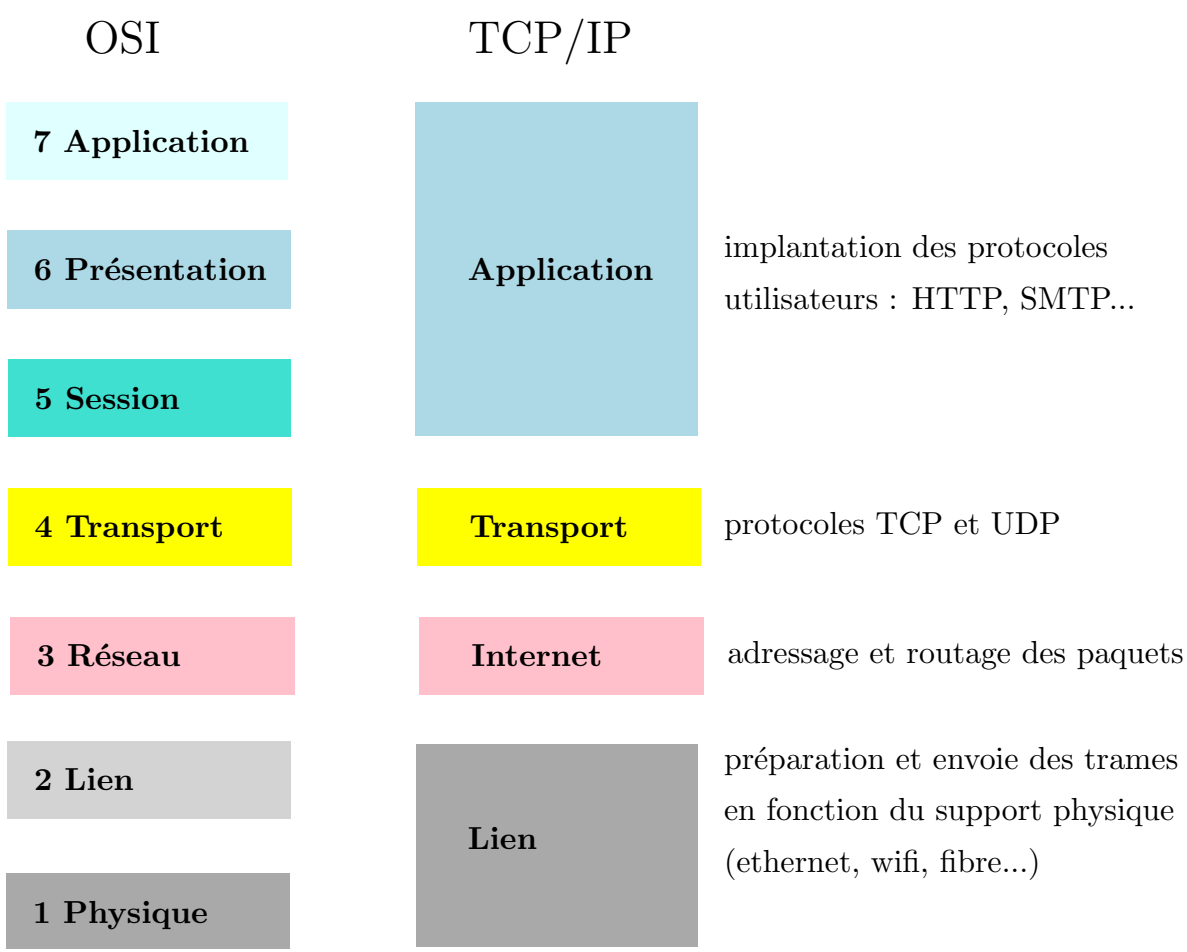
Les modèles en couches (layer)

Le modèle OSI



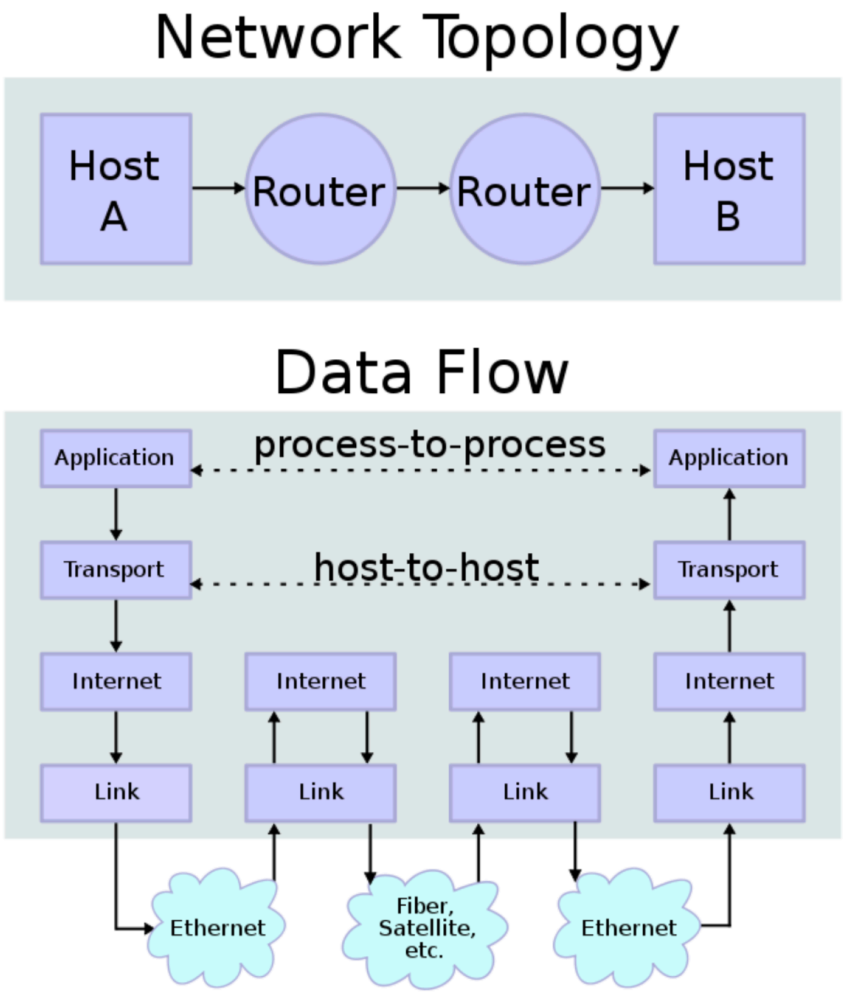
Les modèles en couches (layer)

Le modèle TCP/IP



Les modèles en couches (layer)

Le modèle TCP/IP



Source : Wikipedia

Les modèles de communication

Le modèle client/serveur

Des entités communiquent entre elles :

- une entité est le **serveur** qui répond aux requêtes des autres entités. Il fournit un **service**,
- chaque autre entité est un **client** qui entre en communication avec le serveur et l'interroge.

Le modèle pair à pair

Toutes les entités ont le même rôle. Chaque entité est à la fois un client et un serveur.

RFC

Requests For Comments (RFC) : documents officiels décrivant les standards, protocoles et technologies d'internet.

`https://www.ietf.org/standards/rfcs/`

exemples :

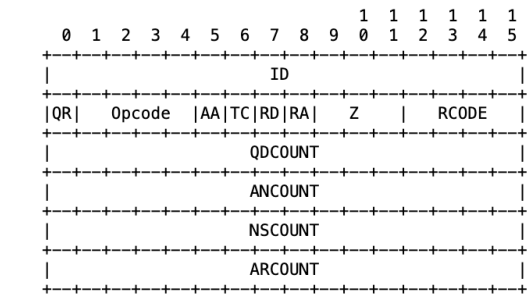
- RFC 2616 décrit le protocole HTTP/1.1
- RFC 2818 décrit le protocole HTTPS

RFC

Format de message

4.1.1. Header section format

The header contains the following fields:



where:

ID — A 16 bit identifier assigned by the program that generates any kind of query. This identifier is copied the corresponding reply and can be used by the requester to match up replies to outstanding queries.

QR — A one bit field that specifies whether this message is a query (0), or a response (1).

OPCODE — A four bit field that specifies kind of query in this message. This value is set by the originator of a query and copied into the response. The values are:

0	a standard query (QUERY)
1	an inverse query (IQUERY)
2	a server status request (STATUS)
3-15	reserved for future use

AA — Authoritative Answer - this bit is valid in responses, and specifies that the responding name server is an authority for the domain name in question section.

Note that the contents of the answer section may have multiple owner names because of aliases. The AA bit

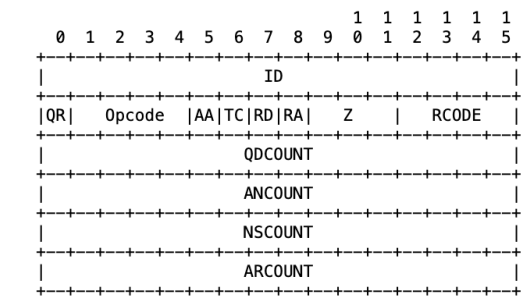
RFC

Format de message

RFC 1035 Domain Implementation and Specification November 1987

4.1.1. Header section format

The header contains the following fields:



where:

ID — A 16 bit identifier assigned by the program that generates any kind of query. This identifier is copied the corresponding reply and can be used by the requester to match up replies to outstanding queries.

QR — A one bit field that specifies whether this message is a query (0), or a response (1).

OPCODE — A four bit field that specifies kind of query in this message. This value is set by the originator of a query and copied into the response. The values are:

0 a standard query (QUERY)

1 an inverse query (IQUERY)

2 a server status request (STATUS)

3-15 reserved for future use

AA — Authoritative Answer - this bit is valid in responses, and specifies that the responding name server is an authority for the domain name in question section.

Note that the contents of the answer section may have multiple owner names because of aliases. The AA bit

Pour préparer un tel message avant son envoi sur le réseau :

- utiliser une structure en interne au programme,
- remplir la structure en utilisant éventuellement les décalages de bits,
- copier les champs de la structure dans un buffer d'octets (`char *`)
- ou bien écrire directement dans le buffer.

Le buffer contient alors le message à envoyer.

Port

Une machine peut avoir plusieurs applications réseaux qui tournent en parallèle.
De plus, ces applications peuvent utiliser la même adresse IP source.

$\Rightarrow$ nécessité de distinguer les données entrantes.

$\Rightarrow$ **port** : entier non-signé de 16 bits qui correspond à une adresse locale à la machine.

adresse IP $\leftrightarrow$ adresse d'un immeuble
port $\leftrightarrow$ numéro d'appartement

Port

Le numéro de port permet la différentiation des services lors de la réception.
Un couple (adresse, port) correspond à un **point de communication**.

- Les ports 0 à 1023 sont reconnus → correspondent à des services internet identifiés.
- Les ports 1024 à 49151 sont réservés → ceux que vous utiliserez. Également services d'usage moins général.
- Les ports 49152 à 65535 sont libres → durée limitée.

Correspondance entre le numéro de port et le service associé :

- sur le site de l'IANA (Internet Assigned Number Authority) :
  `https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml`
- sur votre machine, dans le fichier **/etc/services**

Simuler un serveur ou client simple

Les commandes **telnet** et **netcat** (ou **nc**) permettent entre autre de simuler un client simple :

`telnet <adresse> <port>`

`netcat <adresse> <port>`

netcat permet également de simuler un serveur simple : `netcat -l <port>`

Ordre des octets sur le réseau

Network Byte Order (ordre NBO)

Encodage des entiers sur une machine

exemple :
$2099202 = 2^{21} + 2^{11} + 2 = (00000000\ 00100000\ 00001000\ 00000010)_2$
entier sur 4 octets

- grand-boutiste ou big-endian (BE) : l'octet de poids fort sur la plus petite adresse mémoire

$\rightarrow$

0x0	0x1	0x2	0x3
00000000	00100000	00001000	00000010

Attention les **bits d'un octet** sont écrits de **droite à gauche**.

Encodage des entiers sur une machine

exemple :
$2099202 = 2^{21} + 2^{11} + 2 = (00000000\ 00100000\ 00001000\ 00000010)_2$
entier sur 4 octets

- grand-boutiste ou big-endian (BE) : l'octet de poids fort sur la plus petite adresse mémoire

→

0x0	0x1	0x2	0x3
00000000	00100000	00001000	00000010

- petit-boutiste ou little-endian (LE) : l'octet de poids faible sur la plus petite adresse mémoire

→

0x0	0x1	0x2	0x3
00000010	00001000	00100000	00000000

Attention les **bits d'un octet** sont écrits de **droite à gauche**.

Ordre des octets sur le réseau

Convention : les *adresses IP* et *port* sur le réseau sont encodés en *big endian*.

Sur votre machine, l'encodage des entiers peut être BE ou LE...

$\Rightarrow$ il faut penser à convertir adresse et port en big-endian avant envoi et, éventuellement, se préoccuper de la conversion inverse à la réception.

Conversion big endian/codage hôte

En C, on peut faire appel au fonctions suivantes de la bibliothèque `arpa/inet.h` :

uint32_t htonl(uint32_t hostlong) ;	host to network long
uint16_t htons(uint16_t hostshort) ;	host to network short
uint32_t ntohl(uint32_t netlong) ;	network to host long
uint16_t ntohs(uint16_t netshort) ;	network to host short