DIU “Enseigner l’Informatique au Lycée”

Réseaux

abdou.guermouche@u-bordeaux.fr
aurelien.esnard@u-bordeaux.fr

Université de Bordeaux, le 29 juin 2021.

Plan

- Chapitre 1 : Introduction [9h00 → 10h00]
- Chapitre 2 : Couches Basses & Réseau (Ethernet, IP, Sous-Réseaux) [10h15 → 11h15]
  - Activités : calcul de débit, calcul de sous-réseaux, encapsulation avec scapy & wireshark
- Chapitre 3 : Routage [11h15 -> 12h15]
  - Activités : mise en oeuvre avec un simulateur
- Chapitre 4 : Couches Transport & Application [13h15 → 14h15]
  - Activités : analyse de trames avec wireshark, netcat & netstat
- Chapitre 5 : Programmation Socket en Python [14h15 → 15h15]
  - Activités : requête web à la main, programmation d'un client web
- Chapitre 6 : Sécurité des Communications [15h30 → 16h30]

Chapitre 1

Introduction

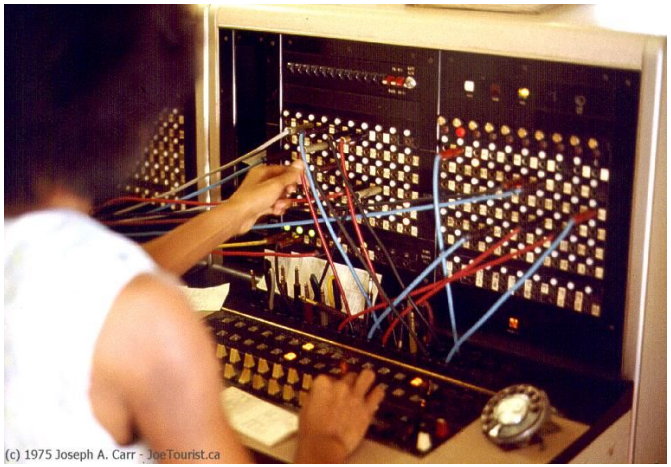
Un peu d'Histoire...

- 1832 – télégraphe électrique de Morse
- 1876 – invention du téléphone par Graham Bell
- 1948 – invention du transistor
- 1955 – premier réseau commercial pour Americal Airline réalisé par IBM (1200 téléscripteurs, infrastructure centralisé)
- 1956 – premier câble téléphonique transocéanique

American Airline. Source : Wikipedia

Un peu d'Histoire...

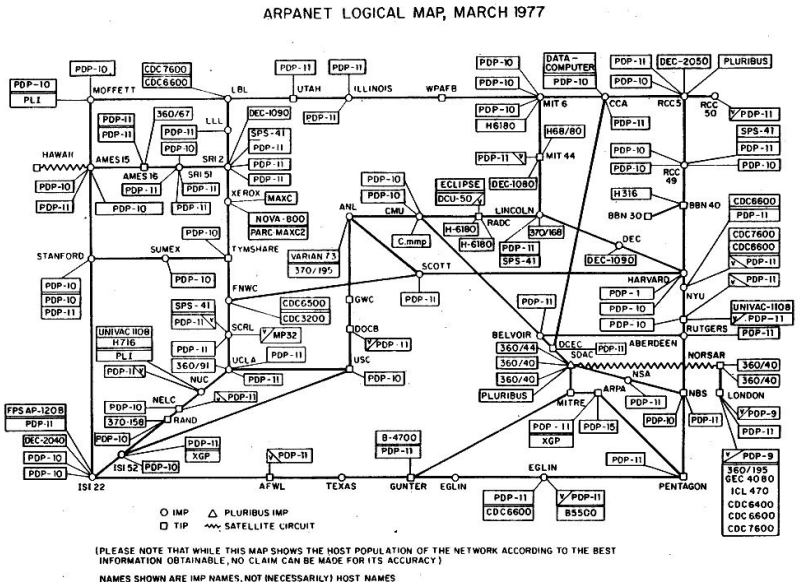
- 1958 – premier Modem (transfert binaire sur ligne téléphonique)
- 1961 – théorie sur la commutation de paquet (L. Kleinrock, MIT)
- 1962 – satellite Telstar1 (première liaison de télévision transocéanique)
- 1969 – permier pas de l'homme sur la lune (en direct)
- 1979 – premier réseau mondial de transmission de données par paquets X.25 ouvert au public (réseau **Transpac** en France)
- 1981-2012 – **Minitel** en France, basé sur Transpac (modem 1200 bits/s)



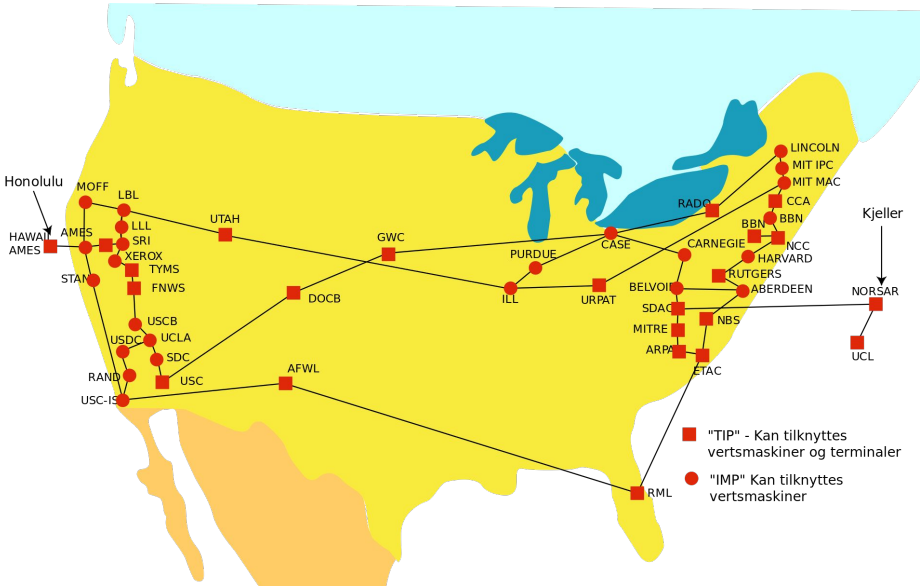
Minitel 1B. Source : Wikipedia

Un peu d'Histoire...

- 1959-1968 – programme ARPA (DoD)
- 1969 – **Arpanet**, basé sur le protocole NCP
- 1971 – **Cyclades**, un Arpanet français à base de datagramme (Louis Pouzin)
- 1973 – première publication sur TCP/IP (Vinton Cerf & Bob Kahn)
- 1983 – naissance d'**Internet** sur la base Arpanet qui adopte TCP/IP
  - mail, newsgroup, telnet, ...



Arpanet map. Source : Wikipedia

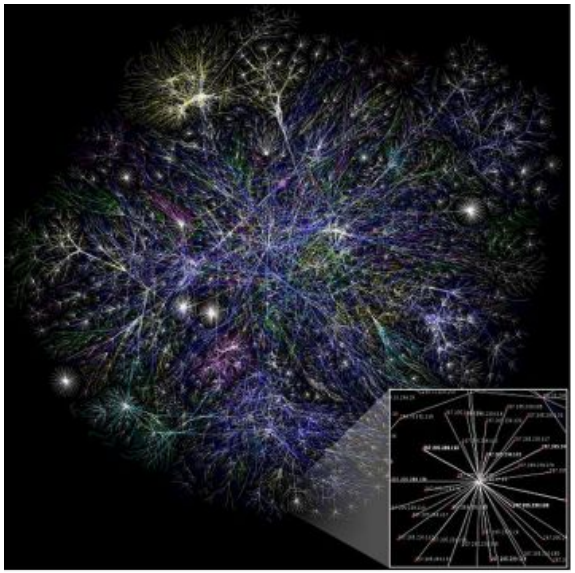


Arpanet en 1974. Source : Wikipedia

Internet

Internet : réseau informatique mondial, résultant de l'interconnexion d'une multitude de réseaux informatiques à travers la planète, unifiées grâce au protocole IP. [1983]

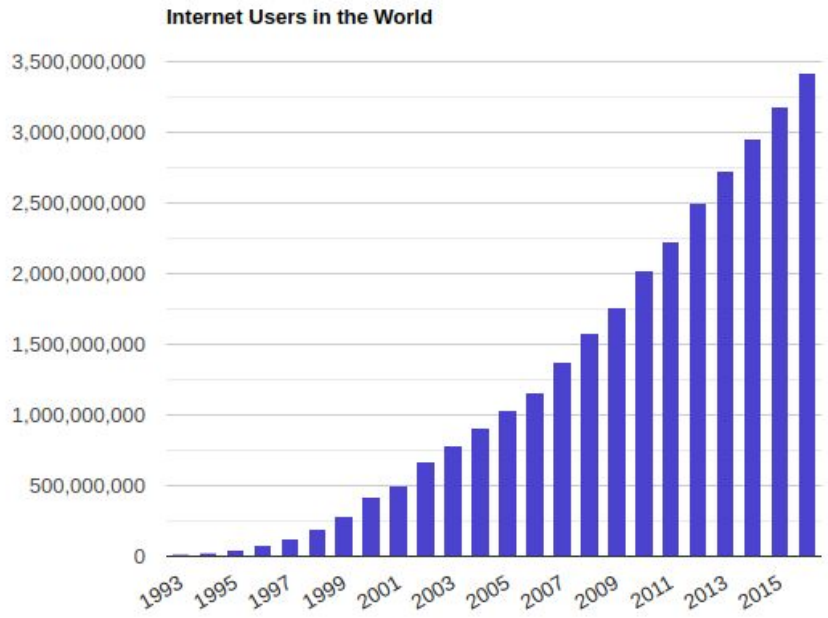
Protocole réseau : un protocole définit de manière formelle et interopérable l'échange des informations entre ordinateurs.

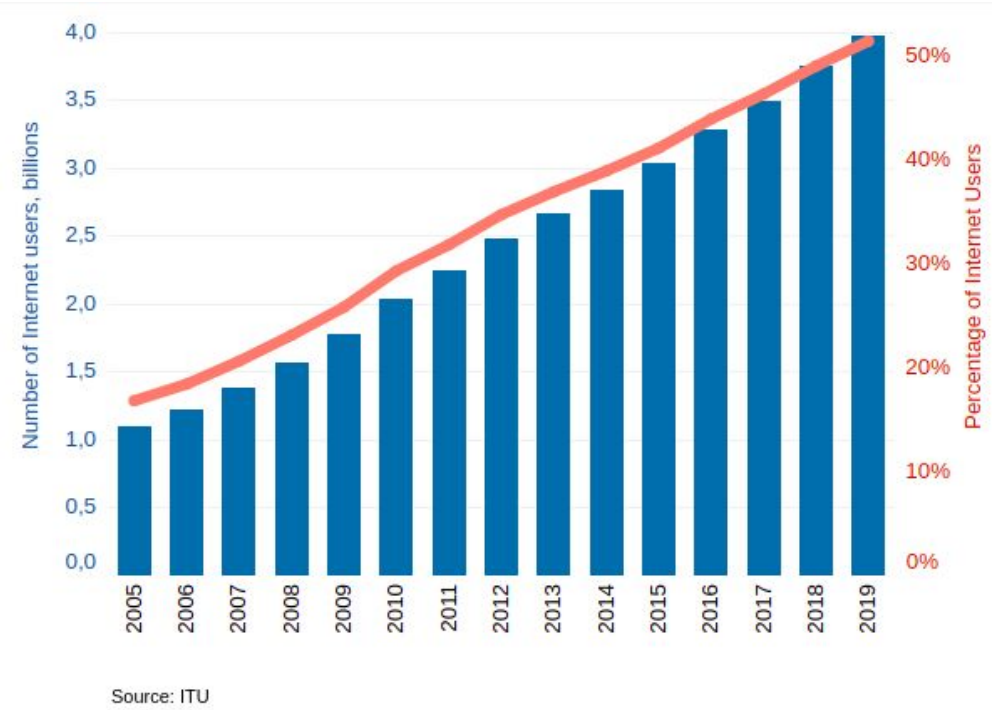


Source : Wikipedia

Internet

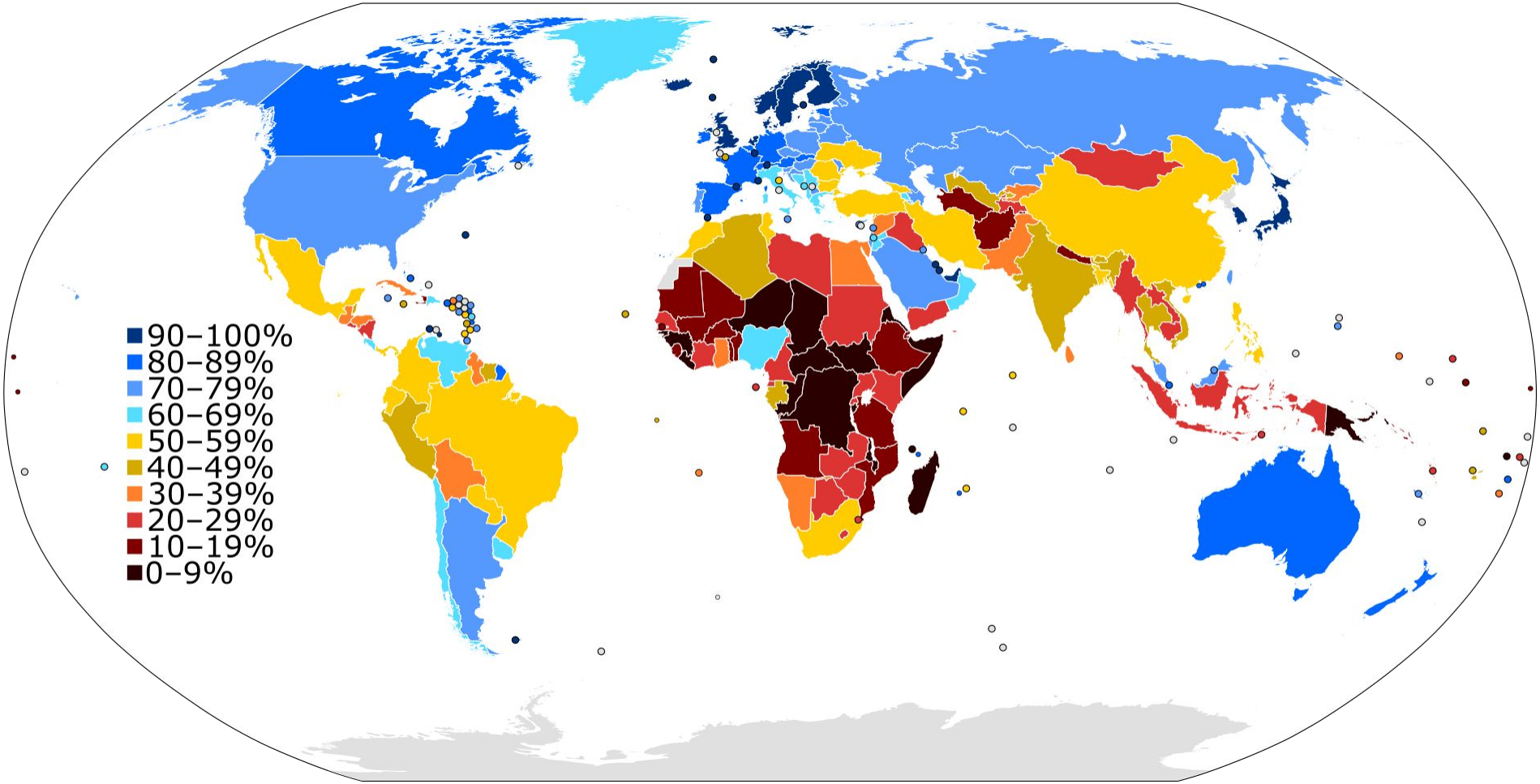
- 1990 – démocratisation d'Internet (invention du web)
- 1990-2000 – ouverture au grand public avec les FAI (ou ISP)
- 2005 – 1 milliard d'internautes
- 2010 – 2 milliard d'internautes
- 2020 – aujourd'hui, 4.8 milliard d'internautes !!!





Source : https://www.internetlivestats.com

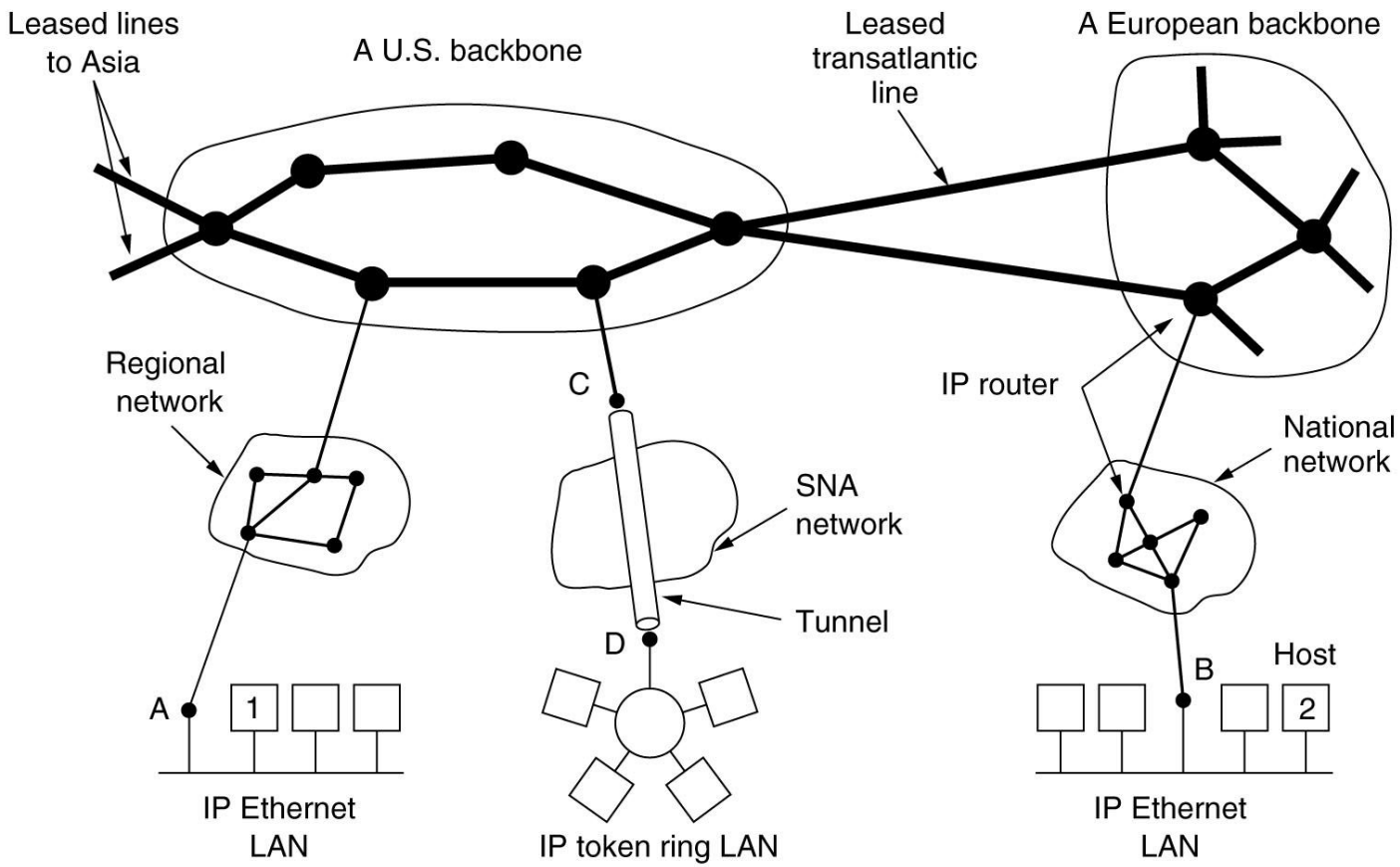
Accès à Internet dans le Monde



Internet users in 2015 as a percentage of a country's population.
Source: International Telecommunications Union.

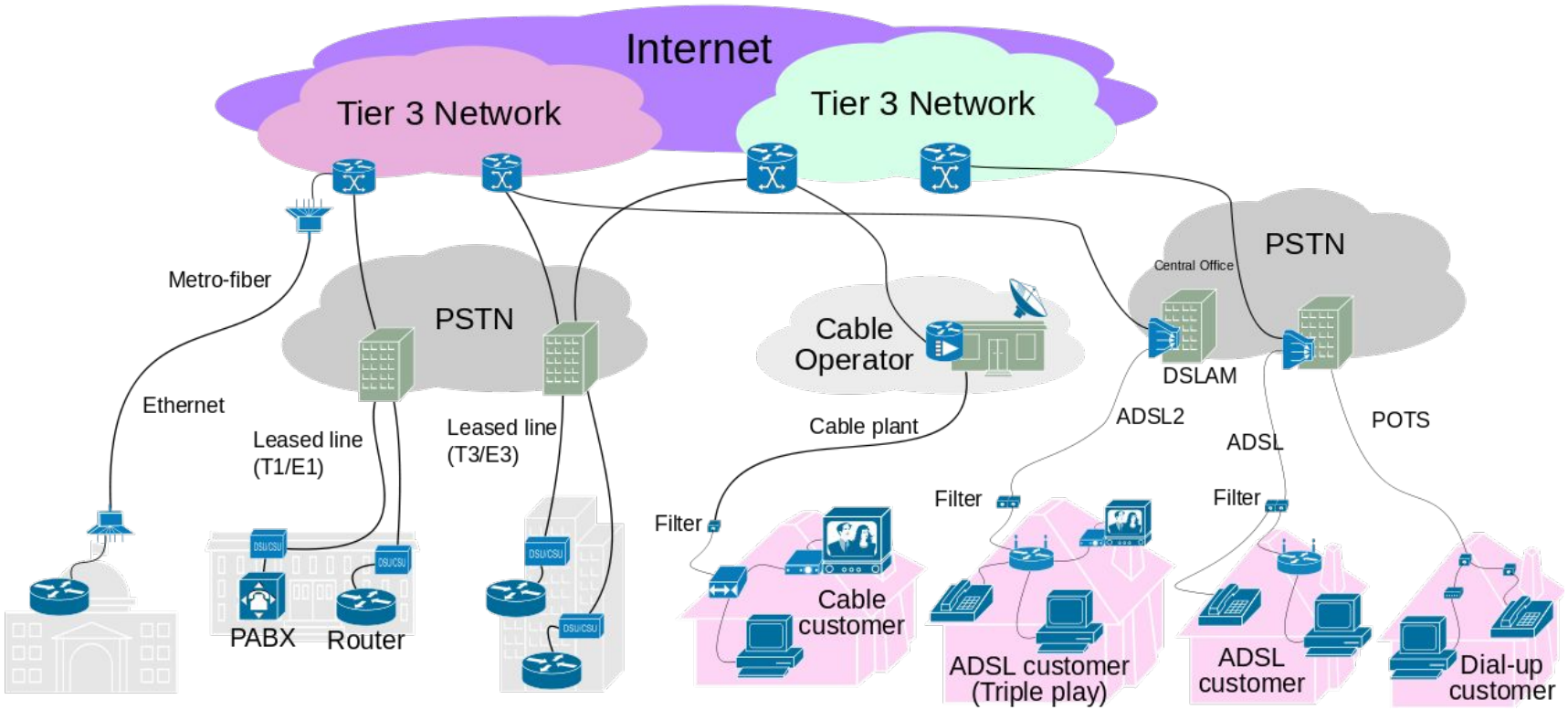
Internet

Interconnexion de multiples réseaux hétérogènes et distants...

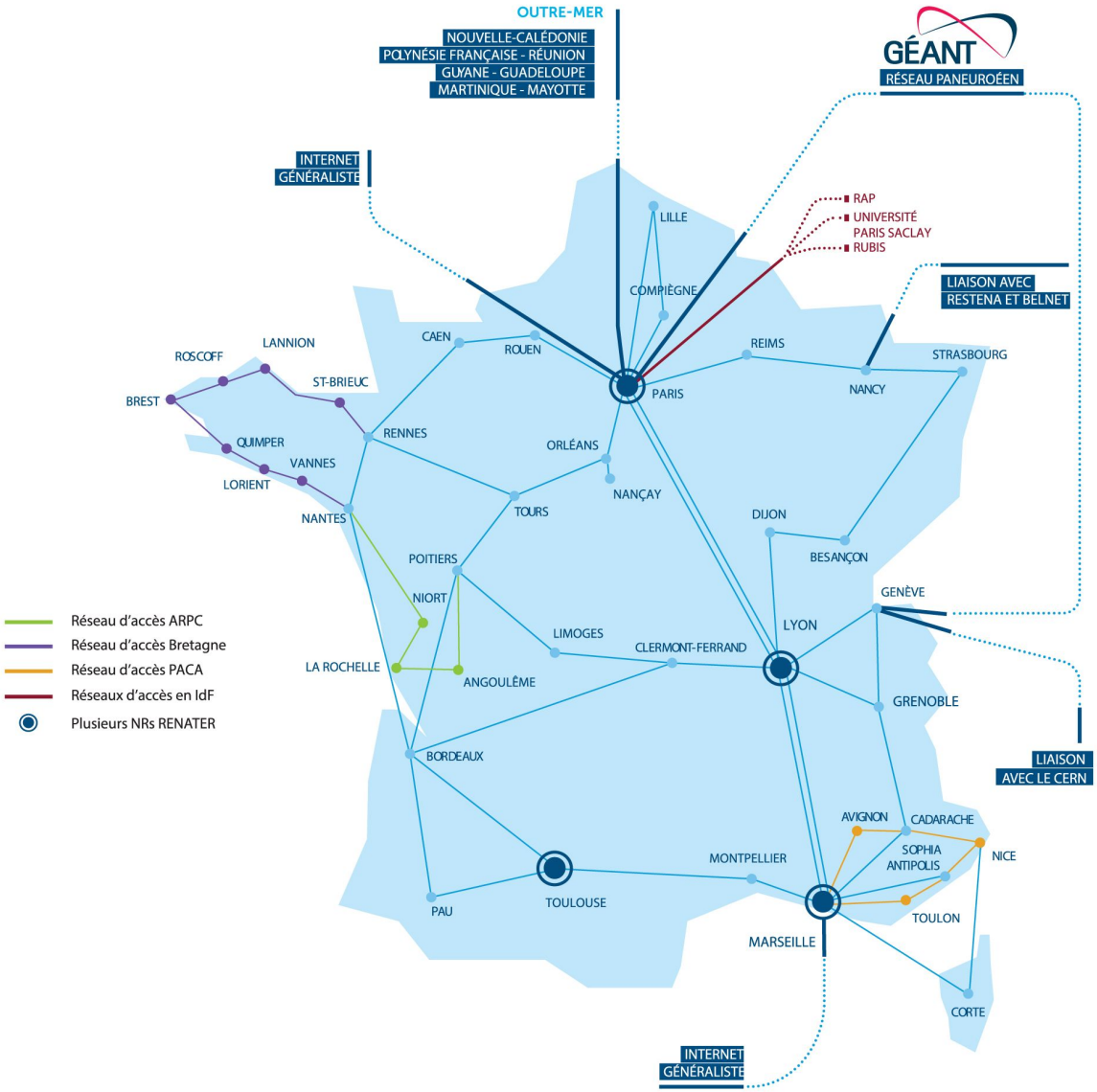


Internet

Une structure hiérarchique...



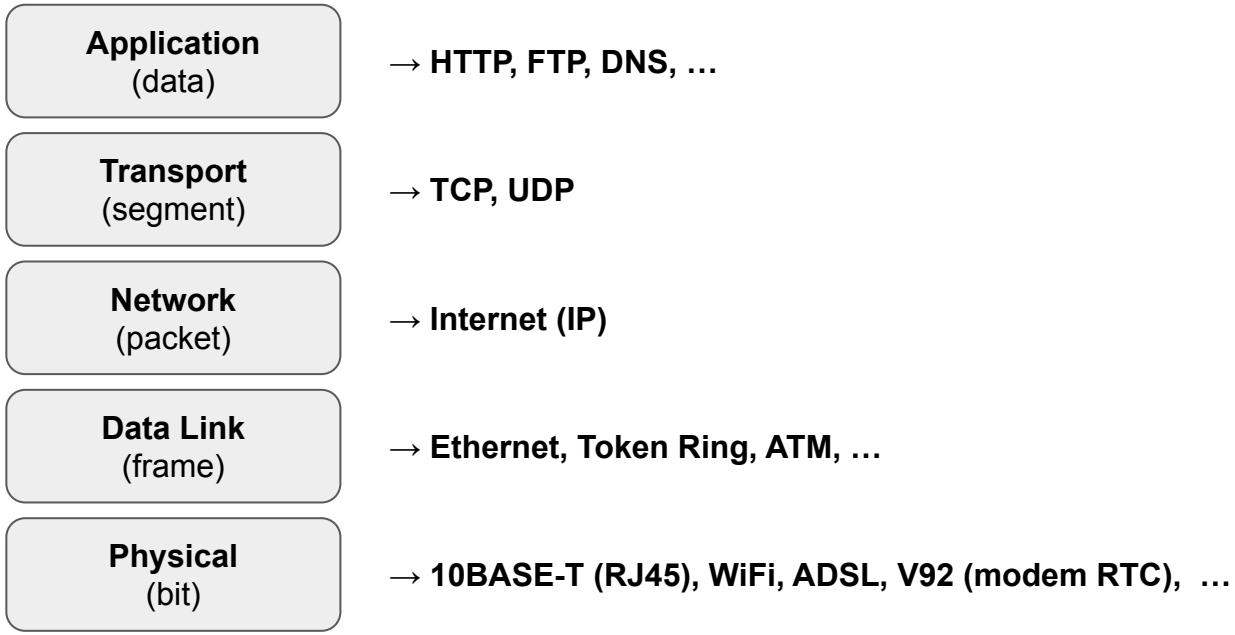
Exemple de Renater



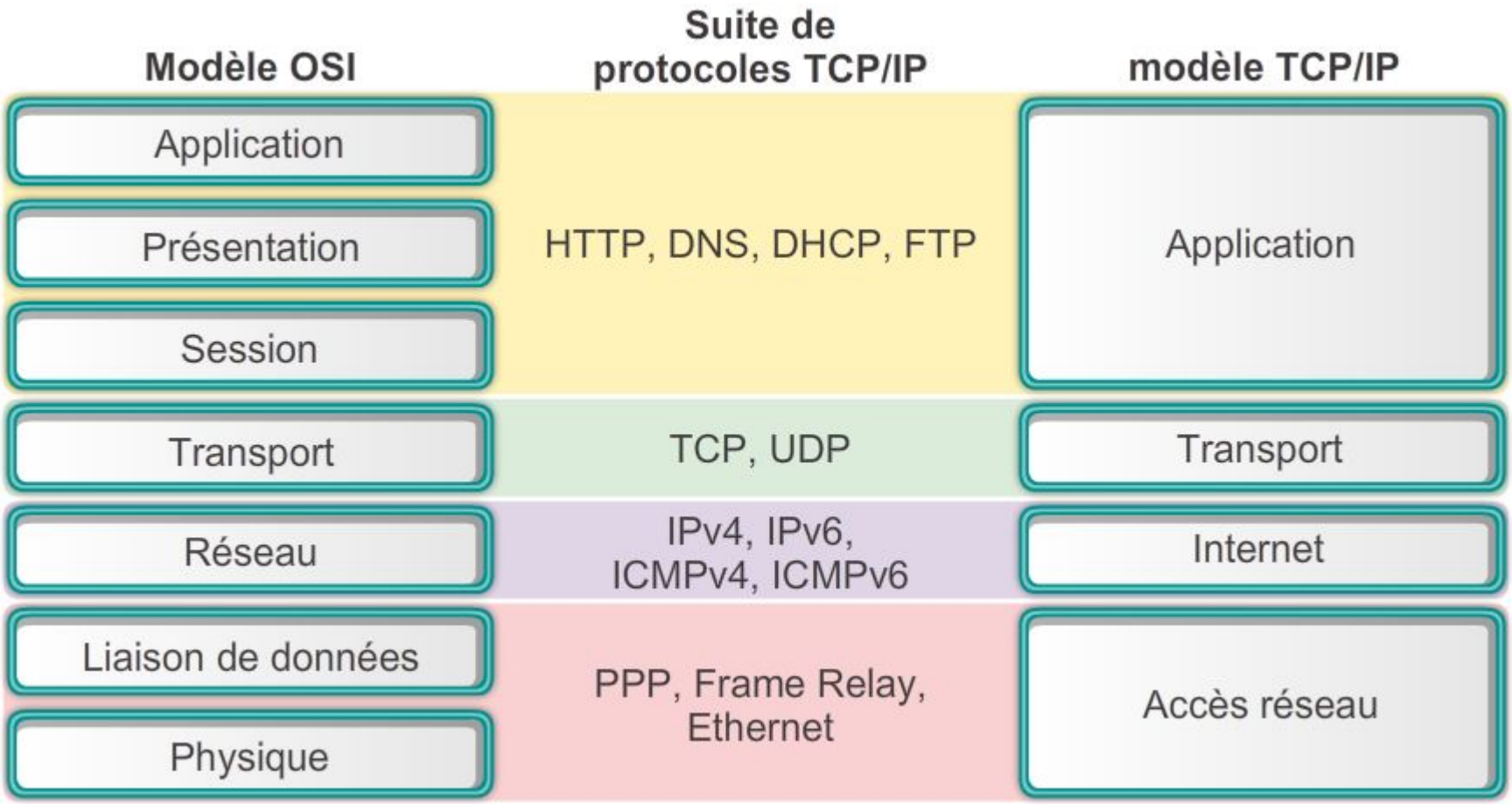
Notion de Protocole

Protocole : Spécification de plusieurs règles pour communiquer sur une même couche d'abstraction entre deux machines.

Modèle en Couche OSI (simplifié)



Modèle OSI vs TCP/IP



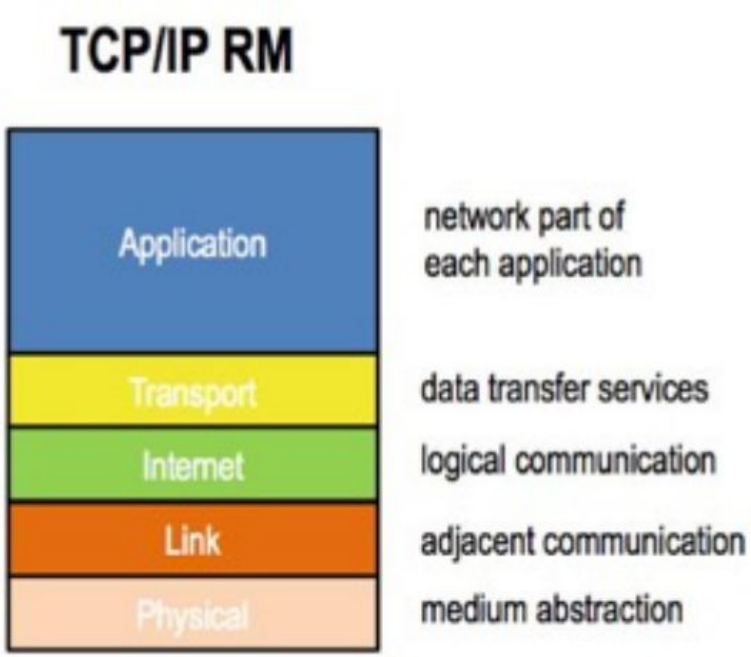
Source : https://linux-note.com/modele-osi-et-tcpip/

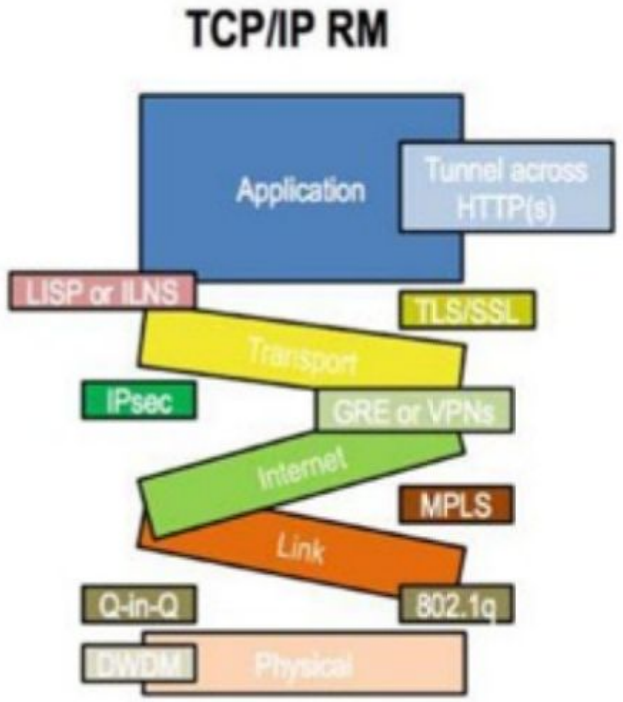
Modèle en Couche OSI

1. **Couche physique** (physical layer) : transmission effective des signaux entre les interlocuteurs ; service typiquement limité à l'émission et la réception d'un bit ou d'un train de bit continu.
2. **Couche liaison de données** (datalink layer) : communications entre deux machines adjacentes, i.e. directement reliés entre elle par un support physique.
3. **Couche réseaux** (network layer) : communications de bout en bout, généralement entre machines (adressage logique et routage des paquets).
4. **Couche transport** (transport layer) : communications de bout en bout entre programmes (UDP, TCP).
5. **Couche session** (session layer) : synchronisation des échanges et transaction, permet l'ouverture et la fermeture de session.
6. **Couche présentation** : codage des données applicatives, et plus précisément conversion entre données manipulées au niveau applicatif et chaînes d'octets effectivement transmises
7. **Couche application** : point d'accès aux services réseaux ; non spécifiée.

De la Théorie à la Réalité…

Evolution du modèle TCP/IP, de patch en patch, 50 ans plus tard !





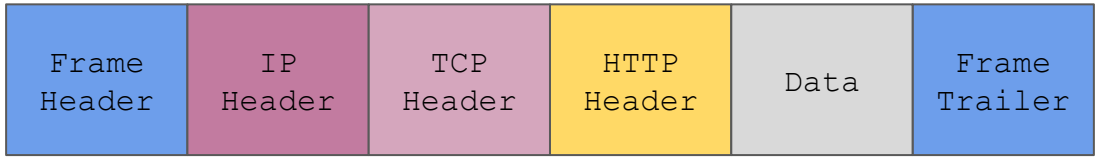
Source : Louis Pouzin

Exemple du Protocole HTTP

En pratique, plusieurs niveaux d'interactions...

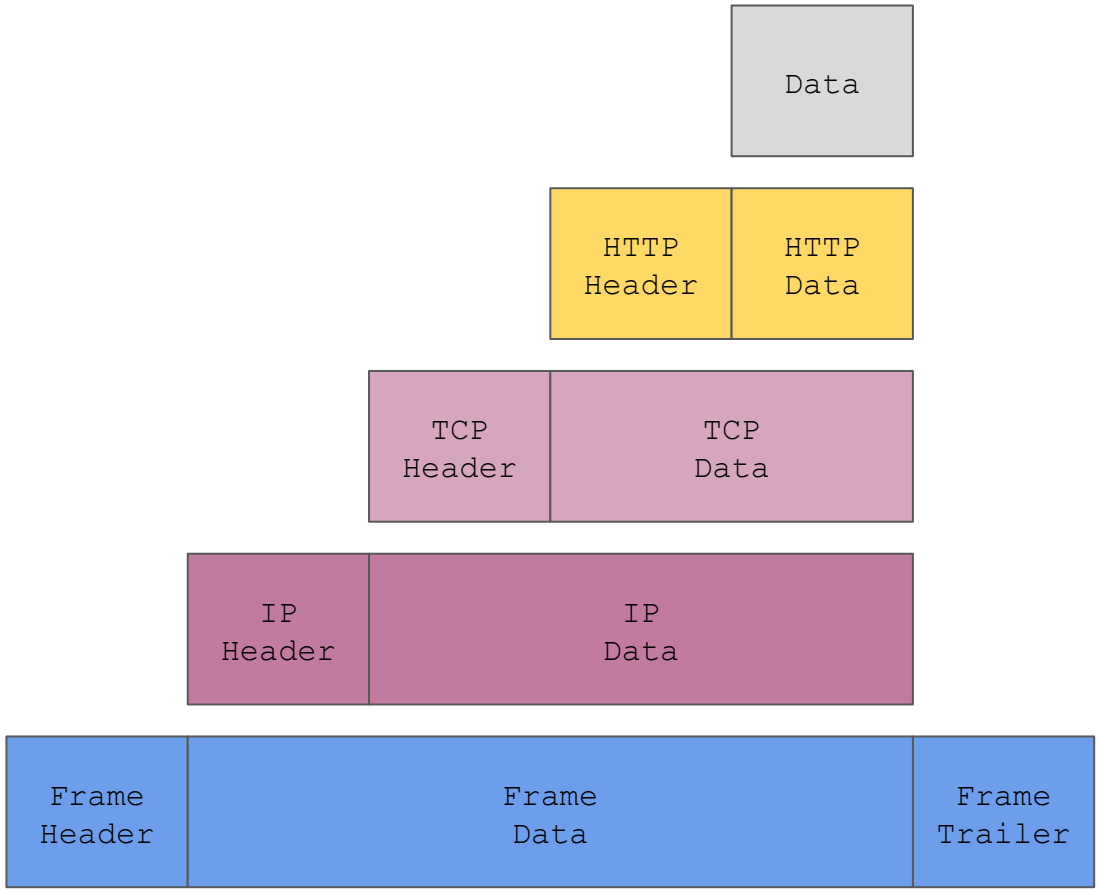
- le niveau de l'application : le client clique sur un lien, le serveur renvoie une page web (HTTP, encodage UTF8 & MIME)
- le niveau des messages : le client envoie un message contenant une URI, le serveur renvoie un message contenant un fichier HTML (TCP, 80)
- le niveau des paquets : le message du client est découpé en paquets IP, les différents routeurs du réseau les acheminent vers le serveur (idem pour le retour)
- le niveau de la transmission des bits : pour envoyer les paquets, chaque bit (0 ou 1) est transmis comme un signal électrique sur une ligne (Ethernet et al)

Chaque niveau utilise les fonctions du niveau inférieur.



Exemple du Protocole HTTP

Encapsulation des protocoles…



Une page web (HTML)

Couche Application (HTTP)

Couche Transport (TCP)

Couche Réseau (IP)

Couche Liaison (Ethernet)

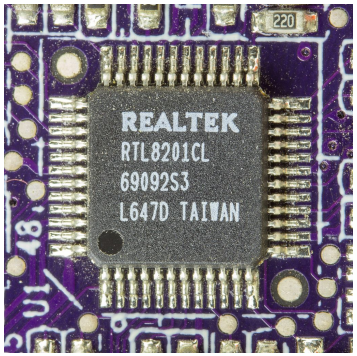
Chapitre 2

Les Couches Basses (Ethernet)

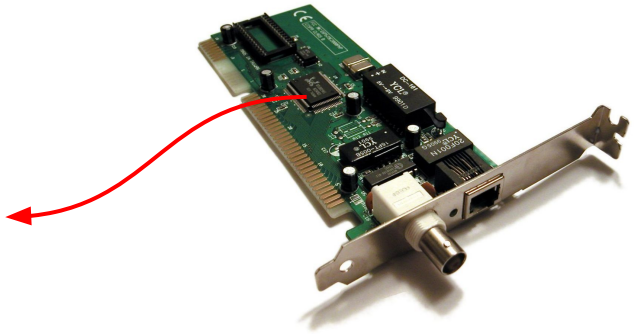
Les Couches Basses

Couche Physique (physical layer)

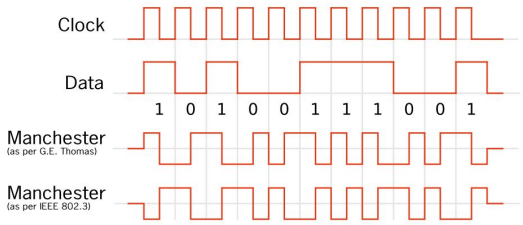
- transmission effective des signaux (électriques, radiogréquences, optiques)
- service typiquement limité à l'émission et la réception d'un bit ou d'un train de bit continu
- réalisé par un circuit électronique spécifique, appelé PHY (*physical transceiver*)

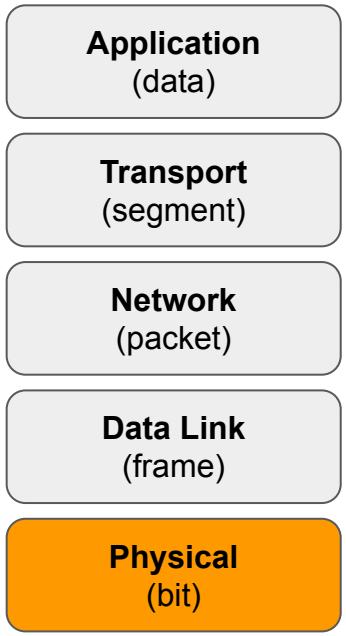


Ethernet PHY



carte réseau Ethernet

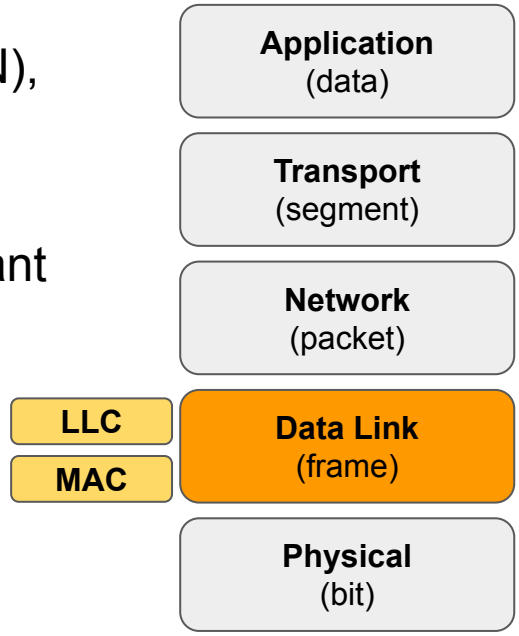


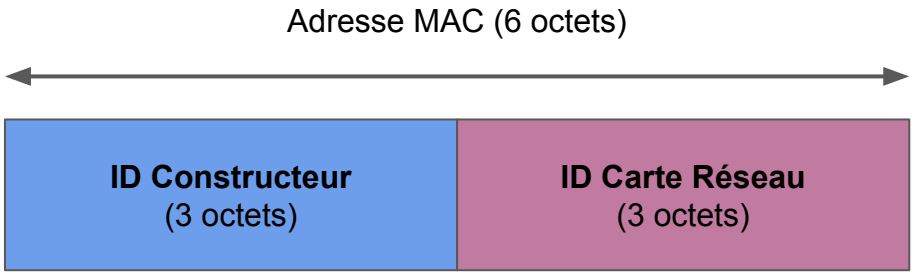


Les Couches Basses

Couche Liaison de Données (data link layer)

- communication entre les noeuds d'un réseau local (LAN), directement reliés par un support physique…
- **LLC** (Logical Link Control) : sous-couche haute
- **MAC** (Media Access Control) : sous-couche basse faisant l'interface avec une couche physique spécifique…
  - détection début & fin de trame, gestion des erreurs (CRC)
  - implantation logicielle ou matérielle sur la carte réseau…
  - adressage MAC : 52:54:00:A1:61:CB





Le Standard Ethernet

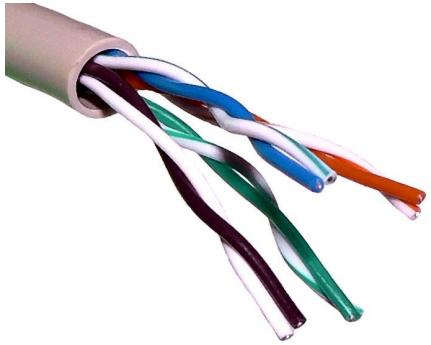
Première technologie LAN grand-public

- inventé au début des années 70 par Xerox, spécifié dans les années 80.
- plusieurs variantes du standard : Ethernet II, IEEE 802.3, …
- évolution du débit : Ethernet (10 Mb/s), Fast Ethernet (100 Mb/s), Giga Ethernet (1000 Mb/s), et plus…

Cablage

- Cable coaxial (10BASE2, 10BASE5)
- Cable UTP : paires torsadées (10BASE-T, 100BASE-TX, …)

cable coaxial



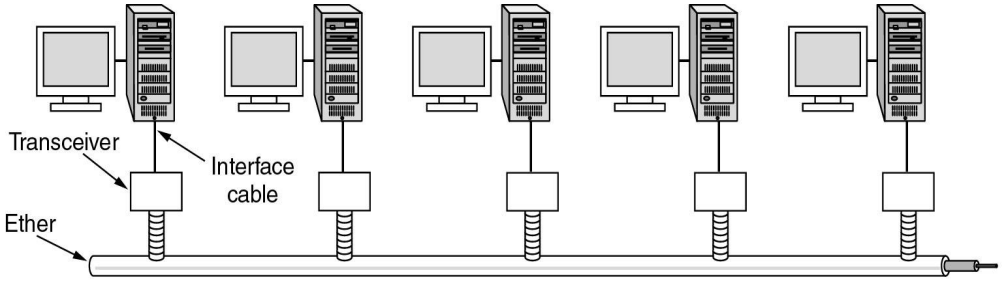
4 paires torsadées

connecteur RJ45

Interconnexion des Machines

Bus

- topologie linéaire, canal partagé
- collision possible des signaux, sujet aux pannes, *désuet*



Hub (concentrateur)

- topologie en étoile, half-duplex, canal partagé
- collision possible des signaux, *désuet*

Switch Ethernet 5 ports

Switch (commutateur)

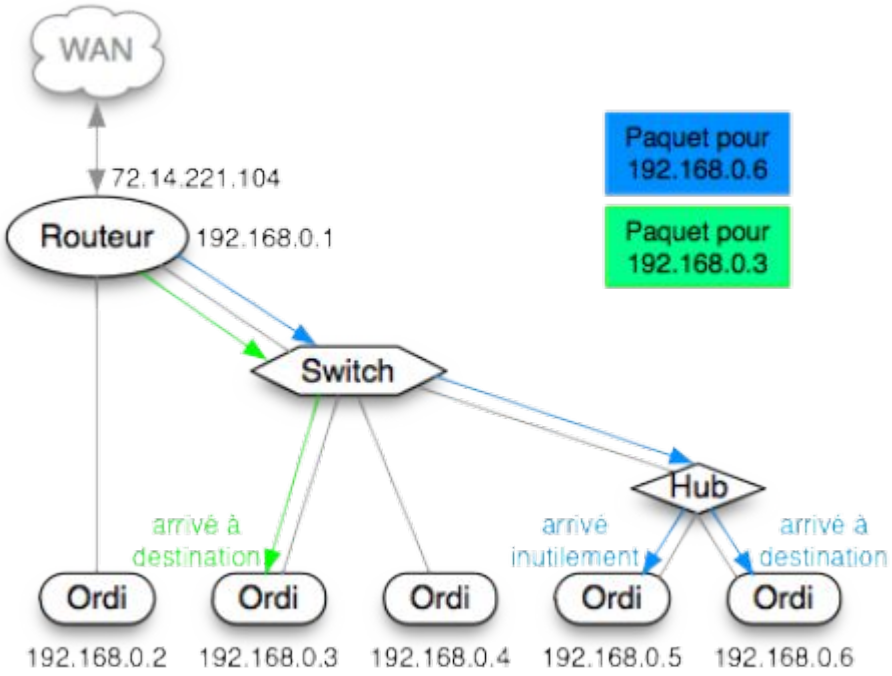
- topologie en étoile, full duplex, canal dédié avec le destinataire choisi (circuit virtuel créé par le commutateur) ⇒ pas de collision

Passerelle / Routeur

- Matériel reliant deux réseaux différents et les faisant communiquer

Interconnexion des Machines

Exemple

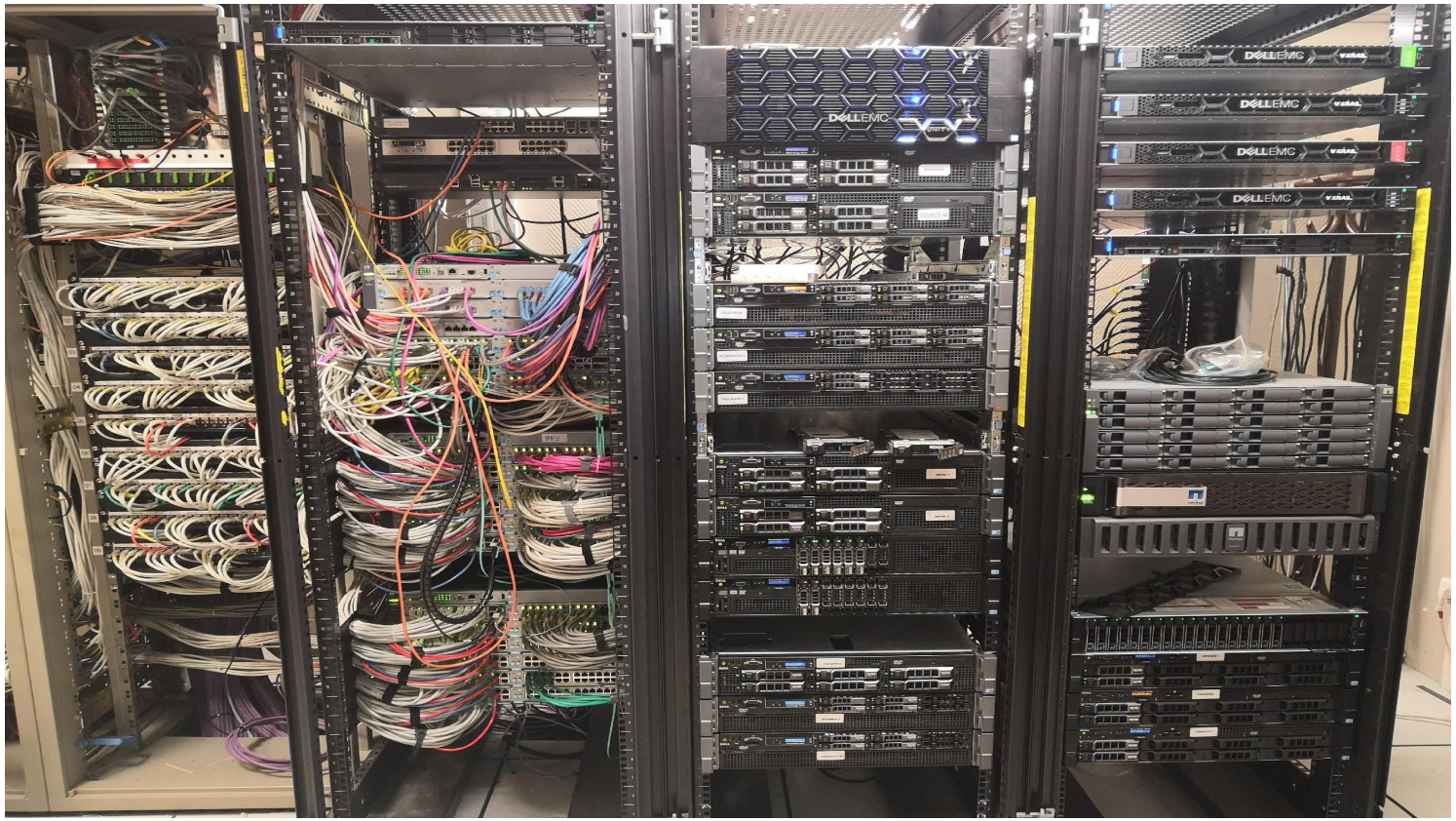


Source : http://bencello.net/Tutos.php

Exercice : Au même moment, deux paires de machines communiquent en saturant un réseau Ethernet à 100 Mbit/s. Quel débit maximal peut-on espérer entre chaque machine avec un Hub ou un Switch ?

Armoire Réseau

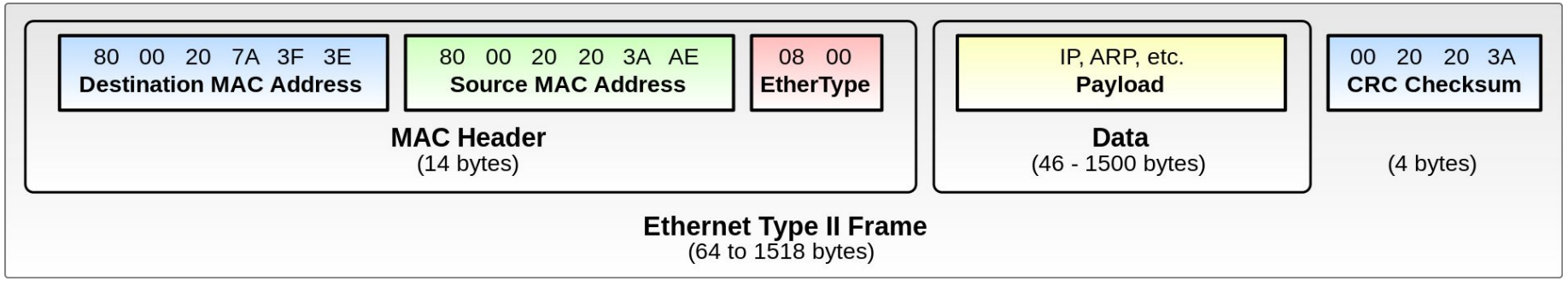
Baie de brassage dans la salle serveur du CREMI.



Trame Ethernet

Cas du standard Ethernet II (64 octets minimum)

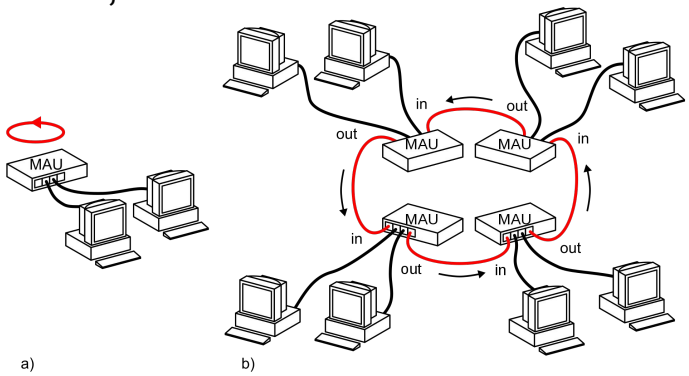
- **Flag Start :** marqueur de début de trame (010101...0101 11) [8 octets]
- **Adresse MAC destination** [6 octets]
- **Adresse MAC source** [6 octets]
- **EtherType** : 0x0800 = IPv4 ; 0x86DD = IPv6 ; 0x0806 = ARP ; … [2 octets]
- **Data :** au minimum 46 octets, jusqu'à 1500 octets (selon MTU), caractères de bourrage (*padding*) si pas assez de données
- **Checksum :** CRC-32 [4 octets]
- **Flag End** : silence à la fin avant la prochaine trame [12 octets]



Une Grande Variété de Technologies

Une séparation pas souvent très claire entre les couches basses !

- **Token Ring** : topologie en anneau, jeton (trame de 3o) qu'il faut passer à son voisin pour qu'il puisse commencer à émettre…
- **FDDI** (Fiber Distributed Data Interface) : mise en oeuvre d'un double anneau à jeton avec la fibre…
- **ATM** (Asynchronous Transfer Mode) : transmission des données par cellules de tailles fixes, très répandu au coeur du réseau ADSL !
- **V92** : protocole physique utilisé par les modems téléphoniques 56K
- **Wi-Fi** (IEEE 802.11) : variante sans fil d'Ethernet ou *Wireless* LAN
- **PPP** (Point-to-Point Protocol), **SLIP** (Serial Line Internet Protocol), **FTTH** (Fiber to the Home), **ADSL**, …



Débit et Latence

Débit : nombre de bits que le réseau peut transporter par seconde…

- asymétrie : débit montant (*upload*) & débit descendant (*download*)

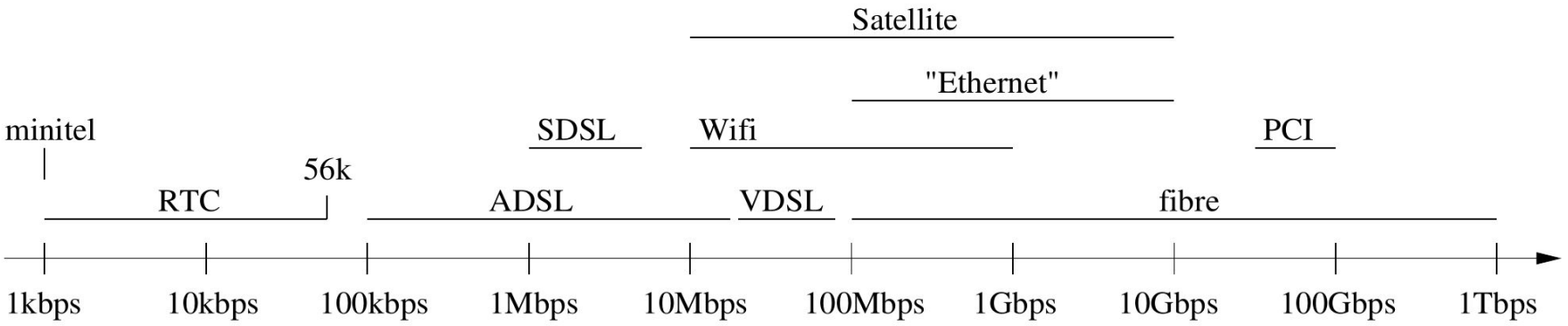
Latence : nombre de secondes que met le premier bit pour aller de la source à la destination…

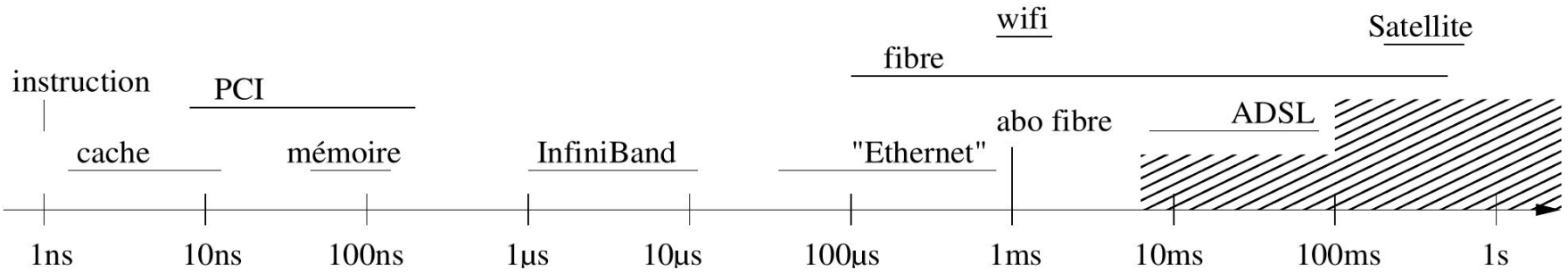
Quelques exemples de débits (en bit/s)

- modem RTC 56K, ADSL (1M à 8M), FTTH (1G)
- Ethernet (10M, 100M, 1G, 10G), ATM (155M), FDDI (100M), …
- sans-fil : IEEE 802.11 (11M à 54M)
- GSM : 3G (144K-1,9M), EDGE (64k-384k), 3G+ (3,6M, 14,4M), 4G (100M-1G), 5G (10G) …

Nota Bene : 1Ko = 10^3 octets et non 1024 = 2^10 !

Débit et Latence





Débit et Latence

Exercice : *Sneakernet*

On souhaite transférer 4 Go de données entre deux villes distantes de 100 km. Plusieurs moyens de transfert sont envisagés :

1. Un pigeon voyageur portant un carte microSD, volant à une vitesse de 60 à 110 km/h selon la direction du vent ;
2. Le réseau Internet avec ligne ADSL2 ayant un débit descendant de 8Mbit/s et montant de 1 Mbit/s.

Calculez le débit du *pigeon* ? Quel moyen de transfert est le plus performant ?

Débit et Latence

Correction

Le pigeon va mettre au plus 100 km / (60 km/h / 3600 s/h) = 6000 s pour transporter 4 Go = 4 000 Mo = 32 000 Mbit, ce qui nous donne un débit de 32 000 Mbit / 6000 s = 5.33 Mbit/s...

Le transfert en ADSL est limité par le débit montant qui est de 1 Mbit/s... Le pigeon voyageur est donc 5 fois plus rapide que l'ADSL2.

Ceci a été réellement expérimenté !

https://en.wikipedia.org/wiki/Sneakernet

Sneakers

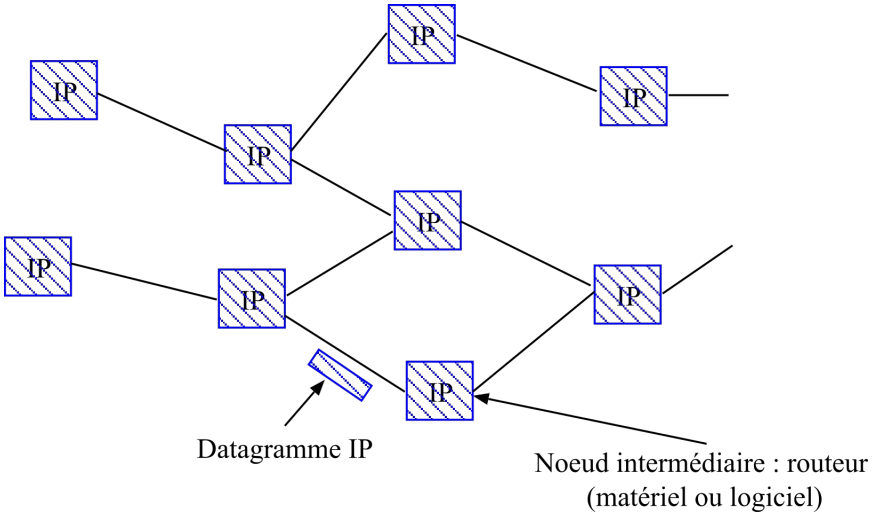
Chapitre 2 (suite)

Couche Réseau (IP)

Le Protocole IP

Internet Protocol : communication de bout en bout entre deux machines qui ne sont pas connectés directements, c'est-à-dire situées dans des réseaux différents (géographie, technologie).

Acheminement des **paquets** (ou datagrammes) à travers le réseau Internet en *best-effort*, sans garantie (non fiable), simple mais robuste (défaillance d'un routeur).



Application
(data)

Transport
(segment)

Network
(packet)

Data Link
(frame)

Physical
(bit)

Versions

- IPv4, RFC 791, sept. 1981
- IPv6, le successeur de IPv4, RFC 2460, déc. 1998

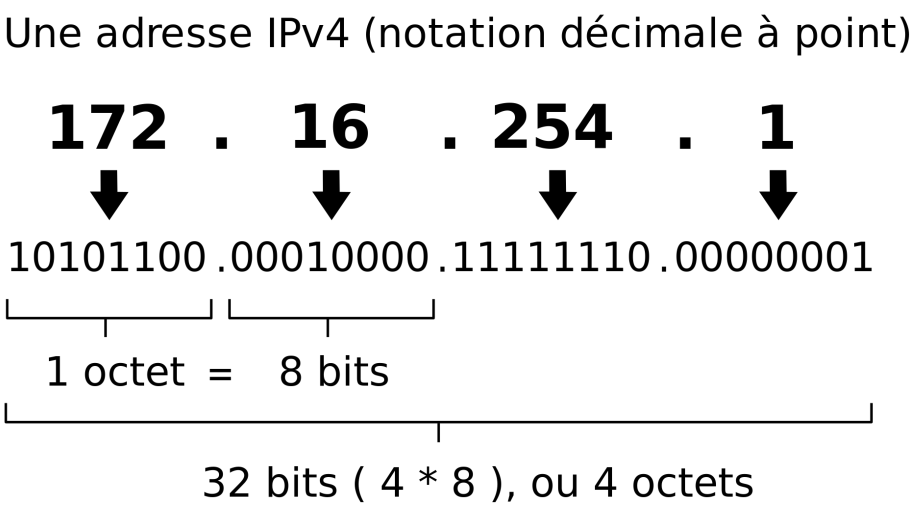
Adresse IPv4

Adressage Logique : identifier une machine unique sur Internet, indépendamment de l'adressage physique (Ethernet, ...)

Adresse IPv4 (32 bits)

- 2^32 adresses, environ 4 milliards d'adresses
- épuisement des adresses IPv4 en 2011 !

Notation



Adresse IPv6

Adresse IPv6 (128 bits)

- 2^128 adresses, soit environ 340.10^36 adresses !!!
- 667 millions de milliards d'appareils connectés sur chaque mm² de la Terre !
- 8 groupes de 2 octets, en hexadécimal (= 128 bits)
- suppression des zéros
  - un exemple

    `FEDC:0000:0000:0065:4321:0000:DEAD:BEEF`
  - on remplace les premiers blocs de 0 consécutifs par ::

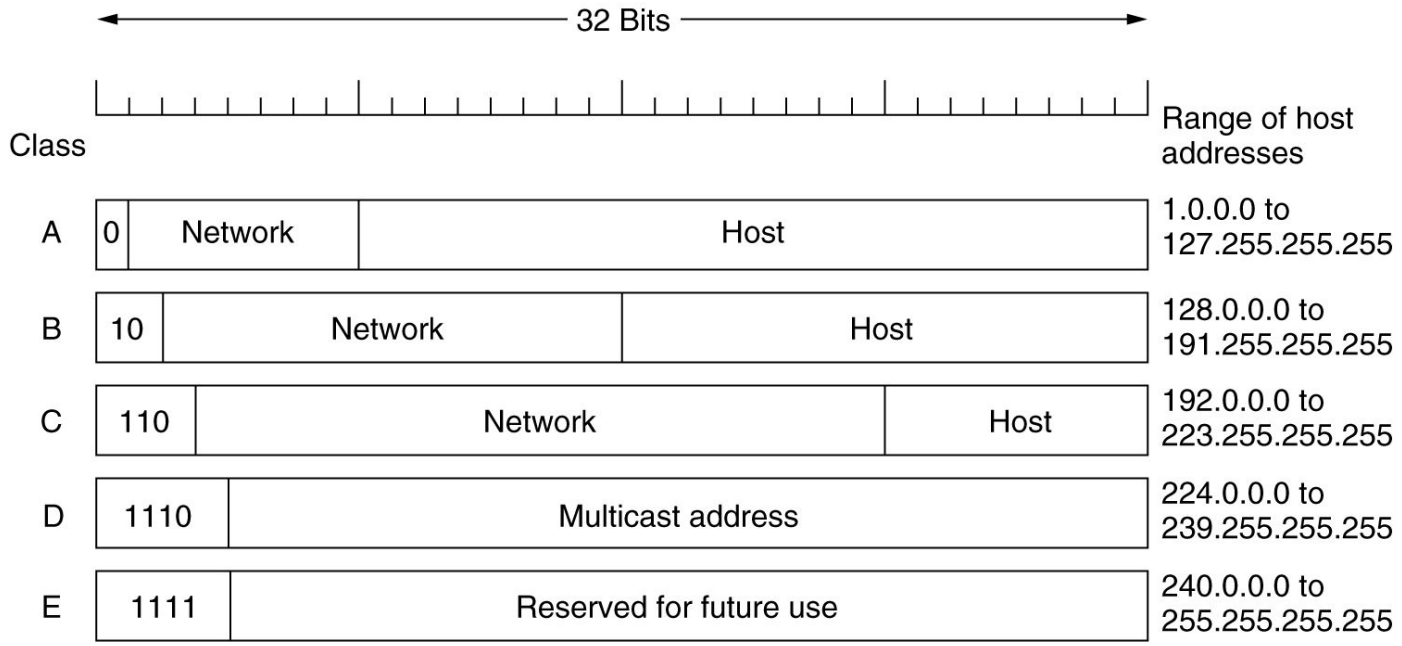
    `FEDC::0065:4321:0000:DEAD:BEEF`
  - on supprime les 0 de poids fort dans chaque bloc

    `FEDC::65:4321:0:DEAD:BEEF`
- plusieurs types d'adresses, dont l'adresse *link local* (`FE80::/10`) et l'adresse *global*, ou encore localhost (`::1`)

Classe d'Adresse IPv4

Les 5 classes historiques d'adresse IP (1990-2010)

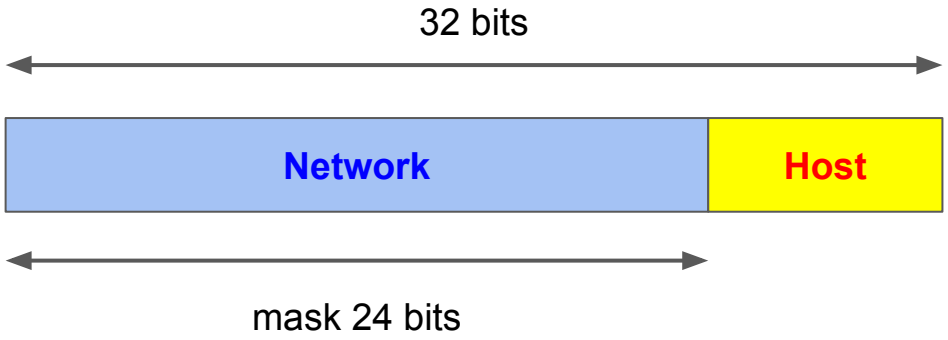
- classes générales A, B, C (unicast)
  - classe A : 8 bits network, 24 bits host (grands réseaux)
  - classe B : 16 bits network, 16 bits host (moyens réseaux)
  - classe C : 24 bits network, 8 bits host (petits réseaux)
- classe D (multicast)
- classe E (réservé pour un usage futur)



Adresse de Réseau

Notion de Masque

- nombre bits séparant la partie *Network* et de la partie *Host*
- toutes les hôtes d'un réseau ont les même bits sur la partie *Network*



Exemple : 192.168.10.7/24 (notation décimale à point)

- adresse du masque : les bits de la partie *Network* à 1, les autres à 0
- adresse du réseau : mettre les bits de la partie *Host* à 0
- adresse de diffusion (*broadcast*) : mettre les bits de la partie *Host* à 1

```
Address:   192.168.10.7/24       11000000.10101000.00001010. 00000111
Netmask:   255.255.255.0 = /24   11111111.11111111.11111111. 00000000
Network:   192.168.10.0/24       11000000.10101000.00001010. 00000000
Broadcast: 192.168.10.255        11000000.10101000.00001010. 11111111
```

Adresse de Réseau

Les adresses spéciales

- Adresse locale (loopback) : 127.0.0.0/8 (127.0.0.1 ou ::1 ou localhost)
- Adresse de ce réseau : 0.0.0.0/8
- Adresse de diffusion : 255.255.255.255/32
- Adresse du routeur (par convention) : adresse de diffusion – 1

Les adresses privés

- 10.0.0.0 /8
- 172.16.0.0 /12
- 192.168.0.0 /16

Ces adresses ne peuvent pas être routées sur Internet.

Leur utilisation par un réseau privé est encouragée pour éviter de réutiliser les adresses publiques enregistrées.

Adresse de Réseau

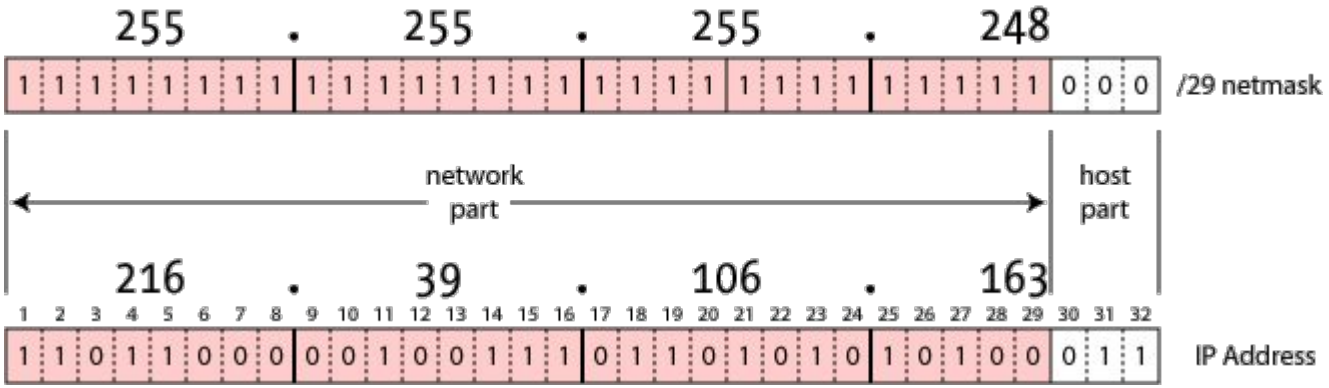
Exercice : Complétez le tableau ci-dessous.

Adresse IP de l'hôte	Adresse du Réseau	Adresse Hôte	Adresse de Broadcast	Masque du Réseau
192.168.10.7/24	192.168.10.0	.7	192.168.10.255	255.255.255.0
216.14.55.137/24				
123.1.1.15/8				
175.12.239.244/16				
216.39.106.163/29				

Adresse de Réseau

Correction

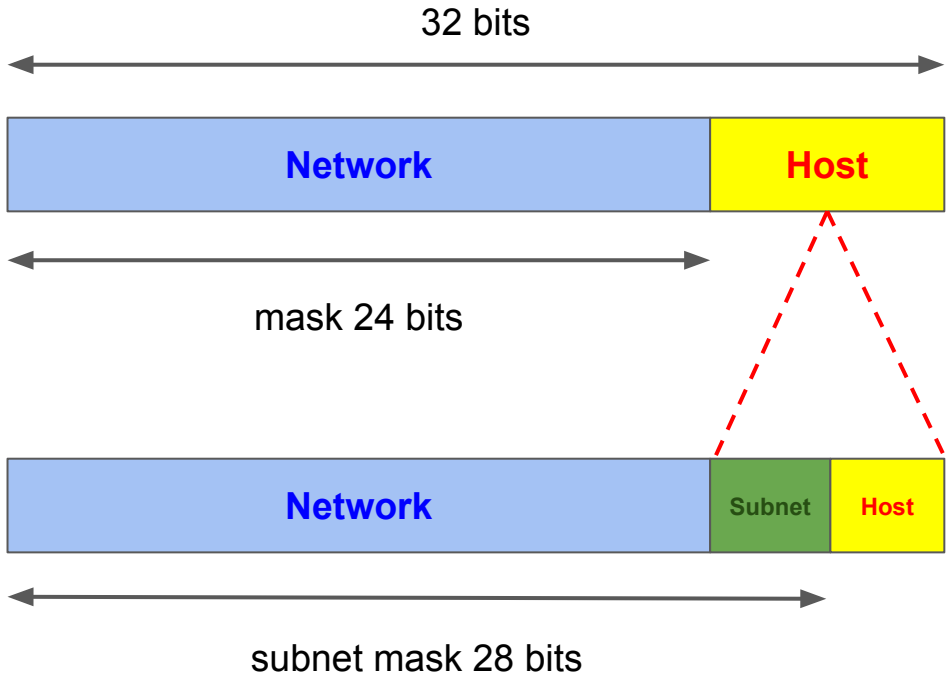
Adresse IP de l'hôte	Adresse du Réseau	Adresse Hôte	Adresse de Broadcast	Masque du Réseau
192.168.10.7/24	192.168.10.0	.7	192.168.10.255	255.255.255.0
216.14.55.137/24	216.14.55.0	.137	216.14.55.255	255.255.255.0
123.1.1.15/8	123.0.0.0	.1.1.15	123.255.255.255	255.0.0.0
175.12.239.244/16	175.12.0.0	.239.244	175.12.255.255	255.255.0.0
216.39.106.163/29	216.39.106.160	.3	216.39.106.167	255.255.255.248



Sous-Réseaux

Découpage d'un réseau en plusieurs sous-réseaux

- Adresse IP découpée en trois parties : *network*, *subnet*, *host*
- Une partie des bits de *host* sert à identifier le sous-réseau (*subnet*)
- Masque de sous-réseau



RFC : https://tools.ietf.org/html/rfc1860 et https://tools.ietf.org/html/rfc1878

Sous-Réseaux

Exercice : Dans un réseau 193.51.199.0/24, on souhaite constituer 5 sous-réseaux (de même taille).

- Combien de bits sont nécessaires pour coder ces sous-réseaux ?
- Combien de machines trouve-t-on dans chaque sous réseau ?
- Quel est le masque de réseau et de sous-réseau ?
- A quel adresse de sous-réseau appartient la machine 193.51.199.67 ?
- Donner l'adresse de diffusion correspondant à ce sous-réseau ?
- Quel sont les adresses des autres sous-réseaux ?

```
orel@prout:~$ ipcalc 193.51.199.0/24
Address:   193.51.199.0          11000001.00110011.11000111. 00000000
Netmask:   255.255.255.0 = 24    11111111.11111111.11111111. 00000000
Wildcard:  0.0.0.255             00000000.00000000.00000000. 11111111
=>
Network:   193.51.199.0/24       11000001.00110011.11000111. 00000000
HostMin:   193.51.199.1          11000001.00110011.11000111. 00000001
HostMax:   193.51.199.254        11000001.00110011.11000111. 11111110
Broadcast: 193.51.199.255        11000001.00110011.11000111. 11111111
Hosts/Net: 254                    Class C
```

Sous-Réseaux

Correction

- Avec 3 bits, on peut coder un maximum de 2^3=8 sous-réseaux. En effet, 2 bits ne sont pas suffisants pour coder 5 sous-réseaux à choisir parmi 000, 001, 010, 011, 100, 101, 110 et 111.
- Il reste 5 bits pour la partie *host* ; le nombre de machines = 2^5-2=30. On retire les adresses min & max, qui sont réservés…
- Le masque du réseau /24 correspond à l'adresse 255.255.255.0.
- Le masque du sous-réseau est 255.255.255.224 car 224 = (1110 0000) en binaire
- Adresse du réseau : 193.51.199.67 & 255.255.255.0 = 193.51.199.0 (avec & l'opérateur "ET" binaire)
- Adresse du sous-réseau : 193.51.199.67 & 255.255.255.224 = 193.51.199.X avec X = 67 & 224 = 64 (en binaire : 010 00011 & 111 00000 = 010 0000) ; adresse sous-réseau = 193.51.199.64/27
- Adresse de diffusion du sous-réseau : 193.51.199.Y avec Y = 010 11111 = 95

Sous-Réseaux

Correction (suite)

- Adresses de sous-réseaux :

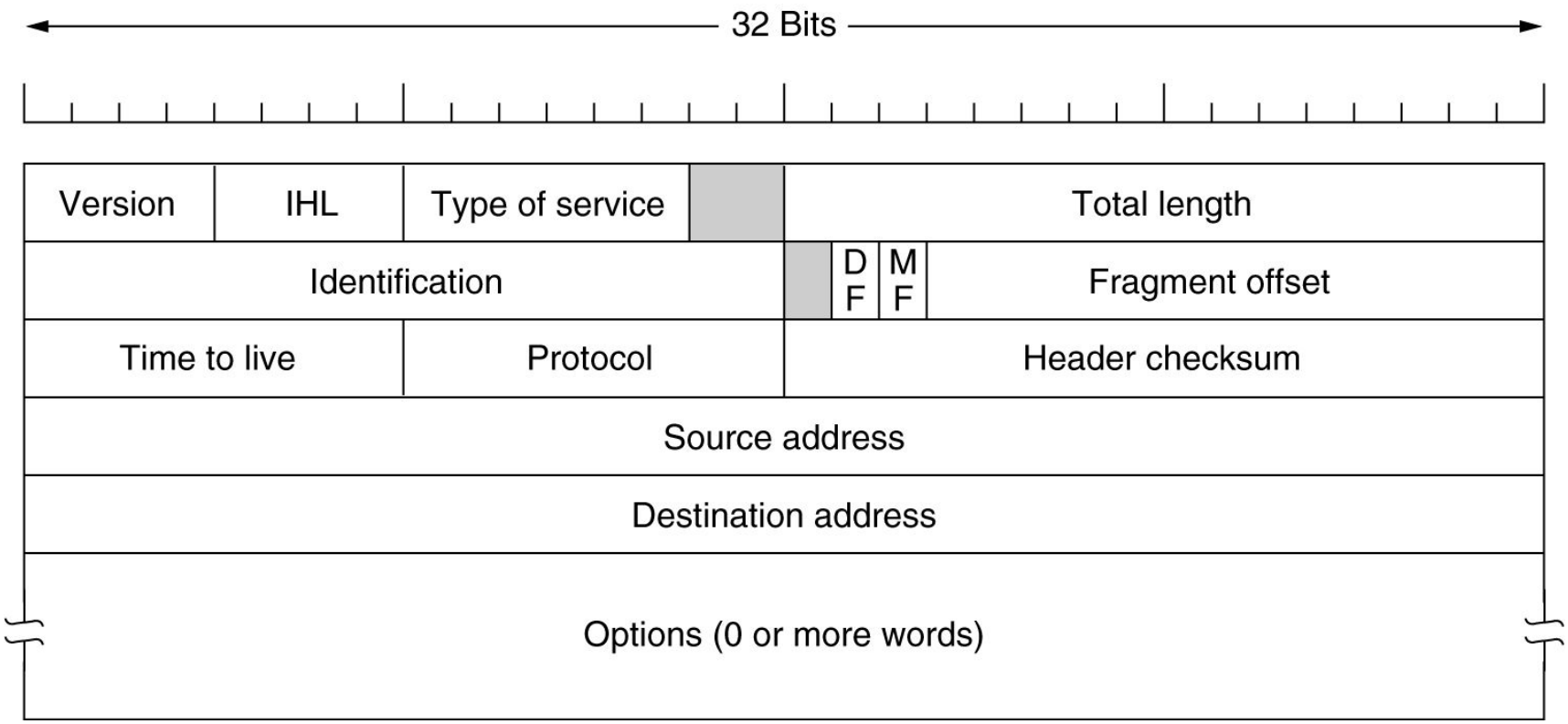
```
193.51.199.0/27
193.51.199.32/27
193.51.199.64/27
193.51.199.96/27
193.51.199.128/27
193.51.199.160/27 (unused)
193.51.199.192/27 (unused)
193.51.199.224/27 (unused)
```

- Pour calculer 5 sous-réseaux de taille 30 machines :

```
$ ipcalc 193.51.199.0/24 -s 30 30 30 30 30
```

```
1. Requested size: 30 hosts
Netmask:   255.255.255.224 = 27 11111111.11111111.11111111.111 00000
Network:   193.51.199.0/27      11000001.00110011.11000111.000 00000
HostMin:   193.51.199.1         11000001.00110011.11000111.000 00001
HostMax:   193.51.199.30        11000001.00110011.11000111.000 11110
Broadcast: 193.51.199.31        11000001.00110011.11000111.000 11111
Hosts/Net: 30                    Class C

2. Requested size: 30 hosts
Netmask:   255.255.255.224 = 27 11111111.11111111.11111111.111 00000
Network:   193.51.199.32/27     11000001.00110011.11000111.001 00000
HostMin:   193.51.199.33        11000001.00110011.11000111.001 00001
HostMax:   193.51.199.62        11000001.00110011.11000111.001 11110
Broadcast: 193.51.199.63        11000001.00110011.11000111.001 11111
Hosts/Net: 30                    Class C

3. Requested size: 30 hosts
Netmask:   255.255.255.224 = 27 11111111.11111111.11111111.111 00000
Network:   193.51.199.64/27     11000001.00110011.11000111.010 00000
HostMin:   193.51.199.65        11000001.00110011.11000111.010 00001
HostMax:   193.51.199.94        11000001.00110011.11000111.010 11110
Broadcast: 193.51.199.95        11000001.00110011.11000111.010 11111
Hosts/Net: 30                    Class C

4. Requested size: 30 hosts
Netmask:   255.255.255.224 = 27 11111111.11111111.11111111.111 00000
Network:   193.51.199.96/27     11000001.00110011.11000111.011 00000
HostMin:   193.51.199.97        11000001.00110011.11000111.011 00001
HostMax:   193.51.199.126       11000001.00110011.11000111.011 11110
Broadcast: 193.51.199.127       11000001.00110011.11000111.011 11111
Hosts/Net: 30                    Class C

5. Requested size: 30 hosts
Netmask:   255.255.255.224 = 27 11111111.11111111.11111111.111 00000
Network:   193.51.199.128/27    11000001.00110011.11000111.100 00000
HostMin:   193.51.199.129       11000001.00110011.11000111.100 00001
HostMax:   193.51.199.158       11000001.00110011.11000111.100 11110
Broadcast: 193.51.199.159       11000001.00110011.11000111.100 11111
Hosts/Net: 30                    Class C
```

En-Tête du Paquet IPv4



En-Tête du Paquet IPv4

- **Version** : 4 [4 bits]
- **Internet Header Length** : longueur de l'en-tête en mot de 4 octets [4 bits]
- **Type of Service** (ToS) : qualité de service (fiabilité, débit, …) [8 bits]
- **Identification** : identifiant d'un fragment pour leur rassemblage [16 bits]
- **Flags** : DF (Don't Fragment) / MF (More Fragment) [3 bits]
- **Fragment Offset** : position dans le paquet initial en mot de 8 octets [13 bits]
- **Time To Live** (TTL) : temps de vie maximal en *hop* [8 bits]
- **Protocol** : protocole encapsulé dans le paquet (ICMP, UDP, TCP, ...) [8 bits]
- **Header Checksum** : contrôle d'erreurs de l'en-tête [16 bits]
- **Source Address** [32 bits]
- **Destination Address** [32 bits]
- **Options** : usage peu fréquent...

Exercice : Quelle est la taille minimale & maximale de l'en-tête ?

*IHL sur 4 bits compris entre 5 (sans options) et 15 et donc l'en-tête a une taille entre 5*4=20 octets et 15*4=60 octets.*

Exemple d'une Trame Ethernet / IP

Un exemple de trame Ethernet II

```
BB BB BB BB BB BB AA AA AA AA AA AA 08 00  45 00   ..............
00 20 00 01 00 00 40 FF F8 8A C0 A8 00 01 C0 A8   ..............
00 02 48 45 4C 4C 4F 20 57 4F 52 4C 44 21  XX XX   ..HELLO WORLD!..
XX XX XX XX XX XX XX XX XX XX XX XX  ZZ ZZ ZZ ZZ   ..............
```

```
Ethernet / IP / RAW
```

Exercice

- Quel est l'adresse MAC de la source et de la destination ?
- Quelle est la taille de cette trame ?
- Quel protocole est encapsulé dans la trame ? Détaillez.
- Que repésente les octets `XX` et `ZZ` à la fin ?

Exemple d'une Trame Ethernet / IP

Correction

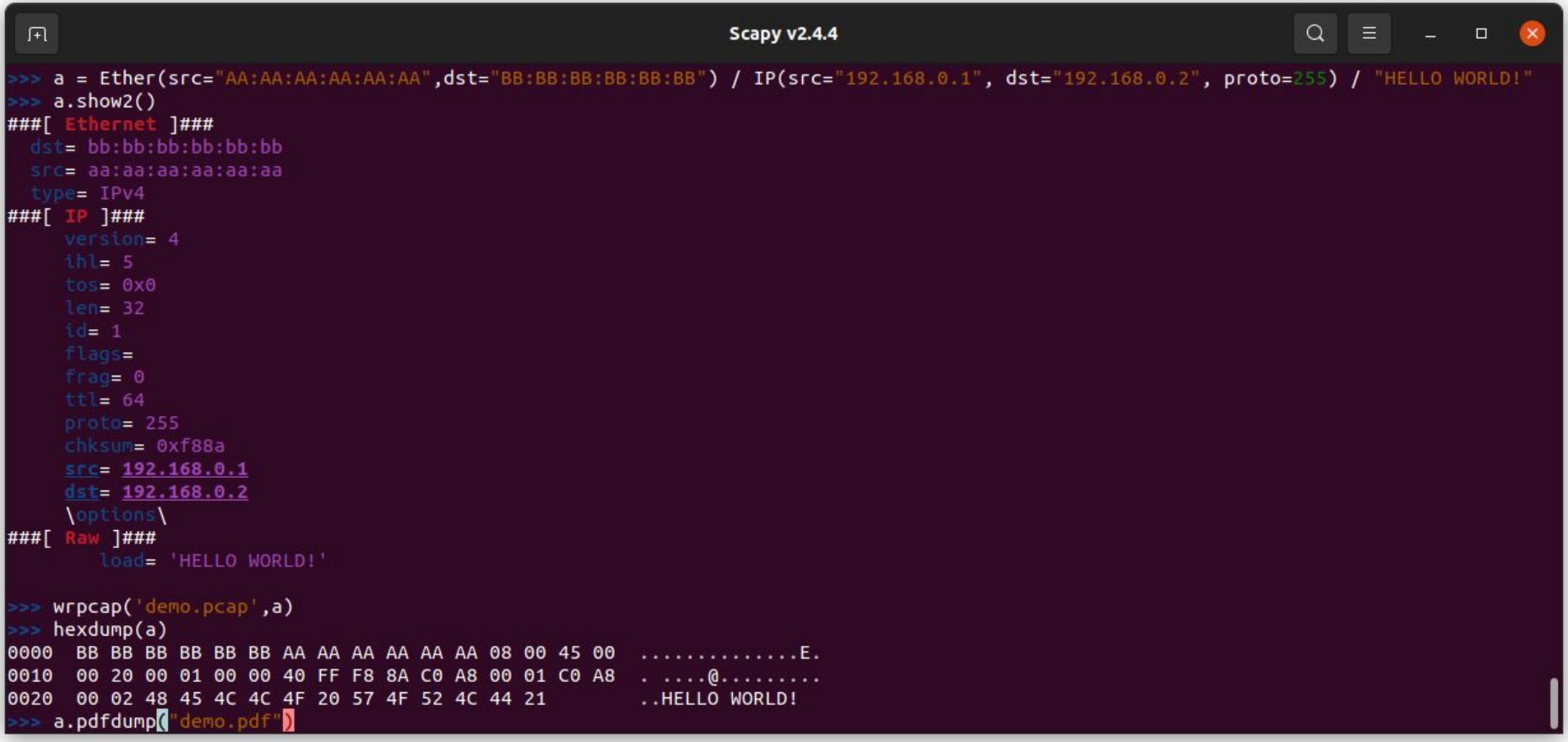
```
BB BB BB BB BB BB AA AA AA AA AA AA 08 00  45 00   ..............
00 20 00 01 00 00 40 FF F8 8A C0 A8 00 01 C0 A8   ..............
00 02 48 45 4C 4C 4F 20 57 4F 52 4C 44 21  XX XX   ..HELLO WORLD!..
XX XX XX XX XX XX XX XX XX XX XX XX  ZZ ZZ ZZ ZZ   ..............
```

```
Ethernet / IP / RAW
```

- Trame Ethernet de taille minimale 64 octets (4 lignes de 16 octets)
- @MAC source = `AA:AA:AA:AA:AA`
- @MAC destination = `BB:BB:BB:BB:BB`
- Type de protocole : IP (`08 00`)
- L'en-tête du paquet IP nous indique que c'est la version 4 de IP.
- La longueur de l'en-tête est de 20 octets (IHL=5)
- Le paquet IP contient un texte brut `HELLO WORLD!` (12 octets)
- Les 4 octets `ZZ` à la fin sont la checksum CRC.
- Les octets `XX` sont en fait des octets de bourrage (ou *padding*)

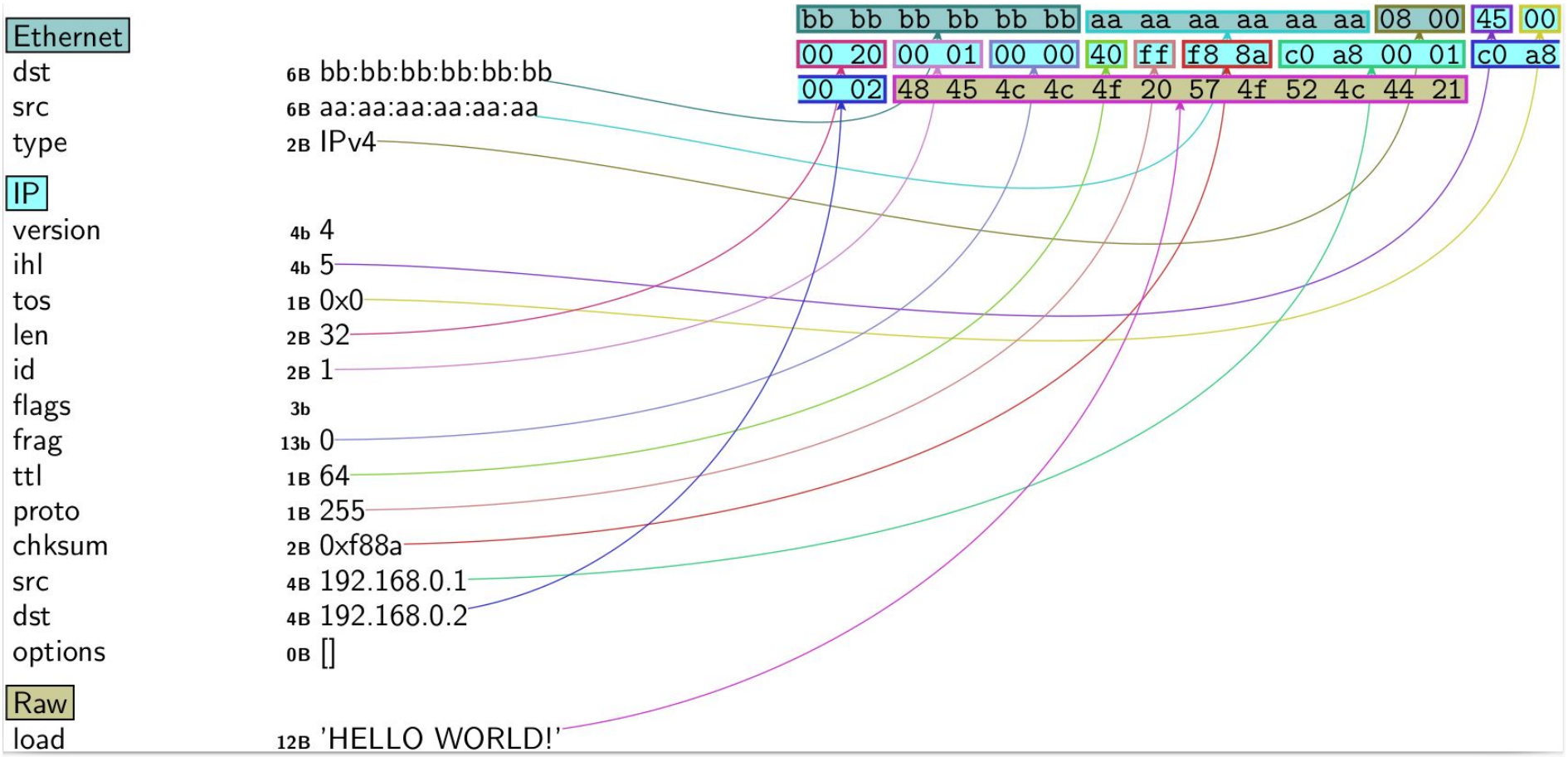
Exemple d'une Trame Ethernet / IP

Génération d'une trame "à la main" avec *Scapy3*



⇒ https://scapy.readthedocs.io/en/latest/usage.html#interactive-tutorial

Exemple d'une Trame Ethernet / IP



ICMP

Internet Control Message Protocol (ICMP), RFC 792

Accompagne IP pour gérer les erreurs et propager des informations de routage...

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo request	Ask a machine if it is alive
Echo reply	Yes, I am alive
Timestamp request	Same as Echo request, but with timestamp
Timestamp reply	Same as Echo reply, but with timestamp

Exemple du ping : envoi d'une requête ICMP *“echo request”* et attente de la réponse *“echo reply”*

ARP

Problèmatique : Au sein d'un réseau local, les adresses MAC sont utilisées pour communiquer entre les machines (trame Ethernet). Mais comment découvrir l'adresse MAC du destinataire ?

ARP (Address Resolution Protocol) : protocole de résolution des adresses MAC à partir des adresses IP, RFC 826.

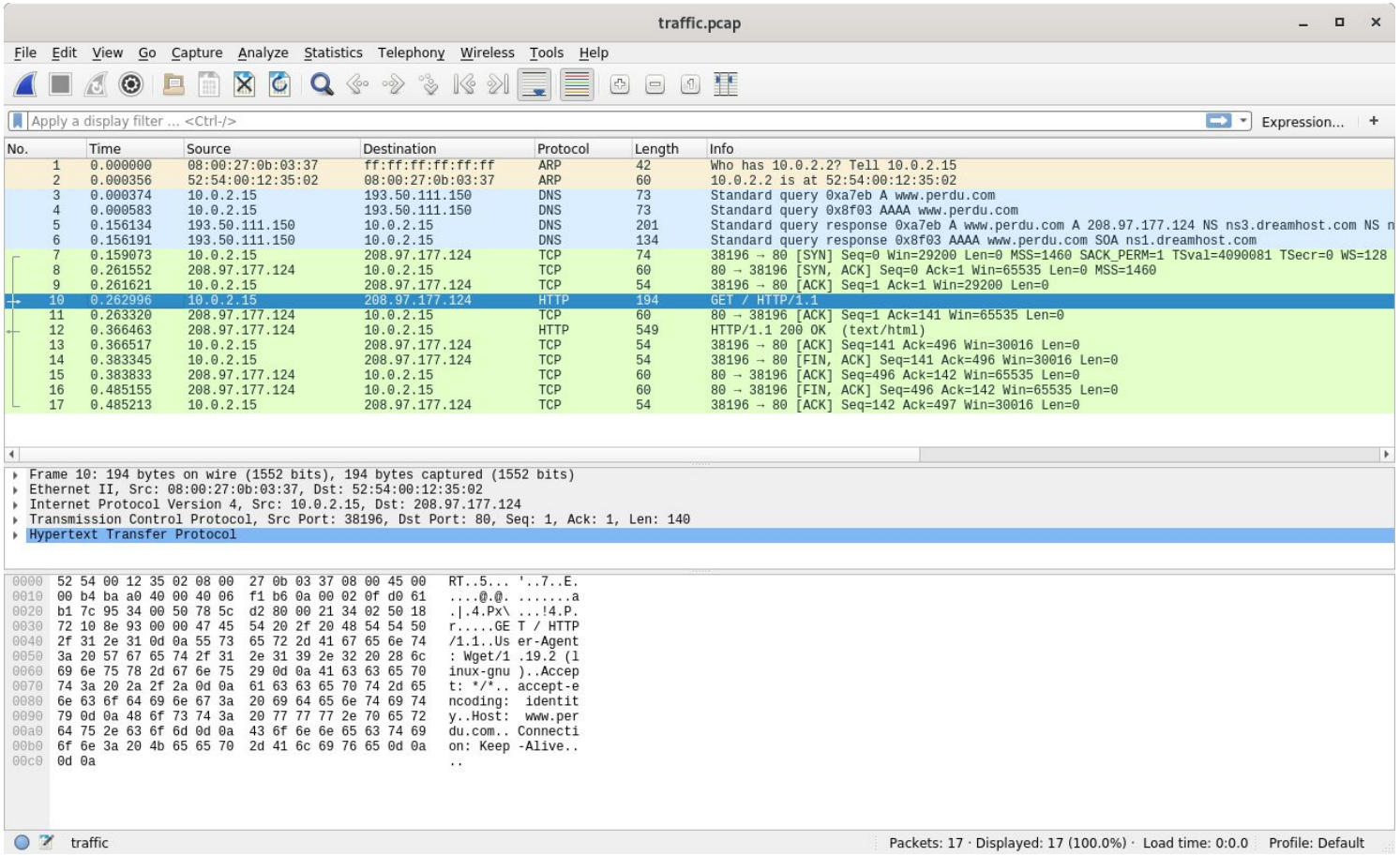
- La source diffuse la requête ARP "WHO HAS @IP?" en broadcast Ethernet dans le réseau local (FF:FF:FF:FF:FF:FF)
- La machine @IP répond avec son @MAC
- La machine source enregistre le résultat dans le cache ARP

Outils : affichage du cache ARP

```
$ arp -n
(192.168.2.109)   00:23:69:15:28:51 [ether] wlan0
(10.20.30.100)    00:22:19:dd:0b:65 [ether] eth0
```

Wireshark

Outil permettant l'analyse et la capture de trames réseau…



Démo : analyse d'un paquet IP/ICMP avec l'exemple [ping.pcap]

Outils

ifconfig : affichage des interfaces réseaux… [Linux]

```
auesnard@buffet:~$ /sbin/ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9000
        inet 10.0.204.4  netmask 255.255.255.0  broadcast 10.0.204.255
        inet6 fe80::24e:1ff:fec4:7f4  prefixlen 64  scopeid 0x20<link>
        inet6 2001:660:6101:800:204::4  prefixlen 80  scopeid 0x0<global>
        ether 00:4e:01:c4:07:f4  txqueuelen 1000  (Ethernet)
        RX packets 3168136  bytes 11489530817 (10.7 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2066596  bytes 280233134 (267.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 20  memory 0x94300000-94320000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Boucle locale)
        RX packets 127449  bytes 9066029 (8.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 127449  bytes 9066029 (8.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Outils

ping : tester si machine est en vie (requête *echo* du protocole **IP/ICMP**)

```
$ ping 10.0.204.4
PING 10.0.204.4 (10.0.204.4) 56(84) bytes of data.
64 bytes from 10.0.204.4: icmp_seq=1 ttl=63 time=0.202 ms
64 bytes from 10.0.204.4: icmp_seq=2 ttl=63 time=0.206 ms
64 bytes from 10.0.204.4: icmp_seq=3 ttl=63 time=0.200 ms
64 bytes from 10.0.204.4: icmp_seq=4 ttl=63 time=0.213 ms
^C
--- 10.0.204.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 57ms
rtt min/avg/max/mdev = 0.200/0.205/0.213/0.011 ms
```

Le RTT (Round-Trip Time) mesure le temps d'aller-retour du paquet…

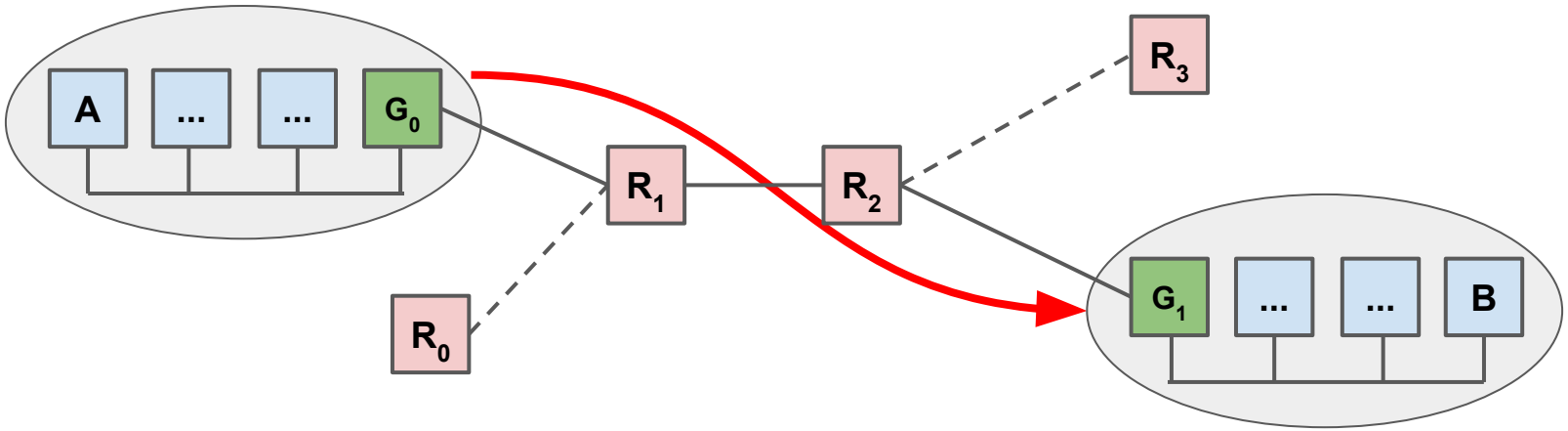
Chapitre 3

Routage

Routage

Principe : Mécanisme par lequel un paquet IP est acheminé d'un expéditeur (A) jusqu'à son destinataire (B), en s'appuyant sur les noeuds intermédiaires (Gi, Ri) du réseau Internet.

Les différents noeuds du réseau : les hôtes (A,B), les passerelles ou *gateway* (G_i) et les routeurs (R_i)



Routage statique & dynamique : manuel, DHCP, OSPF, BGP, ...

Table de Routage

Chaque nœud du réseau a besoin des informations sur le noeud suivant (next hop) vers lequel il doit envoyer un paquet pour atteindre la destination finale…

Exemple : table de routage d'un hôte

```
$ route -n
DestAddr        Gateway          GenMask          Flags    Interface
127.0.0.1       *                0.0.0.0          UH       lo
192.168.0.0     *                255.255.255.0    U        eth0
default         192.168.0.254    0.0.0.0          UG       eth0
```

Il faut distinguer :

- les **routes directes** vers un réseau (ou une machine) ;
- les **routes indirectes**, dont les **routes spécifiques** vers un réseau et la **route par défaut**.

Table de Routage

Un exemple plus compliqué :

```
$ route -n
DestAddr        Gateway         GenMask         Flags   Interface
127.0.0.1       *               0.0.0.0         UH      lo
147.210.10.0    *               255.255.255.0   U       eth1
147.210.0.0     *               255.255.255.0   U       eth0
10.0.0.0        147.210.0.100   255.255.0.0     UG      eth0
default         147.210.0.254   0.0.0.0         UG      eth0
```

Légende des Flags

- U : la route est active (Up)
- G : route indirecte qui passe par un routeur (Gateway) ; sinon route directe (pas G)
- H : l'adresse destination est une adresse de machine (Host) ; sinon l'adresse destination est celle d'un réseau (pas H)

Exercice

- Donner la signification de chaque ligne de la table de routage…
- En déduire la configuration réseau de cette machine…

Table de Routage

Correction

La machine a la configuration suivante :

- interface *eth0* avec une adresse dans le réseau 147.210.0.0/24
- interface *eth1* avec une adresse dans le réseau 147.210.10.0/24
- Il y a dans le réseau 147.210.0.0/24 un passerelle (.100) vers le réseau 10.0.0.0/16.
- La passerelle par défaut est 147.210.0.254… → porte de sortie vers Internet

```
$ route -n
DestAddr        Gateway         GenMask         Flags   Interface
127.0.0.1       *               0.0.0.0         UH      lo
147.210.10.0    *               255.255.255.0   U       eth1
147.210.0.0     *               255.255.255.0   U       eth0
10.0.0.0        147.210.0.100   255.255.0.0     UG      eth0
default         147.210.0.254   0.0.0.0         UG      eth0
```

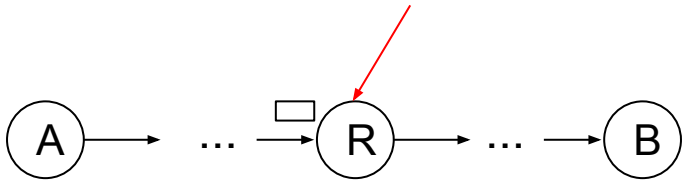
Algorithme de Routage

Algorithme exécuté sur chaque nœud intermédiaire (R)

Supposons que *DestFinal* est l'adresse de destination (B) du paquet à transmettre, *DestAddr* est une adresse dans la table de routage.

Pour chaque ligne dans la table de routage :

```
// host route
if (DestAddr = DestFinal)
     envoyer le paquet via la route directe ou indirecte vers le next hop
// net route
else if (DestAddr = DestFinal & GenMask)
     envoyer le paquet via la route directe ou indirecte vers le next hop
// default route
else
     envoyer au next hop de la route par défaut
```



Outils

traceroute : Outil qui permet de retrouver le chemin d'un paquet sur Internet.

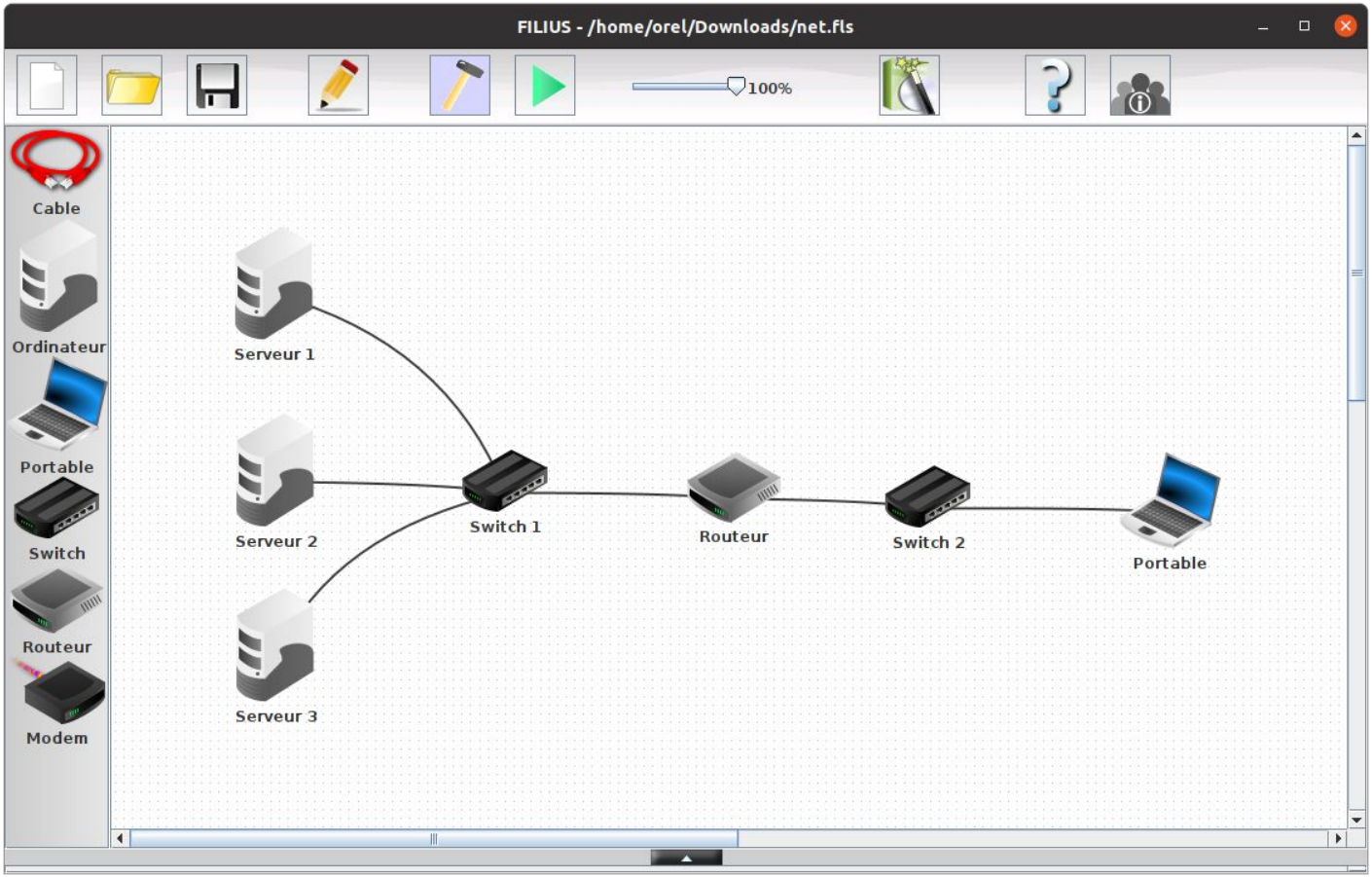
Exemple : route vers le serveur Web du LaBRI

```
$ traceroute www.labri.fr

traceroute to www.labri.fr (147.210.8.59), 30 hops max, 60 byte packets

 1  _gateway (192.168.1.254) 0.931 ms
 2  p25.socabim.isdnet.net (194.149.169.41) 35.552 ms
 3  free-paris-por1.bb.ip-plus.net (193.5.122.58) 36.783 ms
 4  193.51.187.208 (193.51.187.208) 34.955 ms
 5  te2-2-lyon2-rtr-021.noc.renater.fr (193.51.177.43) 49.554
 6  te0-0-0-5-lyon1-rtr-001.noc.renater.fr (193.51.177.216) 50.311 ms
 7  te0-0-0-0-ren-nr-clermont-rtr-091.noc.renater.fr (193.51.177.227) 49.938 ms
 8  te0-0-0-0-ren-nr-bordeaux-rtr-091.noc.renater.fr (193.51.177.84) 51.531 ms
 9  reaumur-vl10-gi8-1-bordeaux-rtr-021.noc.renater.fr (193.51.183.37) 49.163 ms
10  147.210.246.205 (147.210.246.205) 48.959 ms
11  www3.labri.fr (147.210.8.59) 54.947 ms
```

La route est parfois longue ! On part de Talence (chez moi), on monte à Paris avec le *backbone* de Free, avant de revenir pas le réseau Renater par Lyon, Clermont, puis retour sur Talence !

Simulateur Réseau

Mise en oeuvre avec Filius (Linux / Windows / …)



Installation (site allemand) : https://www.lernsoftware-filius.de/Herunterladen
Documentation (anglais) : https://www.lernsoftware-filius.de/downloads/Introduction_Filius.pdf

Chapitre 4

Couche Transport

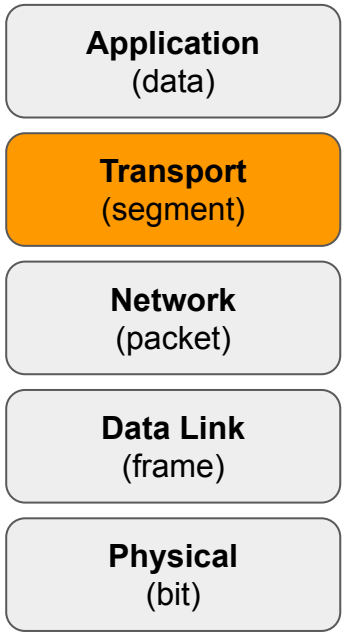
La Couche Transport

La couche réseau (IP)

- Communication de bout-en-bout entre deux machines sur Internet
- Transfert de paquet en “best-effort” (non fiable)

La couche transport

- Communication de bout-en-bout entre deux applications (processus).



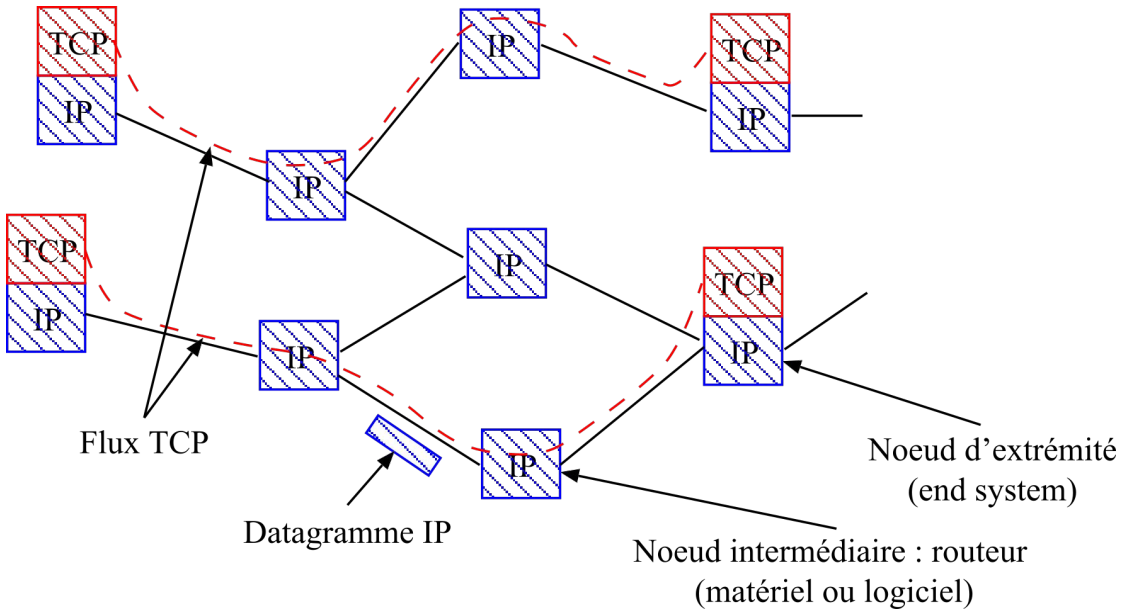
Les deux principaux protocoles de transport

- **TCP** (Transmission Control Protocol) : orienté connexion, fiable.
- **UDP** (User Datagram Protocol) : sans connexion, non fiable, rapide.

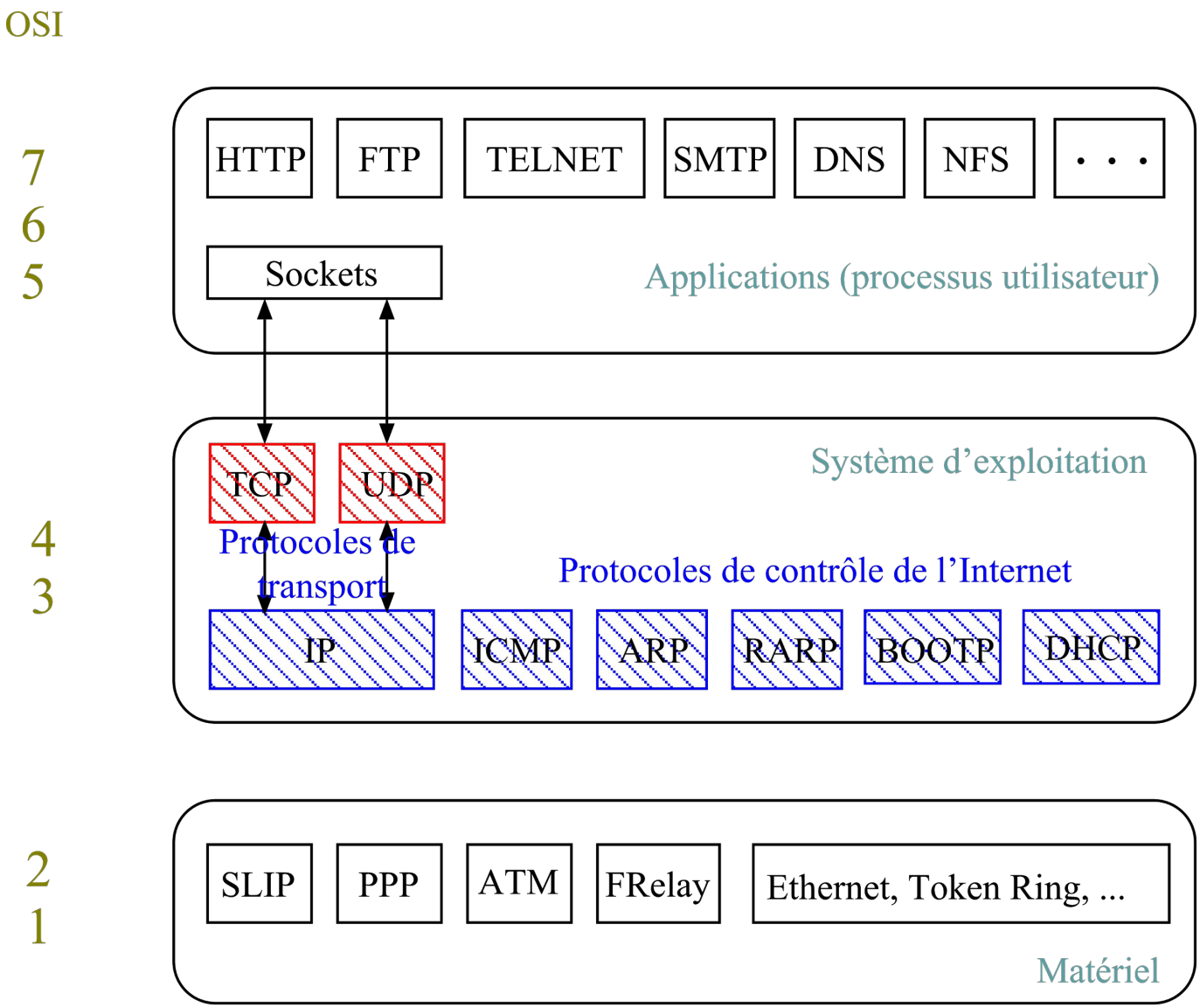
Le Protocole TCP

Caractéristiques

- uniquement présent aux extrémités
- conversation bidirectionnelle en mode connecté
- transport fiable de segments (séquencement)
- protocole complexe : retransmission, gestion des erreurs, congestion, …

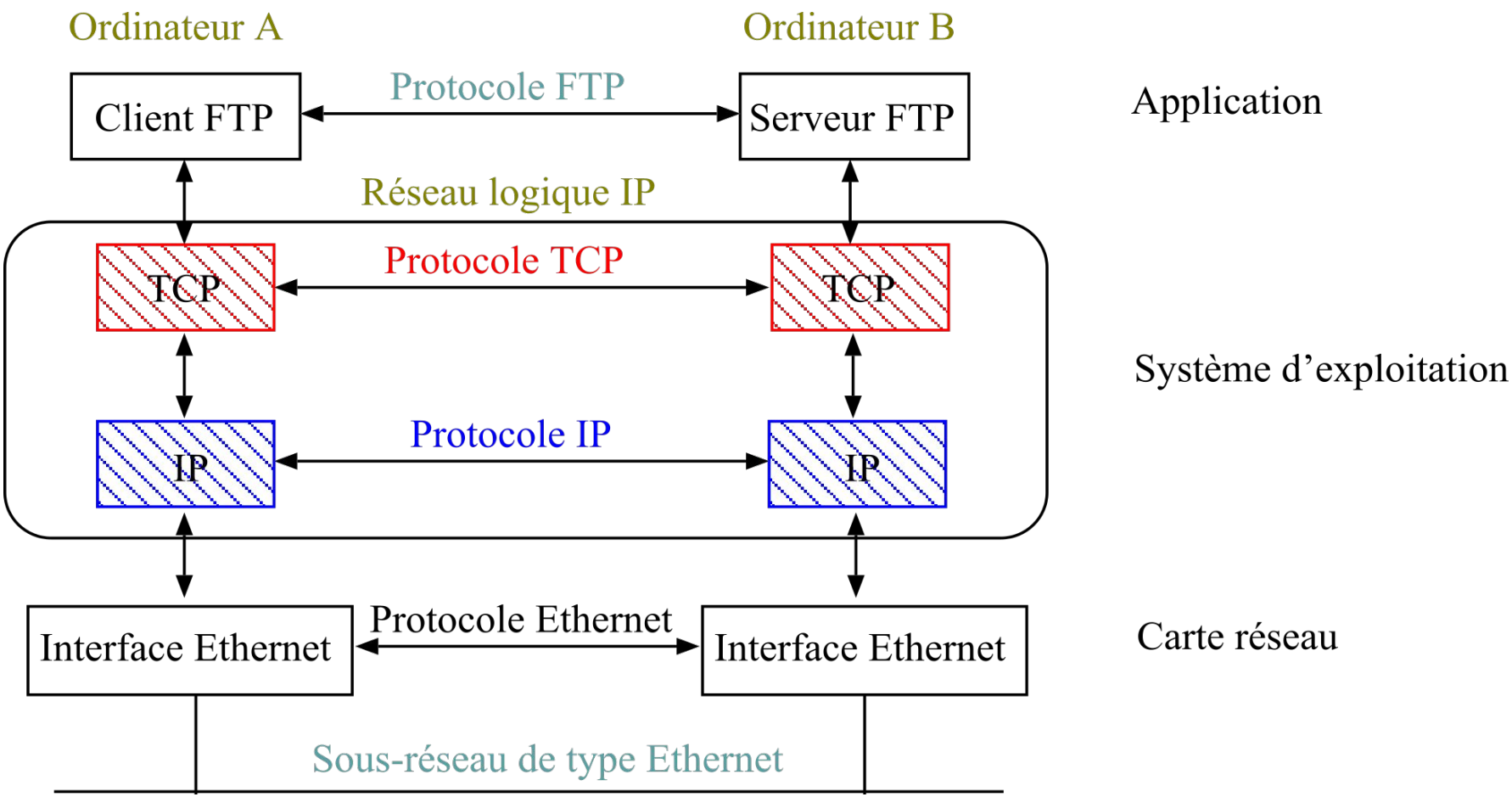


Pile de Protocoles



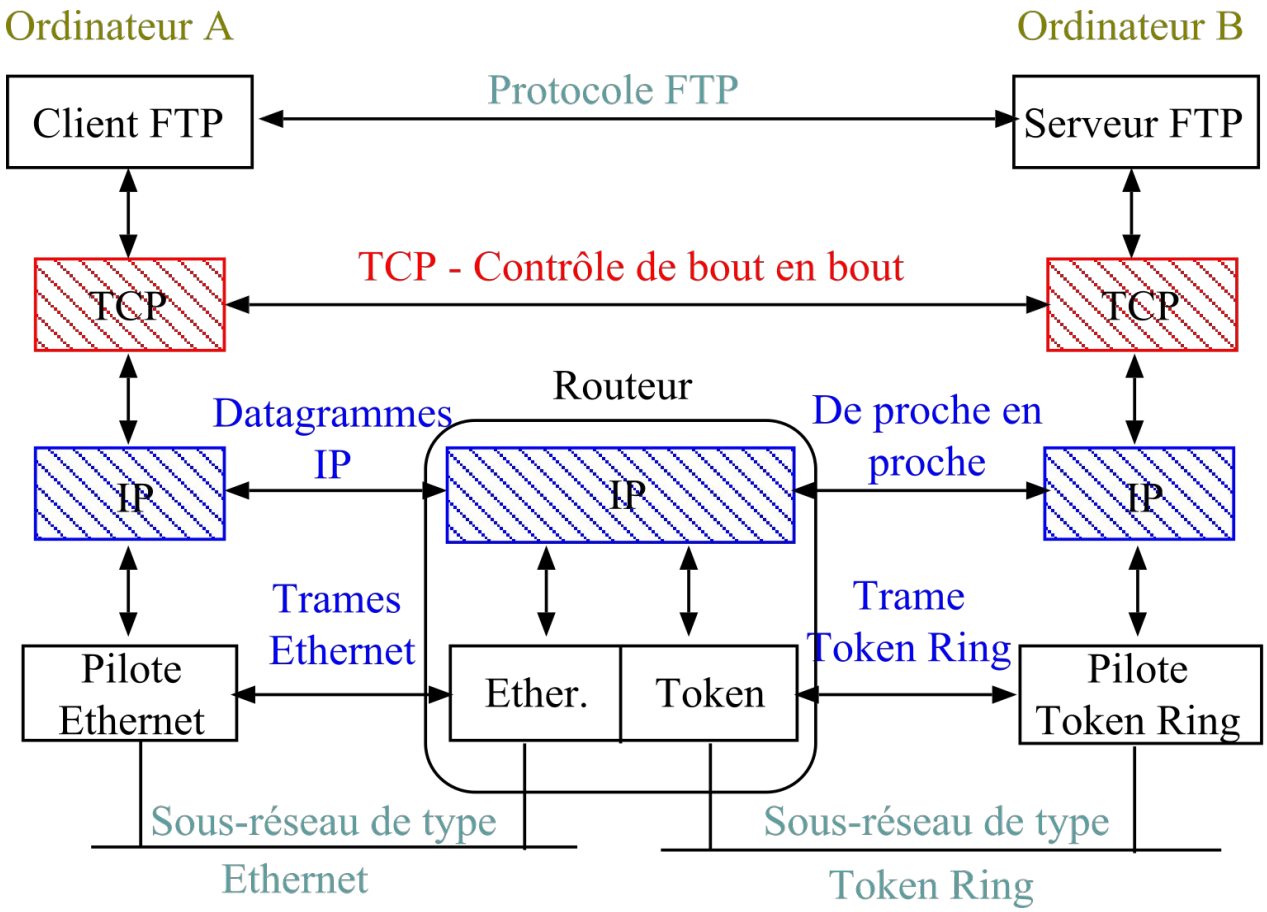
Pile de Protocoles

Deux machines dans un même réseau local et homogène...



Pile de Protocoles

Deux machines dans des réseaux distants et hétérogènes...

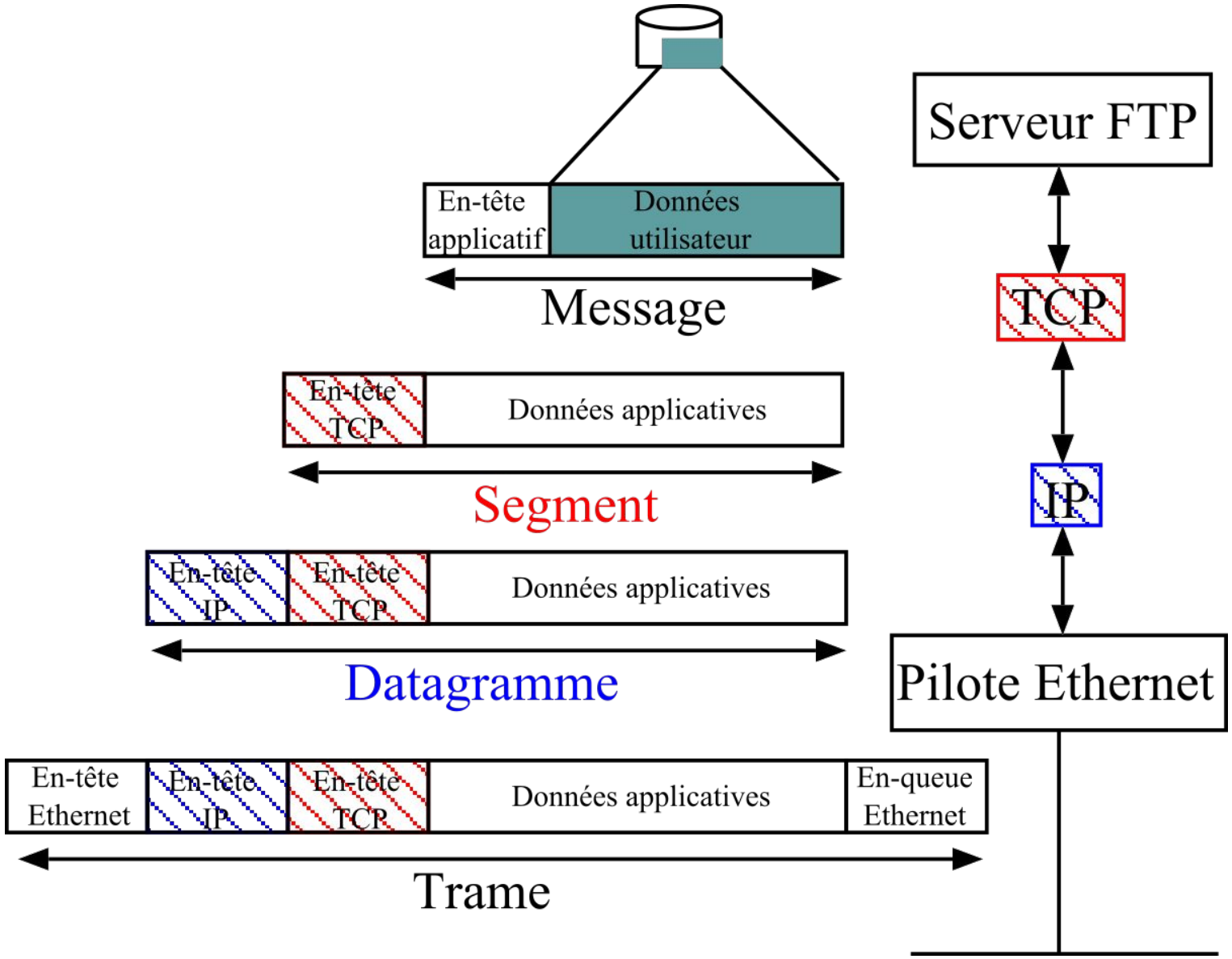


Application

Système d'exploitation

Carte réseau

Encapsulation



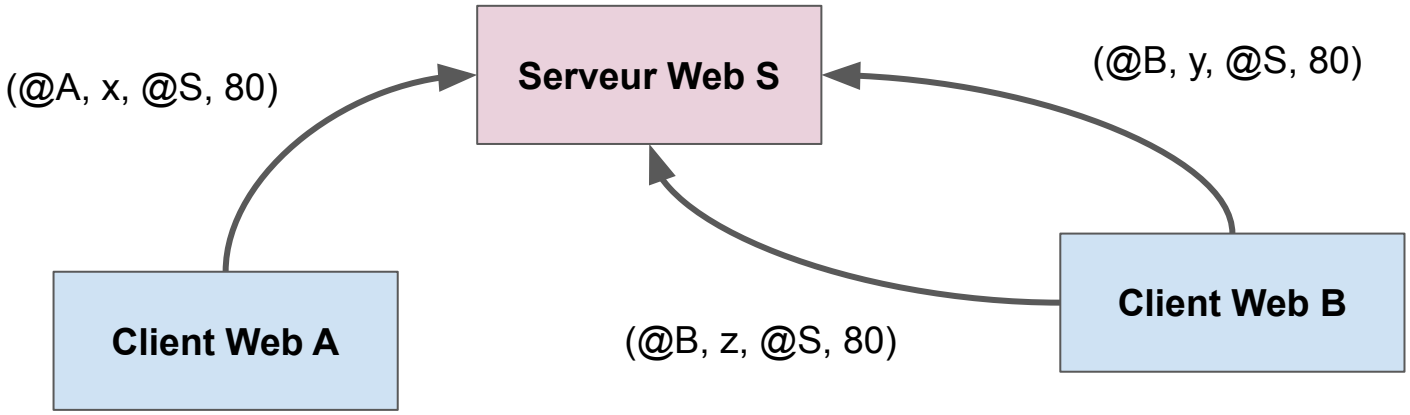
Numéro de Ports et Connexion

Adresse de Transport : une adresse IP (32 bits) + un numéro de port (16 bits)

Une connexion point-à-point : un quadruplet ($@IP_{src}$, $\#Port_{src}$, $@IP_{dest}$, $\#Port_{dest}$)

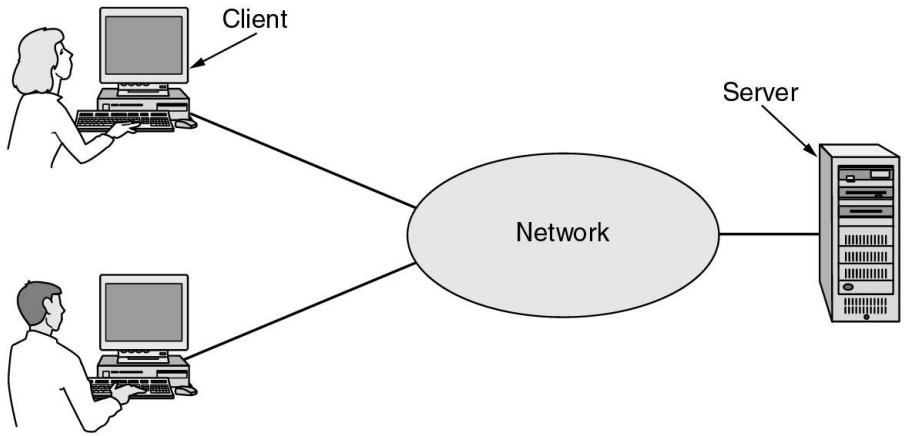
Numéro de Port (< 65535)

- Les ports permettent un multiplexage de connexions au niveau transport.
- Les services standards utilisent des numéros de ports réservés, inférieurs à 1024. Par exemple : web → 80.
- Le numéro de port désigne un processus et un seul dans le système.
- Le client utilise le plus souvent un numéro de port aléatoire.



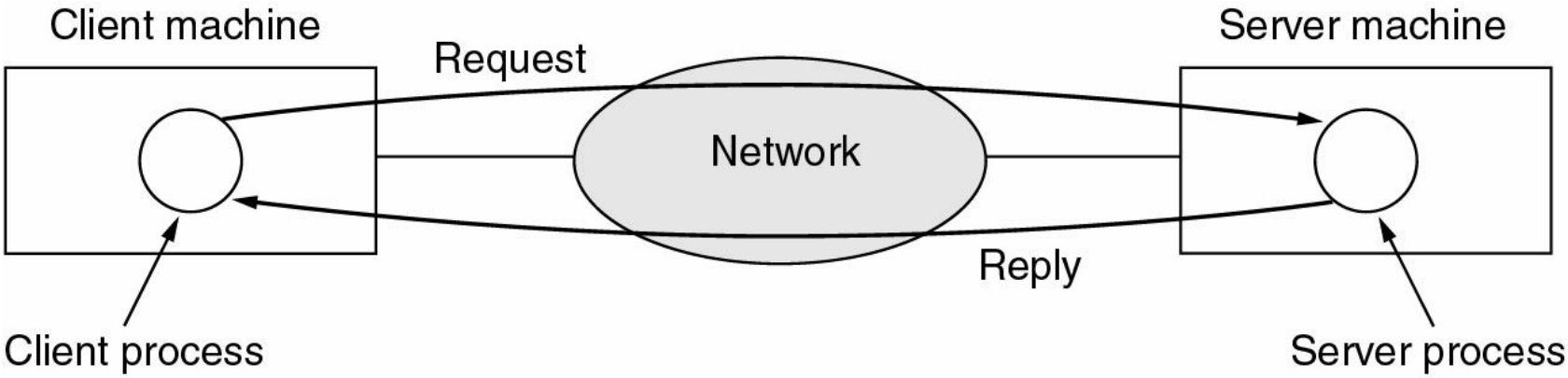
Nota Bene : *y* et *z* doivent être différents pour arriver à distinguer les connexions entre B et S !

Modèle Client-Serveur (TCP/IP)



- Un serveur S est une application, qui offre un service réseau à de multiples clients.
- Le serveur S est à l'écoute (*listen*) sur un port P des demandes de connexion des clients.
- Pour utiliser le service de S, un client C doit initier une demande de connexion auprès de S sur le port P.
- Plusieurs clients peuvent être connectés simultanément à un même serveur.

Modèle Client-Serveur (TCP/IP)



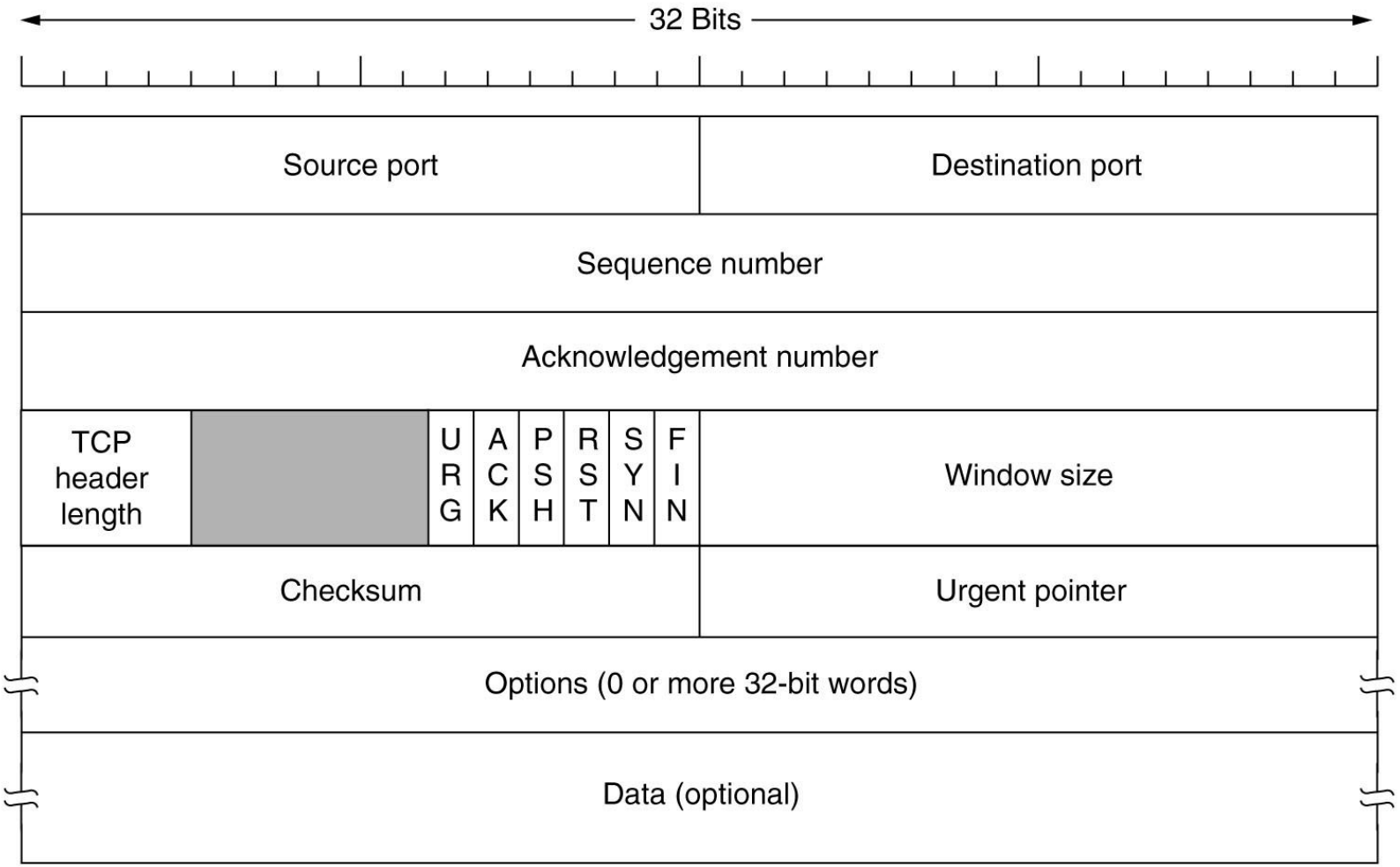
- Une fois la connexion établie (*established*) démarre une session d'échange de messages entre C & S.
- La communication C/S est bidirectionnelle, mais utilise le plus souvent le modèle requête-réponse.
  - Web : Le client effectue une requête HTTTP GET d'une certaine page HTML...
- Plusieurs requêtes & réponses peuvent s'effectuer durant la même possibles au cours d'une connexion… avant la déconnexion.
- La session se termine à la demande de déconnexion du client ou du serveur.

Services Standards

Quelques services standards de TCP

- 21 : FTP (File Transfer Protocol)
- 22 : SSH
- 23 : Telnet
- 25 : SMTP
- 69 : TFTP
- 80 : HTTP
- 110 : POP3 (Post Office Protocol)
- 123 : NTP (Network Time Protocol)
- 143 : IMAP (Internet Message Access Protocol)
- 194 : IRC
- 443 : HTTPS (HTTP Secure)

En-Tête TCP


32 Bits
Source port
Destination port
Sequence number
Acknowledgement number
TCP header length
URG
ACK
PSH
RST
SYN
FIN
Window size
Checksum
Urgent pointer
Options (0 or more 32-bit words)
Data (optional)

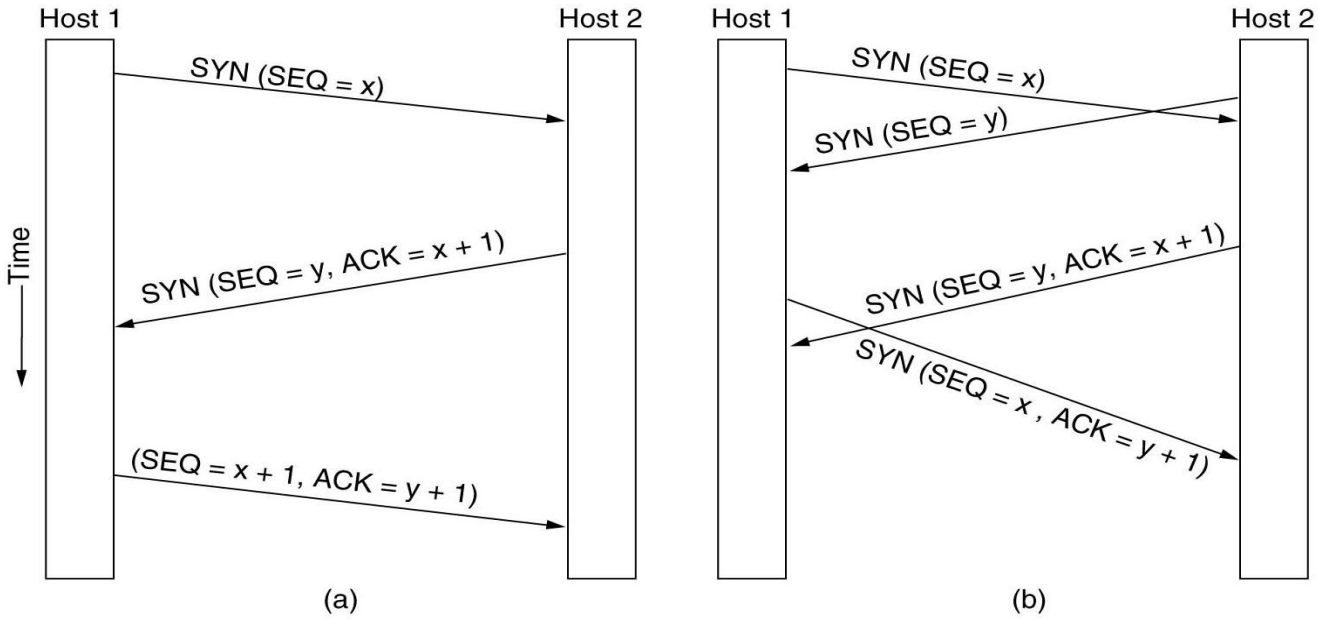
En-Tête TCP

- **Source Port** : numéro de port source [16 bits]
- **Destination Port** : numéro de port destination [16 bits]
- **Sequence Number**: numéro de séquence du premier octet de ce segment [32 bits]
- **Acknowledgement Number** : numéro de séquence du prochain octet attendu [32 bits]
- **Header Length** : longueur de l'en-tête en mots de 32 bits [4 bits]
- **Flags** (binaires) [6 bits]
  - ACK : le paquet est un accusé de réception
  - SYN : demande d'établissement d'une connexion
  - FIN : interruption de la connexion
  - RST : réinitialisation ou rejet de la connexion (*reset*)
  - PSH : données à recevoir tout de suite
  - URG : paquet à traiter de manière urgente
- **Window Size** : nombre d'octets souhaités pour la réception (0 pour stopper temporairement la transmission) [16 bits]
- **Checksum** : somme de contrôle calculée sur l'en-tête et les données [16 bits]
- **Urgent Pointer** [16 bits]
- **Options** : facultatives...

Établissement de Connexion

La *poignée de main* TCP en 3 étapes

- Synchronisation des numéros de séquence



Connexion standard

Connexion simultanée

Lister les Connexions

netstat : lister les services à l'écoute et les connexions en cours sur ma machine...

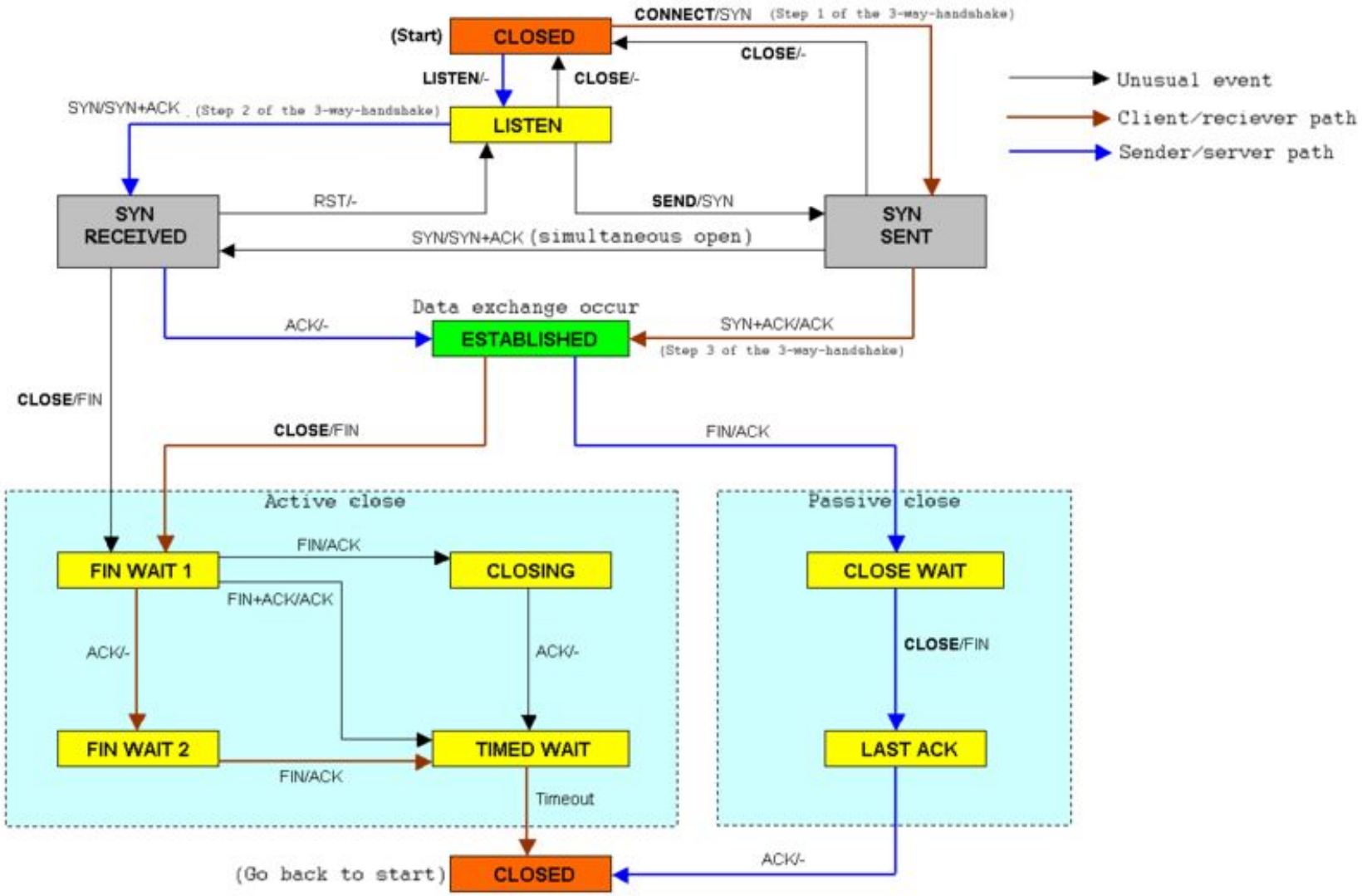
```
$ netstat -tanp
Proto      R-Q     S-Q          Local Address          Foreign Address    State       PID/Program name
tcp        0       0            127.0.0.1:2208         *:*                LISTEN      3266/hpiod
tcp        0       0            127.0.0.1:34818        *:*                LISTEN      3275/python
tcp        0       0            127.0.0.1:3306         *:*                LISTEN      3642/mysqld
tcp        0       0            0.0.0.0:25             *:*                LISTEN      3525/exim4
tcp        0       0            82.225.96.37:35551     147.210.8.143:993  ESTABLISHED 10503/mozilla
tcp        0       0            82.225.96.37:39243     147.210.13.65:22   ESTABLISHED 13758/ssh
tcp        0       0            82.225.96.37:35750     147.210.9.15:22    ESTABLISHED 13763/ssh
tcp6       0       0            *:80                   *:*                LISTEN      3979/apache2
tcp6       0       0            *:22                   *:*                LISTEN      3746/sshd
tcp6       0       0            *:25                   *:*                LISTEN      3525/exim4
```

Les principaux états d'une connexion TCP/IP

- LISTEN : un service à l'écoute
- ESTABLISHED : une connexion établie
- CLOSED : connexion fermée

State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIMED WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

Un Protocole Complexe !



E/M : Lorsque l'evènement E se produit, envoyé le message M ou ne rien faire si M='-'.

Outils

netcat : utilitaire client/serveur qui permet envoyer & recevoir du texte dans un terminal (en interactif)

```
# 1) démarrage du serveur netcat sur la machine hagrid à l'écoute sur le port 8888
hagrid$ netcat -l -p 8888
coucou la terre !          [envoi]

# 2) démarrage du client netcat sur une autre machine (connexion TCP/IP)
casper$ netcat hagrid 8888
coucou la terre !          [reception]

# 3) lister les connexions
hagrid$ netstat -apnt
Proto      Adresse locale       Adresse distante     Etat           PID/Program name
...
tcp        0.0.0.0:8888         0.0.0.0:*            LISTEN         1714761/netcat
tcp        10.0.230.3:8888      10.0.9.6:40116       ESTABLISHED    1714761/netcat
...
```

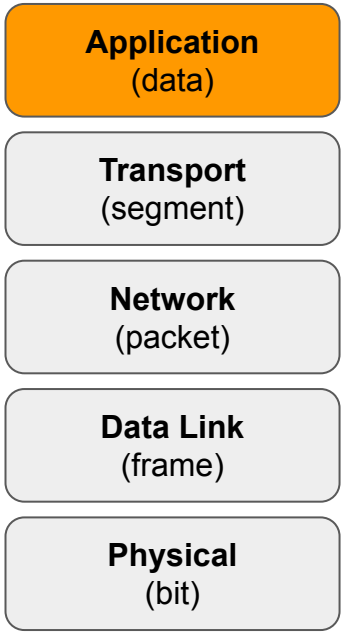
Chapitre 4 (suite)

Couche Application

La Couche Application

La **Couche Application** du modèle Internet se divise en 3 couches dans le **Modèle OSI** :

- **Couche Session** : gestion d'une session qui persiste au-delà d'une connexion, mécanisme d'ouverture et de fermeture de session, identifier un utilisateur, authentification, …
- **Couche Présentation** : encodage des données applicatives (conversion des données au format "machine" dans un format "échangeable"), compression, chiffrement / déchiffrement, …
- **Couche Application** : point d'accès au service réseau ; non spécifiée dans le modèle OSI.



Exemples

- FTP, NFS, SMTP, POP, IMAP, NNTP, Telnet, SSH, X, HTTP, DNS, …

Web

Web (ou la toile) : l'ensemble des hyperliens (ou liens hypertextes) qui relient les pages web entre elles. [1990]

Ne pas confondre Internet et le Web, qui est un des nombreux services Internet !

Premier serveur Web, Tim Berners-Lee au CERN . Source : Wikipedia

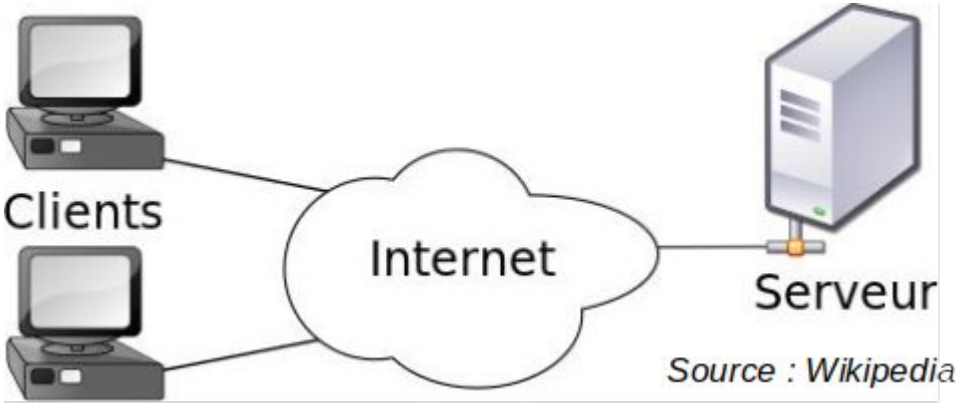
Web

Serveur Web : ordinateur qui contient les ressources du Web (pages, media, …) et les met à disposition sur Internet.

- Ex. : www.google.com, fr.wikipedia.org, …

Navigateur Web : logiciel (client du serveur Web) permettant de consulter les ressources du Web.

- Ex. : Internet Explorer, Firefox, Chromium, …



Web

HTTP (HyperText Transfert Protocol) : protocole de transfert des pages HTML permettant de naviguer sur le Web (HTTPS pour la version sécurisée).

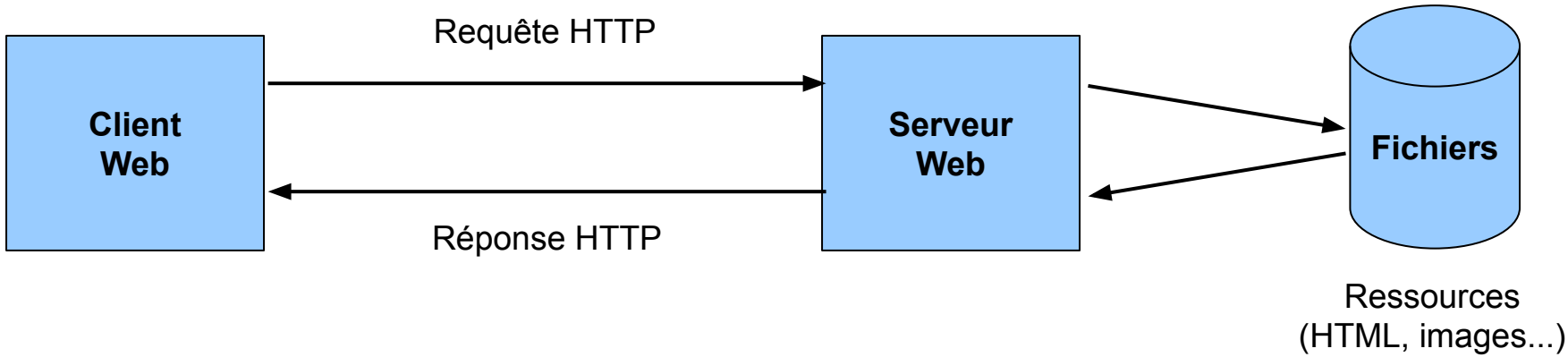
HTML (Hypertext Markup Language) : langage à balise pour représenter les pages Web (mise en forme, liens hypertextes, ressources multimédias, …).

```
<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
 <head>
  <title>
   Exemple de HTML
  </title>
 </head>
 <body>
  Ceci est une phrase avec un <a href="cible.html">hyperlien</a>.
  <p>
   Ceci est un paragraphe où il n'y a pas d'hyperlien.
  </p>
 </body>
</html>
```

Protocole HTTP

HTTP (Hypertext Transfer Protocol)

- Protocole *stateless* basé sur TCP/IP inventé en 1990 (RFC 2616)
- Le serveur est à l'écoute sur le port 80 (ex. Apache, Nginx, …)
- Le client est en général un navigateur (ex. Chrome, Firefox, Edge, …)
- Le navigateur effectue une requête HTTP pour obtenir une ressource à partir d'une URI (Uniform Ressource Identifier)
- Le serveur traite la requête puis retourne une réponse HTTP, typiquement une page HTML…
- HTTPS : version sécurisée de HTTP (port 443)



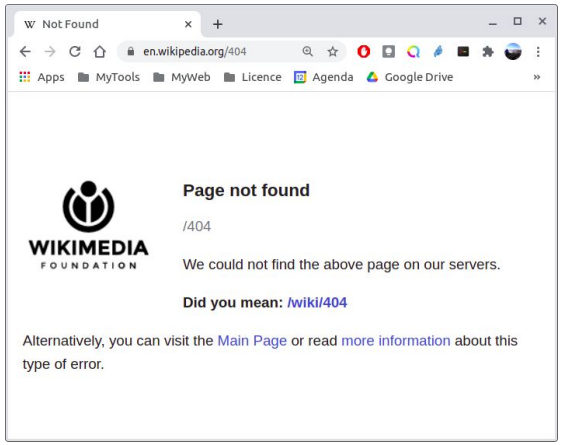
Protocole HTTP

Les principales requêtes

- **GET** : demander une ressource (ex. pages web, scripts, images, …)
- **POST** : envoyer des données au serveur (ex. message forum, formulaire)
- **Divers** : HEAD, TRACE, CONNECT, PUT, DELETE, ...

Codes d'état

- **200** : succès de la requête
- **301** : redirection permanente
- **404** : page non trouvée (erreur)



Historique

- **Version 0.9** : requête GET, réponse HTML
- **Version 1.0** : gestion de cache, type MIME (content-type), ...
- **Version 1.1** : connexion persistante (keep-alive), négociation de contenu (accept-*), ...

Un peu de HTML…

HTML (Hypertext Markup Language) : version 5 en 2014, W3C.

- Langage à balise ouvrante & fermante : <html> … </html>
- En-tête avec des metadonnées : <title>, <meta>, …
- Structuration hiérarchique : <body>, <h1>, <h2>, … <p>, …
- De la mise en forme : , <b>, <pre>, …
- Mais encore : des liens hypertextes <a>, des images <img>, des scripts <script>, des formulaires <form>, …

Exemple

```
<html>
   <head>
      <title>Hello World!</title>
   </head>
   <!-- un commentaire -->
   <body>
      <h1>Hello World!</h1>
      <h2>Subtitle</h2>
      <p>Ceci est un paragraphe.</p>
   </body>
</html>
```

Capture d'une Trace HTTP

Requête GET (en rouge) vers le site www.perdu.com et réponse en bleu…

```
GET / HTTP/1.1
User-Agent: Wget/1.20.1 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: perdu.com
Connection: Close

HTTP/1.1 200 OK
Date: Wed, 19 Feb 2020 18:44:39 GMT
Server: Apache
Upgrade: h2
Connection: Upgrade, close
Last-Modified: Thu, 02 Jun 2016 06:01:08 GMT
ETag: "cc-5344555136fe9"
Accept-Ranges: bytes
Content-Length: 204
Vary: Accept-Encoding
Content-Type: text/html

<html><head><title>Vous Etes Perdu ?</title></head><body><h1>Perdu sur l'Internet ?
</h1><h2>Pas de panique, on va vous aider</h2><strong><pre>    * <----- vous
&ecirc;tes ici</pre></strong></body></html>
```

Capture Wireshark : http://aurelien-esnard.emi.u-bordeaux.fr/trace/http.pcap

Outils

```
$ wget http://www.perdu.com   # ou curl
Resolving www.perdu.com...
Connecting to 208.97.177.124:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 204 [text/html]
Saving to: index.html
```

```
$ cat index.html
<html><head><title>Vous Etes Perdu ?</title></head><body><h1>Perdu sur l'Internet
?</h1><h2>Pas de panique, on va vous aider</h2><strong><pre>    * <----- vous &ecirc;tes
ici</pre></strong></body></html>
```

```
$ tidy index.html
<html>
<head>
<title>Vous Etes Perdu ?</title>
</head>
<body>
<h1>Perdu sur l'Internet ?</h1>
<h2>Pas de panique, on va vous aider</h2>
<pre><strong>    * <----- vous êtes ici</strong></pre>
</body>
</html>
```

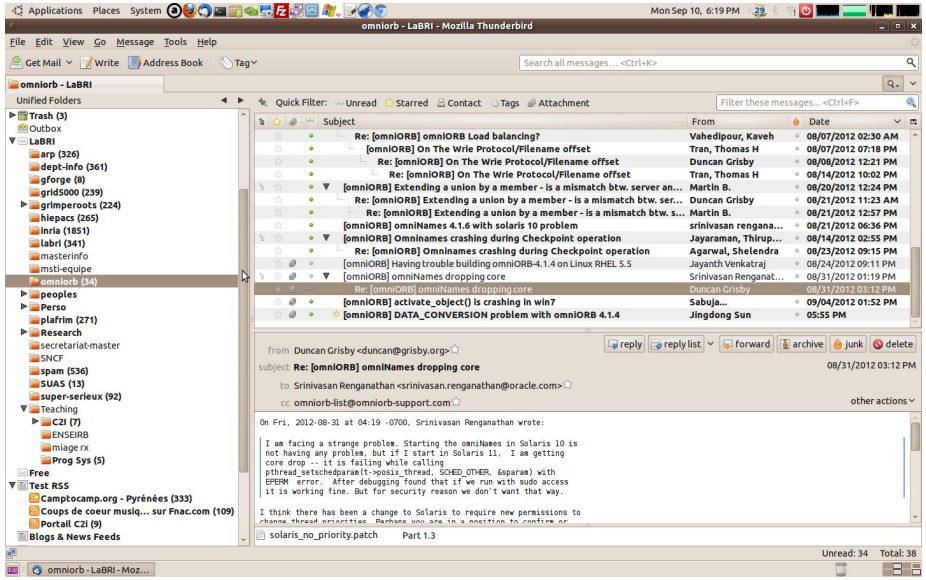
Messagerie Electronique

Messagerie électronique : outil permettant d'échanger des messages (courriel ou mail) de manière asynchrone par l'intermédiaire d'une boîte à lettres électronique identifiée par une adresse électronique.

Adresse életronique : prenom.nom@etu.u-bordeaux.fr

Client de messagerie local ou application webmail

- Ex. : Thunderbird, Outlook, … vs Gmail, Yahoo!, ...

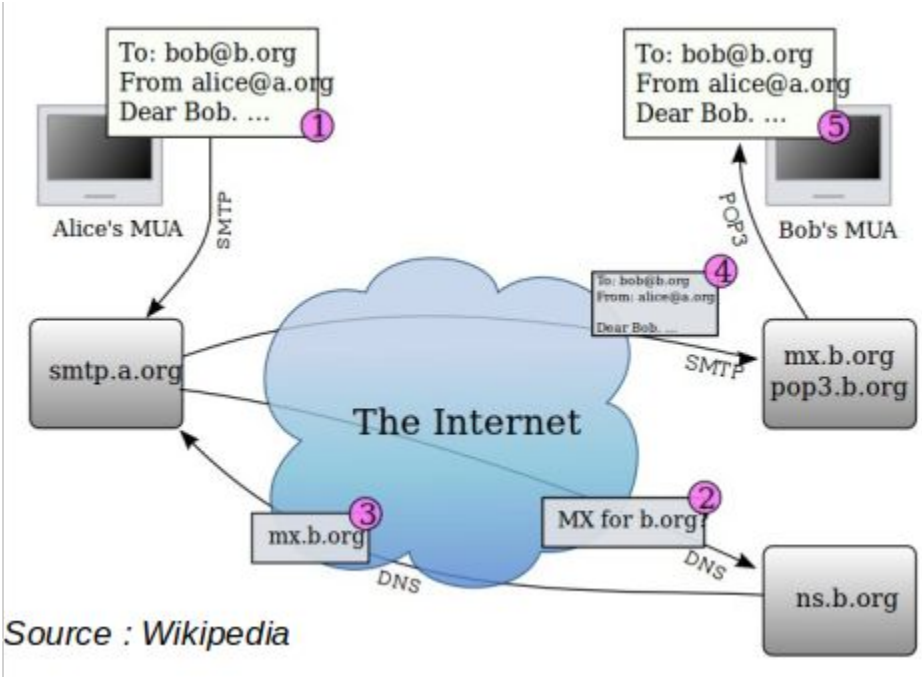


Messagerie Electronique

Principe d'acheminement d'un courriel

Envoi : lorsqu'un expéditeur envoie un courriel, son ordinateur soumet une reqûete au serveur sortant (SMTP), qui l'achemine vers le serveur entrant du destinataire

Réception : lorsqu'un destinataire relève ses courriels, ils sont téléchargés sur son ordinateur depuis le serveur entrant (POP3 ou IMAP)



Source : Wikipedia

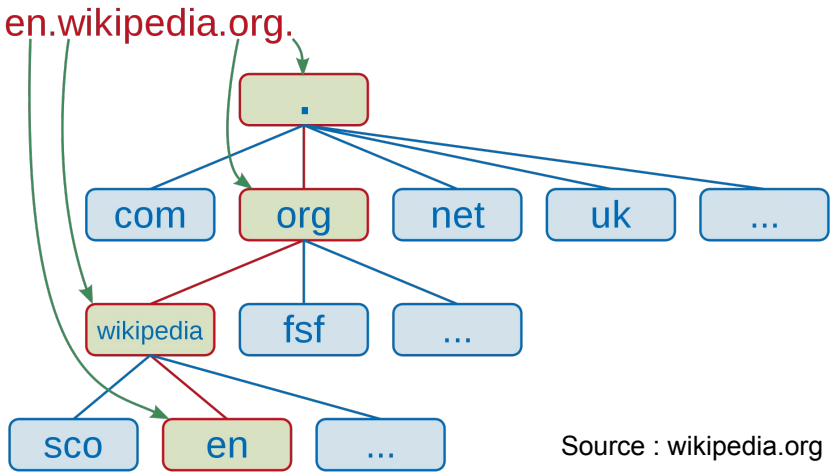
DNS

Problématique : Comment associer un nom lisible à une adresse IP numérique ?

Exemple : en.wikipedia.org ⇒ 91.198.174.192

Protocole DNS (Domain Name System)

- Protocole central dans Internet depuis 1985 (UDP, 53)
- Espace de noms hiérarchique gérés par une hiérarchie de serveurs
- Modèle client/serveur
  - Un client interroge un serveur de noms (serveur DNS) et attend la réponse (UDP, port 53)
  - Utilisé automatiquement par les applications, rarement directement par l'utilisateur



Source : wikipedia.org

DNS

nslookup & host : interroger le serveur **DNS** (Domain Name System) pour trouver l'adresse IP (IPv4 et/ou IPv6) associée à un nom de machine…

```
$ nslookup www.google.com

Server:         10.0.220.1                  <-- Server DNS local
Address:        10.0.220.13#53

Name:           www.google.com
Address:        172.217.18.228

Name:           www.google.com
Address:        2a00:1450:4006:809::2004
```

```
$ host www.google.com

www.google.com has address 172.217.19.228
www.google.com has IPv6 address 2a00:1450:4007:817::2004
```

SSH

SSH (Secure Shell) : session à distance sécurisée

- à la fois un protocole et une commande…
- basé sur TCP (port 22)
- version sécurisée des outils telnet, rsh, …
- authentification par mot de passe ou par clé (RSA, …)

```
# 1) Depuis sa machine locale, se connecter à une machine distante
casper$ ssh hagrid

# 2) Travailler sur la machine distante (en mode texte)
hagrid$ ...

#3) Se déconnecter
hagrid$ exit
casper$
```

Chapitre 5

Programmation Socket

Requête à la Main avec Telnet

```
$ telnet www.perdu.com 80
Trying 208.97.177.124…
Connected to www.perdu.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.perdu.com

HTTP/1.1 200 OK
Date: Tue, 23 Feb 2021 10:02:10 GMT
Server: Apache
Upgrade: h2
Connection: Upgrade
Last-Modified: Thu, 02 Jun 2016 06:01:08 GMT
ETag: "cc-5344555136fe9"
Accept-Ranges: bytes
Content-Length: 204
Cache-Control: max-age=600
Expires: Tue, 23 Feb 2021 10:12:10 GMT
Vary: Accept-Encoding,User-Agent
Content-Type: text/html

<html><head><title>Vous Etes Perdu ?</title></head><body><h1>Perdu sur l'Internet
?</h1><h2>Pas de panique, on va vous aider</h2><strong><pre>    * <----- vous &ecirc;tes
ici</pre></strong></body></html>
Connection closed by foreign host.
```

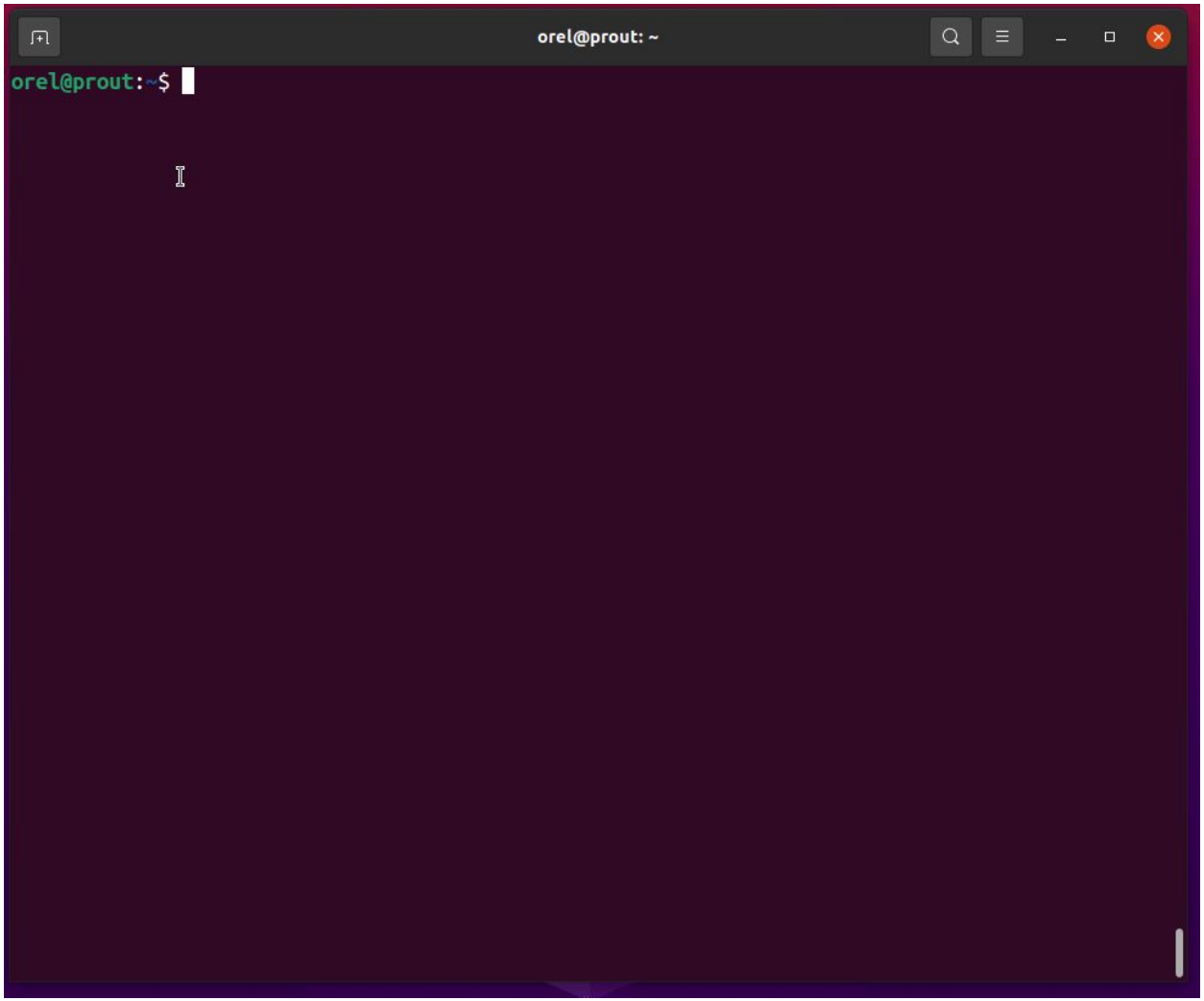
Requête minimale en HTTP/1.1

En-tête de la réponse HTTP

Corps HTML de la réponse HTTP

Requête à la Main avec Telnet

Démo



Socket

Comment programmer des applications réseaux au dessus de la couche Transport ?

- Création d'une *socket*
  - IPv4 (AF_INET) ou IPv6 (AF_INET6)
  - TCP (SOCK_STREAM) ou UDP (SOCK_DGRAM)
- Connexion TCP
  - côté client : connect()
  - côté serveur : accept()
- Configuration d'un serveur : listen() / bind()
- Envoi et réception de données :
  - send() / sendall() / recv() en TCP
  - sendto() / recvfrom() en UDP
- Fermeture de la socket : close()

Documentation pour le langage Python3

- API en Python : https://docs.python.org/3/library/socket.html
- How To : https://docs.python.org/3/howto/sockets.html

Python Tips

Les fonctions de la famille send()/recv() ne manipulent pas des string classiques, mais des byte-array :

```
string = "coucou"      # string classique de type str
byterray = b"coucou"   # byte array (notez le prefixe b)
sock.send(bytearray)
```

Pour convertir une string en byte-array (et inversement), vous pouvez utiliser les fonctions suivantes :

```
bytearray = "coucou".encode()
string = b"coucou".decode()
```

Support de nombreux encodages

```
"é".encode("latin-1")
b'\xe9'
"é".encode("utf-8")
b'\xc3\xa9'
```

Socket Python

Client Daytime (UDP)

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.sendto(b'', ('time-c.nist.gov',13))
data = s.recvfrom(1024)
print(data)
s.close()
```

Client Daytime (TCP)

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('time-c.nist.gov',13))
data = s.recv(1024)
print(data)
s.close()
```

Socket Python

Client HTTP, requête GET (TCP)

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('www.perdu.com', 80))
s.sendall(b'GET / HTTP/1.1\r\nHost: www.perdu.com\r\nConnection:
close\r\n\r\n')
data = s.recv(1024)
s.close()
print (data)
```

Socket Python (Bonus)

Exemple d'un Serveur Echo (TCP) : un seul client à la fois…

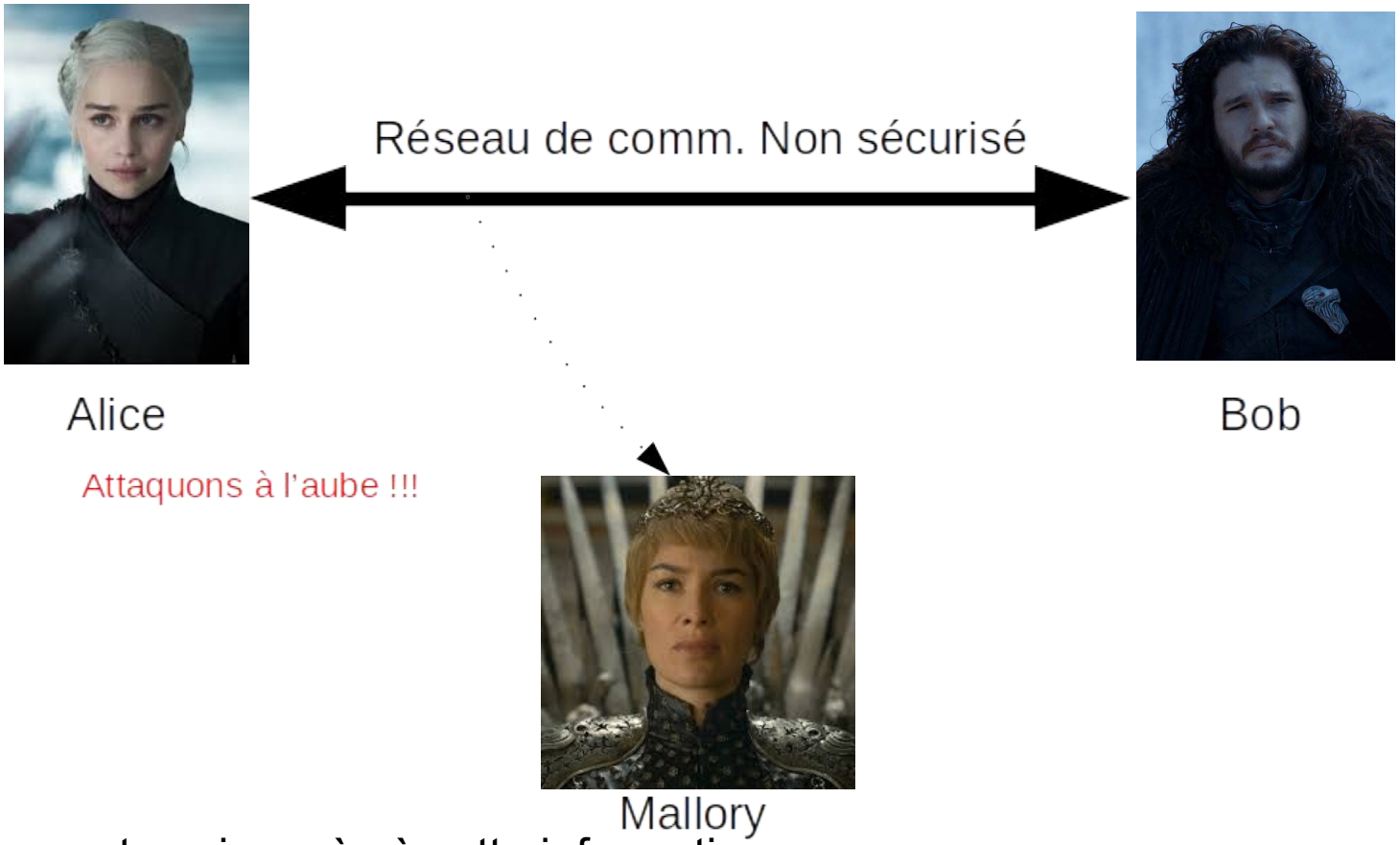
```
import socket
HOST = ''
PORT = 7777
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind((HOST, PORT))
s.listen(1)
while True:
    sclient, addr = s.accept()
    print('Connected by', addr)
    while True:
        data = sclient.recv(1500)
        if data == b'' or data == b'\n' : break
        print(data)
        sclient.sendall(data)
    print('Disconnected by', addr)
    sclient.close()
```

Chapitre 6

Sécurité des Communications

Contexte

Alice veut transmettre une information secrète à Bob (et seulement a Bob) en utilisant un réseau non sécurisé.



Mallory veut avoir accès à cette information.

Cryptographie

Un élément clé dans tous les systèmes de sécurité, essentiel pour assurer 4 objectifs :

- Confidentialité : seules les personnes autorisées ont accès aux données.
- Intégrité des données : seules les personnes autorisées peuvent modifier les données.
- Authentification : prouver l'identité.
- Non répudiation : l'émetteur d'un message ne peut pas dire qu'il ne l'a pas fait.

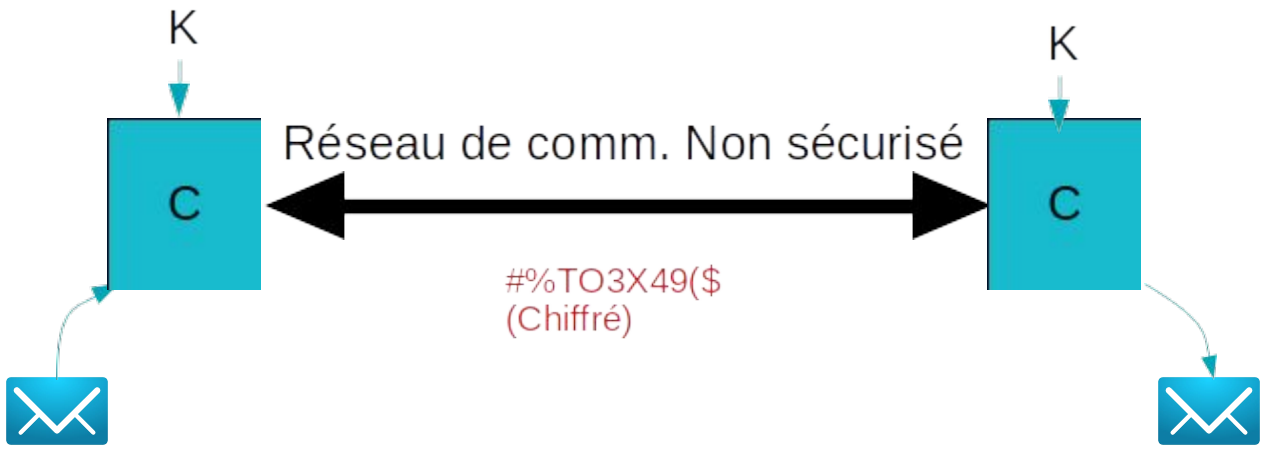
Utilisation du Chiffrement

Alice veut transmettre une information secrète à Bob (et seulement a Bob) en utilisant un réseau non sécurisé.

- Comment gérer les clés ?
- Quel algorithme utiliser ?

Chiffrement Symétrique

- Chiffrement et déchiffrement avec la même clé : $K_E=K_D$
- La clé doit être connue d'Alice et de Bob.
- Algorithmes : AES, DES, …



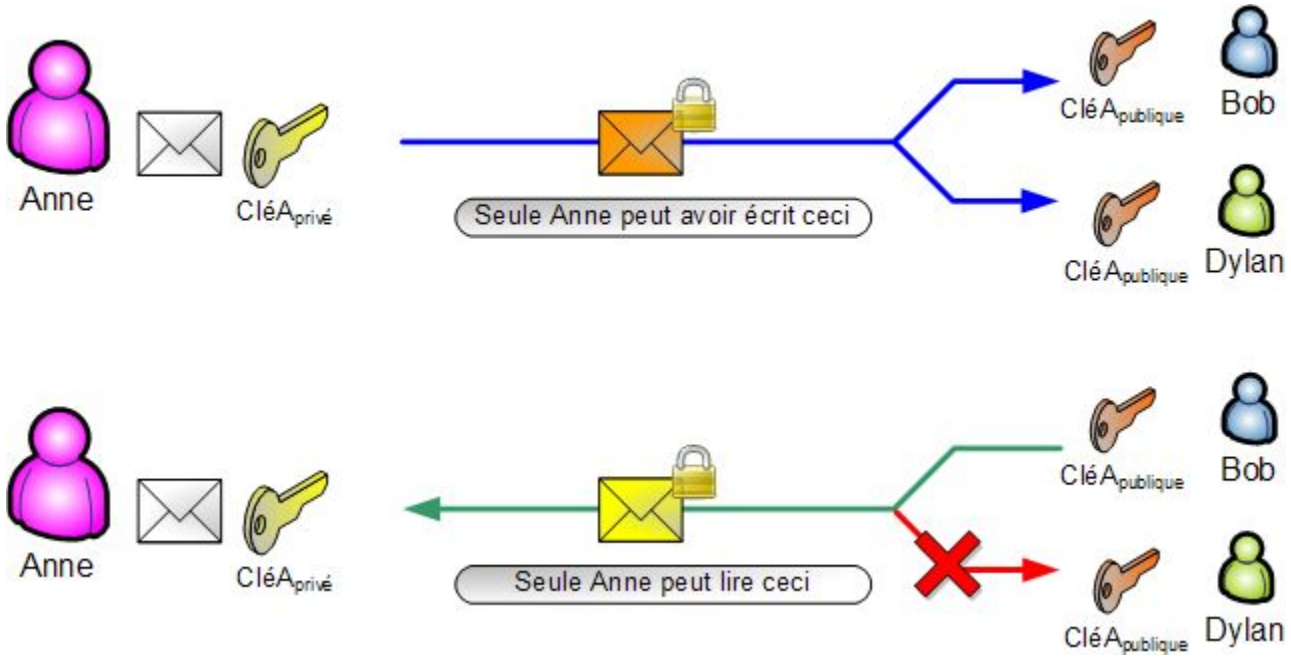
Alice

Attaquons à l'aube !!!

Attaquons à l'aube !!!

Chiffrement Asymétrique

- Clé de chiffrement et de déchiffrement différente : $K_E \neq K_D$
- Alice et Bob possèdent chacun une paire de clé (C,K) telles que :
  - K_{Alice} est privée à Alice et C_{Alice} est publique ;
  - Tout ce qui est chiffré avec C_{Alice} peut être déchiffré avec K_{Alice} ;
  - Tout ce qui est chiffré avec K_{Alice} peut être déchiffré avec C_{Alice} ;
  - De même pour Bob.
- Algorithmes : RSA, ECC, …



Chiffrement Asymétrique

Scénario simple

- Chiffrer avec la clé publique C
- Déchiffrer avec la clé privée K

Chiffrement Asymétrique

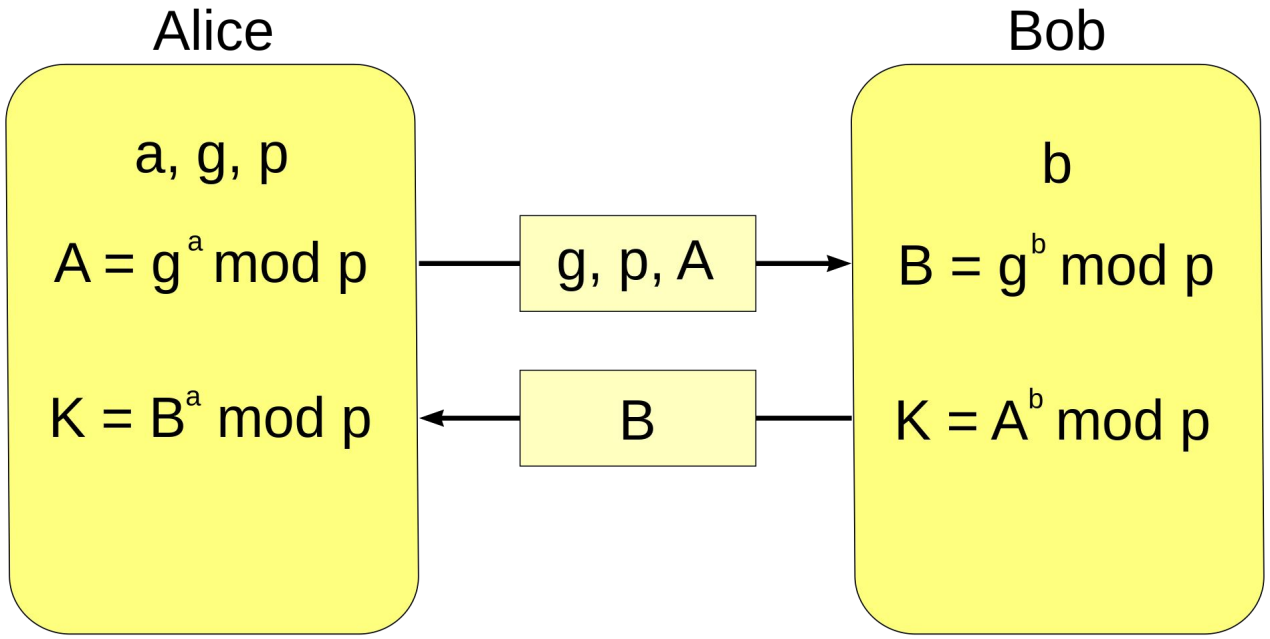
Scénario réaliste

- Générer une clé aléatoire secréte S (symétrique) → clé de session
- Chiffrer S avec C et l'envoyer ; Déchiffrer S avec K
- Utiliser S pour chiffrer le trafic
- Changer S régulièrement au cours de la session…

Confidentialité Persistante

- Que se passe-t-il si un adversaire découvre la clé privée de Bob ou Alice ?
- Comment ne pas compromettre la confidentialité des communications passées ?

Confidentialité Persistante (ou *Perfect Forward Secrecy*) : algorithme de Diffie-Hellman pour le calcul d'une clé de session inviolable…



$K = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = B^a \bmod p$

Algorithmes de Hachage

Permettent la vérification de l'intégrité du message...

- Fonctions à sens unique calculant une empreinte / condensat du message
  - Facilité de calcul du hachage d'un message
  - Impossibilité de retrouver le message à partir du hachage
  - Impossibilité de construire deux messages ayant le même hachage
  - Impossibilité de modifier un message sans mise à jour du hachage
- Algorithmes : SHA256, SHA1, MD5, ...

Exemples :

```
$ echo "bonjour" | sha1sum
1F71E0F4AC9B47CD93BF269E4017ABAAB9D3BD63

$ echo "Attaquons à l'aube!!!" | sha1sum
8073B9D9B2EB74F31F9AE87359AF440883380D7E
```

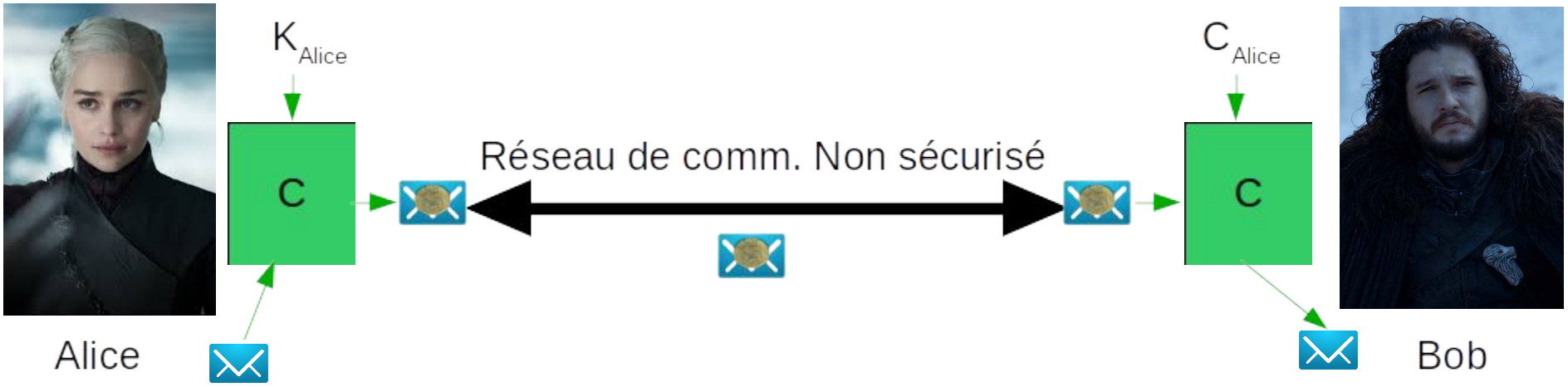
Signature Électronique

Permet de vérifier l'authenticité du message

- Générer le hachage H du message
- Chiffrer H avec K_{Alice} et envoyer le résultat avec le message

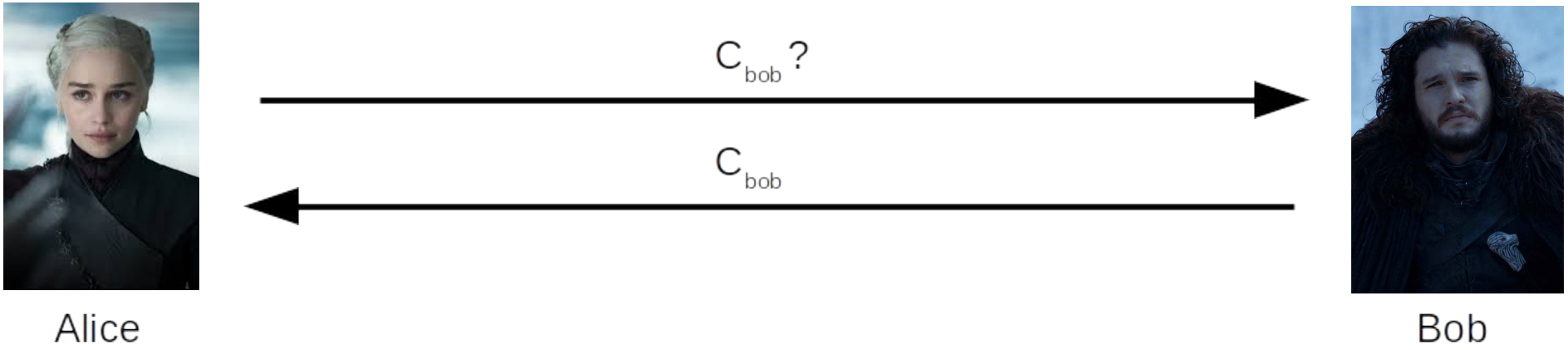
Bob peut vérifier la signature en utilisant C_{Alice}

- Bob est sûr que le message n'est pas corrompu si le résultat du déchiffrement est identique au hachage qu'il calcule
- Bob est sûr qu'Alice est l'émetteur du message



Certificats Électroniques

Que se passe-t-il si Alice n'a pas C_{bob} initialement ?



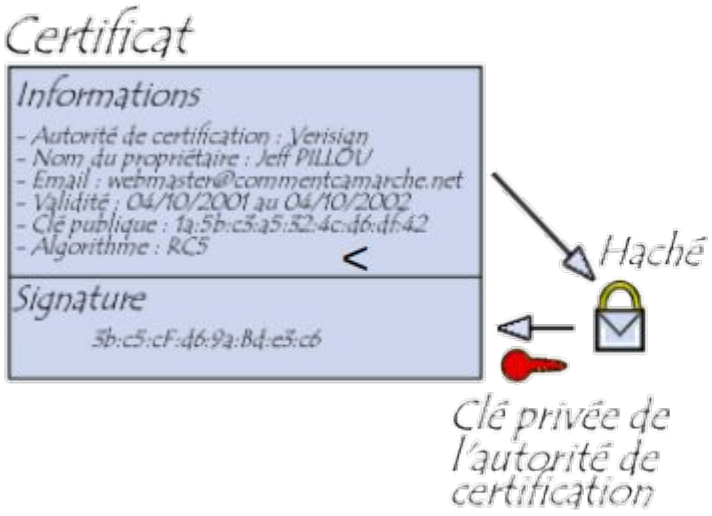
Problème du Man-In-The-Middle !

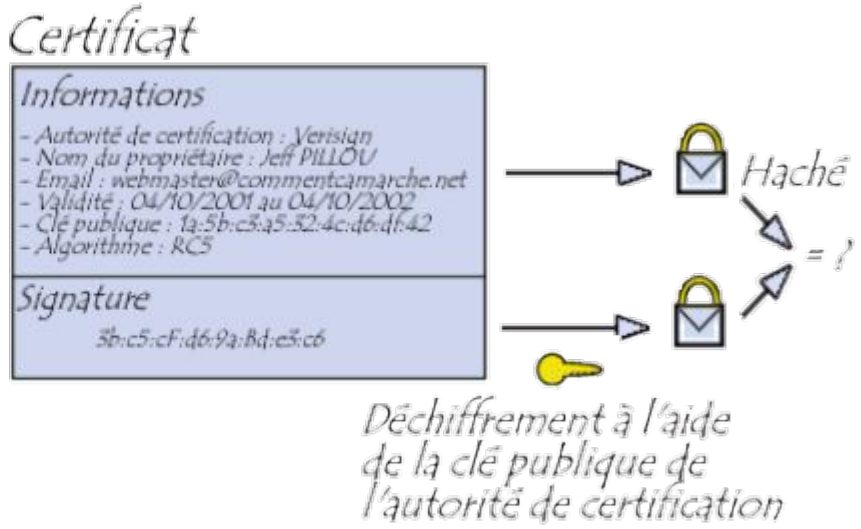
Certificats Électroniques

Un certificat contient :

- Une clé publique + une identité (dans un format clé/valeur)
- Une signature par une autorité de confiance (ou CA) dont la clé publique est connue
- Les clés publiques des CA sont pré-chargées dans votre système d'exploitation…

Vérification d'un certificat

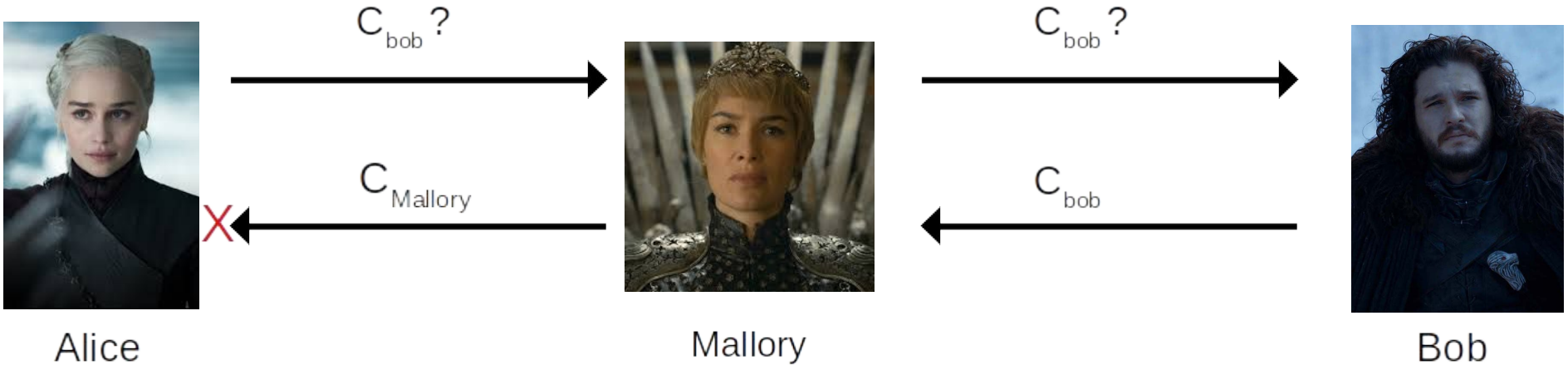




Certificats Électroniques

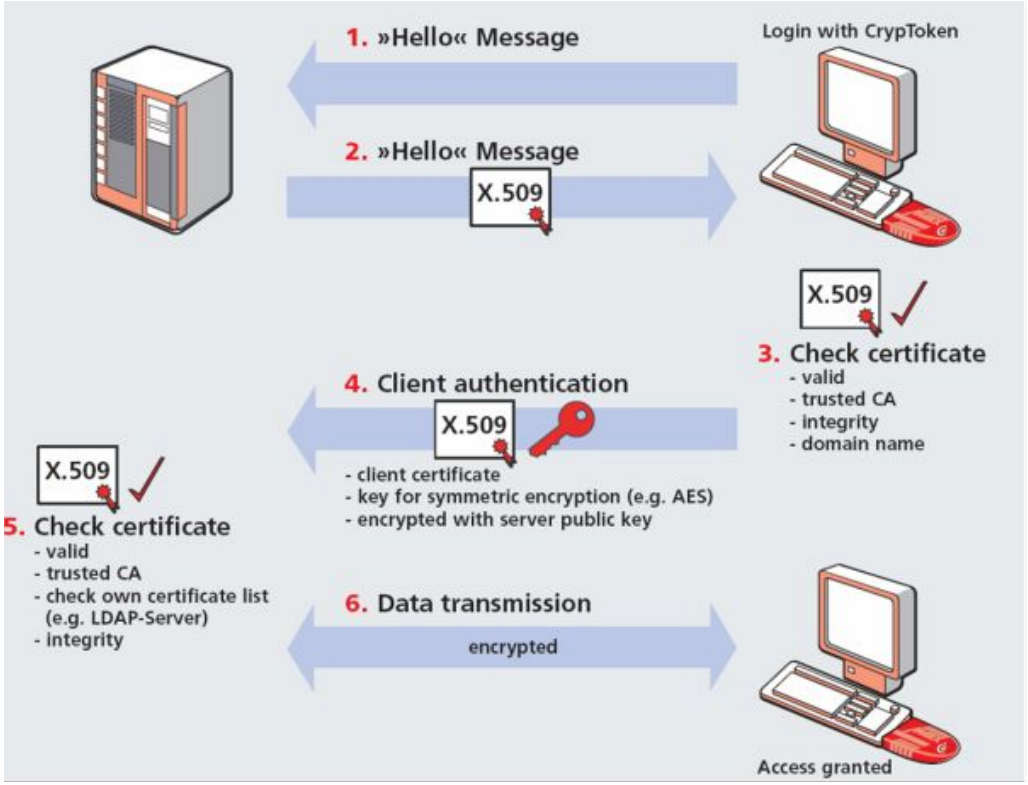
À la réception du certificat de Bob, Alice peut vérifier que le certificat appartient bien à Bob

- Mallory ne peut plus usurper l'identité de Bob...



SSL/TLS

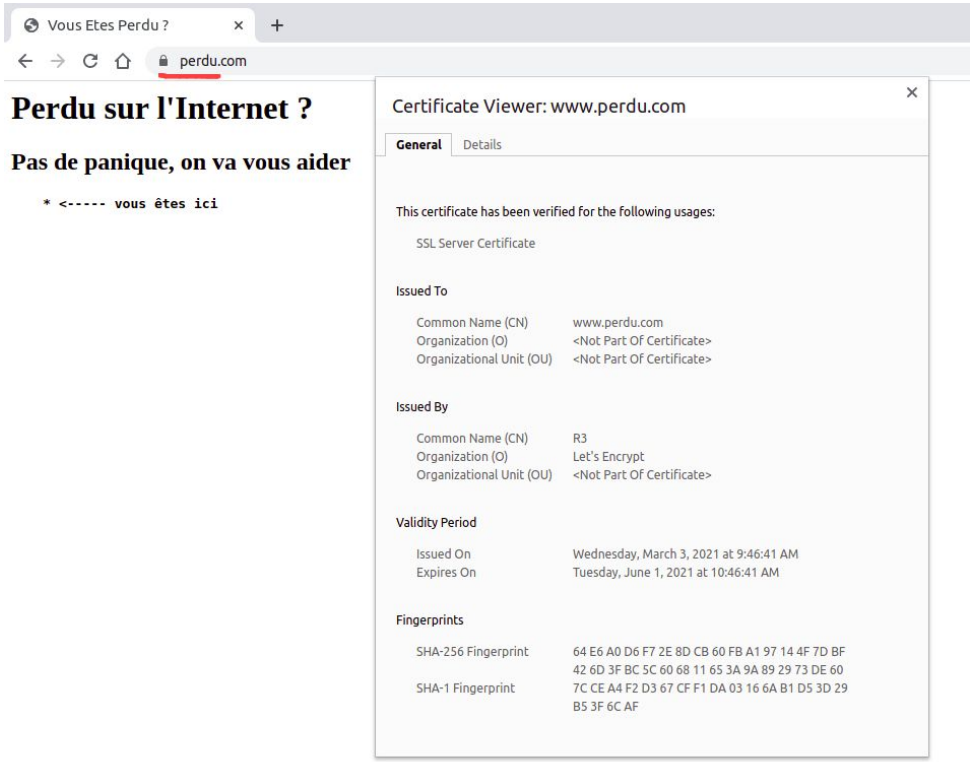
- Protocole de sécurisation des échanges sur Internet
- Basé sur l'utilisation de certificats
- Utilisé pour l'implémentation de versions sécurisées des protocoles standards (HTTPS, SMTPS, IMAPS, ...)



Source : wikipedia

HTTPS

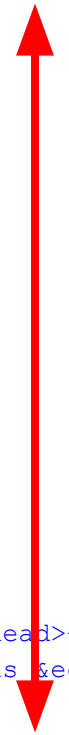
- Utilisation transparente du protocole HTTP au-dessus de TLS/SSL (port 443 au lieu de 80)
- Authentification du serveur web via son certificat (signé du CA)
- Confidentialité et intégrité des données envoyées au serveur
- En général, pas d'authentification du client



Démo HTTPS

https://rx2.gitlabpages.inria.fr/support/data/https.pcap

```
$ guntls-cli --crlf www.perdu.com
Resolving 'www.perdu.com:443'...
Connecting to '208.97.177.124:443'...
- Certificate type: X.509
- Got a certificate list of 2 certificates.
- Certificate[0] info:
 - subject `CN=www.perdu.com', issuer `CN=R3,O=Let's Encrypt,C=US', ...
- Certificate[1] info:
 - subject `CN=R3,O=Let's Encrypt,C=US', issuer `CN=DST Root CA X3,O=Digital Signature Trust Co.', ...
- Status: The certificate is trusted.
- Handshake was completed
- Simple Client Mode:
GET / HTTP/1.1
Host: www.perdu.com

HTTP/1.1 200 OK
Date: Sun, 28 Mar 2021 21:34:49 GMT
Server: Apache
Upgrade: h2
Connection: Upgrade
Last-Modified: Thu, 02 Jun 2016 06:01:08 GMT
ETag: "cc-5344555136fe9"
Accept-Ranges: bytes
Content-Length: 204
Cache-Control: max-age=600
Expires: Sun, 28 Mar 2021 21:44:49 GMT
Vary: Accept-Encoding,User-Agent
Content-Type: text/html

<html><head><title>Vous Etes Perdu ?</title></head><body><h1>Perdu sur l'Internet ?</h1><h2>Pas de panique, on
va vous aider</h2><strong><pre>    * <----- vous &ecirc;tes ici</pre></strong></body></html>
```



échanges sécurisés entre le client et le serveur web

Annexes

Les Outils

Les Outils pour le Réseau

Les outils de base sous Linux & Windows…

Description	Linux	Windows
Afficher toutes les interfaces réseaux	ifconfig -a	ipconfig /all
Tester si la machine d'adresse <ip> est vivante	ping <ip>	ping <ip>
Afficher l'état des connexions réseaux (TCP)	netstat -tapn	netstat
Afficher la table ARP (correspondance des adresses IP / Ethernet)	arp -n	arp -a
Afficher la table de routage	route -n	route print
Afficher le chemin que va suivre un paquet IP pour atteindre la machine <ip>	traceroute <ip>	tracert <ip>
Ouvrir une connexion TCP vers la machine <ip> (port <port>) et lire/écrire des caractères	netcat <ip> <port>	ncat <ip> <port>
Outil d'exploration réseau avec de nombreuses possibilités...	nmap <...>	nmap <...>
Obtenir l'adresse IP d'une machine à partir de son nom <name> en interrogeant le serveur DNS	nsloopup <name>	nslookup <name>
Afficher le traffic réseau associé à toutes (*any*) les interfaces réseaux	tcpdump -i any -n	

Outils

ipcalc : une calculatrice pour les réseaux IP

```
$ ipcalc 192.168.10.1/24
Address:   192.168.10.1           11000000.10101000.00001010. 00000001
Netmask:   255.255.255.0 = 24     11111111.11111111.11111111. 00000000
Wildcard:  0.0.0.255              00000000.00000000.00000000. 11111111
Network:   192.168.10.0/24        11000000.10101000.00001010. 00000000
HostMin:   192.168.10.1           11000000.10101000.00001010. 00000001
HostMax:   192.168.10.254         11000000.10101000.00001010. 11111110
Broadcast: 192.168.10.255         11000000.10101000.00001010. 11111111
Hosts/Net: 254                    Class C, Private Internet
```

Scan d’un Réseau

nmap : outil permettant de découvrir les machines “en vie” dans un réseau, et les services disponibles sur une machine !

Un port peut être ouvert, fermé, ou filtré (cas d’un firewall).

Exemple de scan dans mon réseau domestique

```
# ping sweep
$ nmap -sP -n 192.168.0.0/24

Nmap scan report for 192.168.0.1 => Host is up (0.0035s latency).
Nmap scan report for 192.168.0.100 => Host is up (0.10s latency).
Nmap scan report for 192.168.0.101 => Host is up (0.036s latency).
Nmap scan report for 192.168.0.106 => Host is up (0.00026s latency).
Nmap scan report for 192.168.0.10 => Host is up (0.0064s latency).
Nmap scan report for 192.168.0.11 => Host is up (0.0026s latency).
Nmap scan report for 192.168.0.50 => Host is up (0.013s latency).
Nmap scan report for 192.168.0.254 => Host is up (0.0030s latency).
Nmap done: 512 IP addresses (8 hosts up) scanned in 4.56 seconds
```

Scan d’un Réseau

```
# basic scan
$ nmap 192.168.0.254

Nmap scan report for 192.168.0.254
Host is up (0.0031s latency).
Not shown: 985 filtered ports
PORT     STATE  SERVICE
21/tcp   closed ftp
53/tcp   open   domain
80/tcp   open   http
139/tcp  open   netbios-ssn
443/tcp  open   https
445/tcp  open   microsoft-ds
548/tcp  closed afp
554/tcp  open   rtsp
1723/tcp closed pptp
5000/tcp open   upnp
5001/tcp closed commplex-link
5678/tcp open   rrac
6000/tcp closed X11
8090/tcp open   opsmessaging
9091/tcp open   xmltec-xmlmail
```

```
# syn scan (root privilege required)
$ nmap -sS 192.168.0.100 -p 1-100

Nmap scan report for 192.168.1.11
Host is up (0.0045s latency).
Not shown: 96 closed ports
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
23/tcp open  telnet
80/tcp open  http
```

Annexes

Administration

Configuration d'un LAN

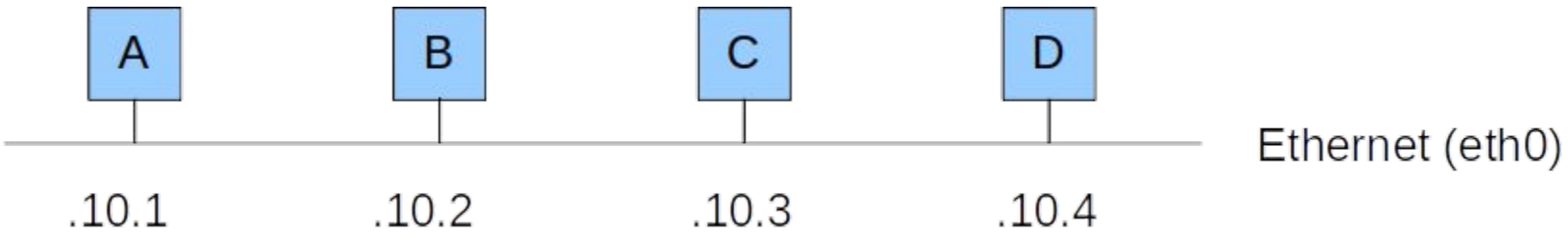
Configuration du réseau local 192.168.10.0/24

- Configuration de la machine A (masque /24 bits)

```
A$ ifconfig eth0 192.168.10.1/24
```

- De même pour toutes les machines B, C et D.
- On peut ensuite effectuer des tests avec *ping*

```
A$ ping 192.168.1O.2
```



⇒ Mise en oeuvre dans le TP2 avec QemuNet.

Routage

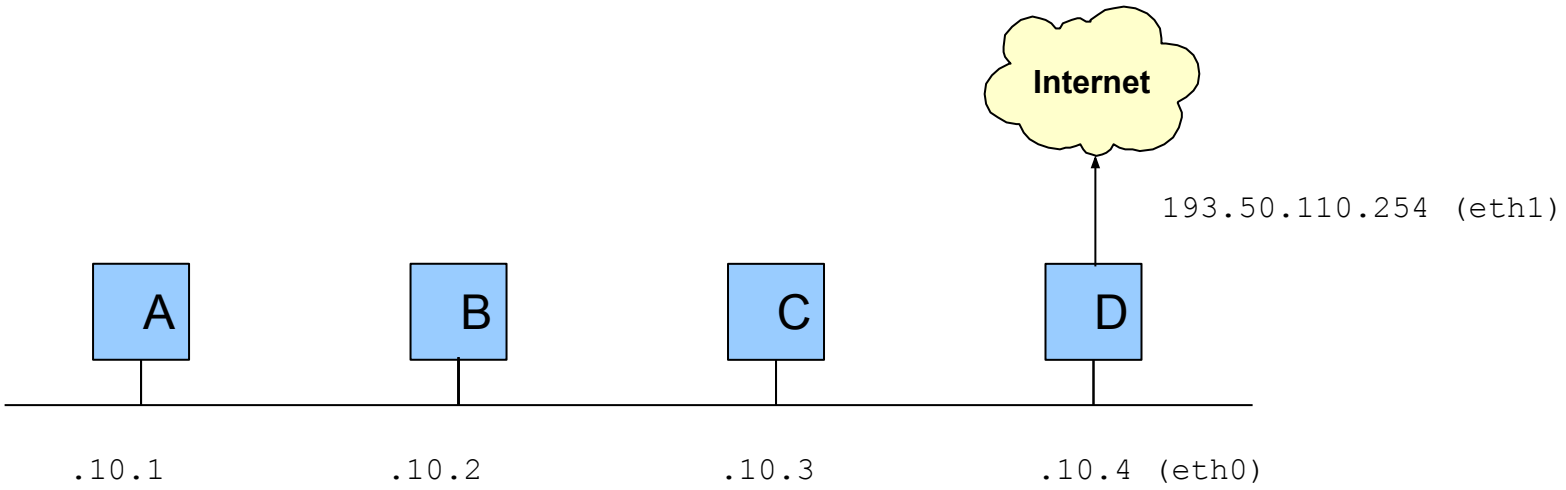
Configuration d'une passerelle D pour le réseau 192.168.10.0/24 permettant d'accéder à Internet

Activer la machine D comme passerelle (IP Forward)

```
$ echo 1 >  /proc/sys/net/ipv4/ip_forward
```

Configuration d'une route par défaut vers l'extérieur pour A, B, C, ...

```
$ route add default gw 192.168.10.4
```



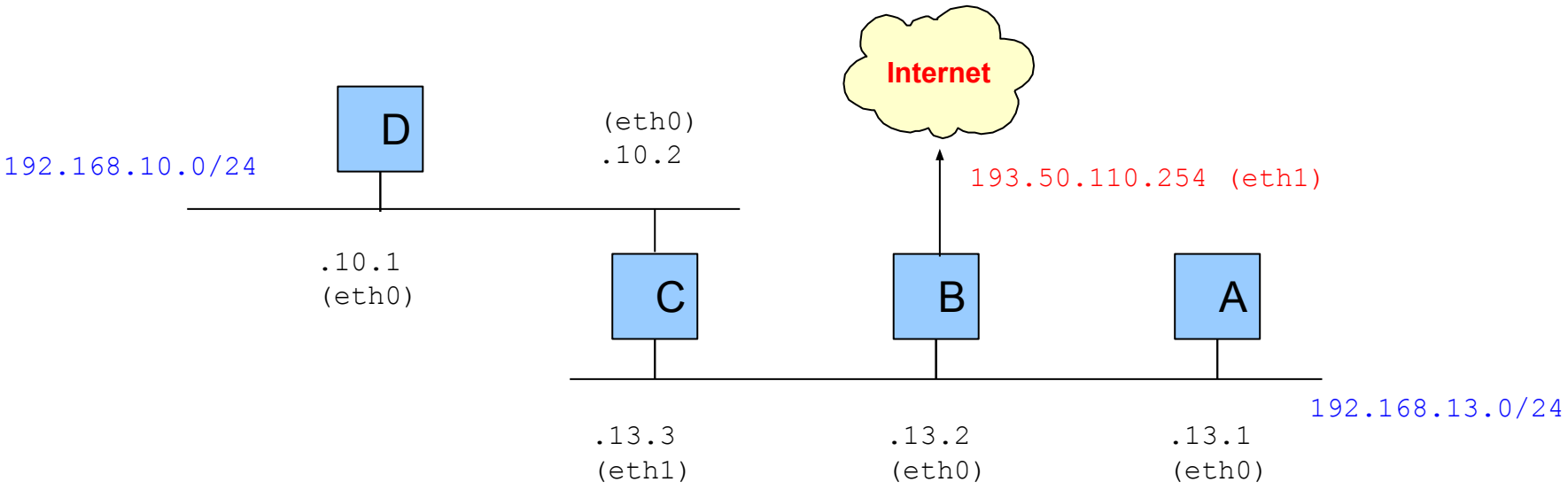
Routage

Pour les machines de 192.168.10.0/24, C joue le rôle de passerelle par défaut

```
D$ route add default gw 192.168.10.2
```

Pour 192.168.13.0/24, C joue le rôle de passerelle vers 192.168.10.0/24 et B joue le rôle de passerelle par défaut

```
A$ route add -net 192.168.10.0 netmask 255.255.255.0 gw 192.168.13.3
A$ route add default gw 192.168.13.2
...
```



Routage : Memento

Activer le routage sur une machine (ip forward)

```
$ echo 1 > /proc/sys/net/ipv4/ip_forward
$ sysctl -w net.ipv4.ip_forward=1            # /etc/sysctl.conf
```

Afficher la table de routage :

```
$ route -n
```

Définir une route par défaut

```
route add default gw <@gateway>
```

Ajouter une route vers un réseau ou une machine particulière

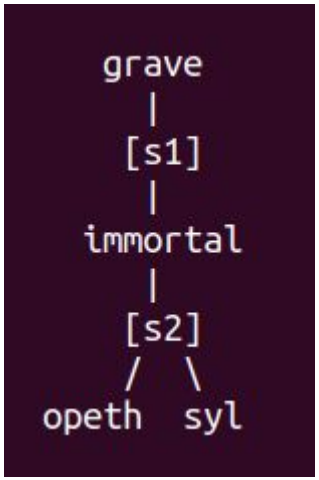
```
$ route add -net <@network> netmask <mask> gw <@gateway>
$ route add -host <@host> gw <@gateway>
```

Pour supprimer une règle, il faut taper la commande *route del* avec exactement les mêmes arguments que pour la commande *route add*.

Routage : Démo

Configurez les IP du réseau suivant.

```
opeth$ ifconfig eth0 192.168.0.2/24
syl$ ifconfig eth0 192.168.0.3/24
immortal$ ifconfig eth1 192.168.0.1/24
immortal$ ifconfig eth0 147.210.0.1/24
grave$ ifconfig eth0 147.210.0.2/24
```



qemunet/demo/gw.topo

Configurez le routage

```
opeth$ route add default gw 192.168.0.1
syl$ route add default gw 192.168.0.1
grave$ route add default gw 147.210.0.1
immortal$ echo 1 > /proc/sys/net/ipv4/ip_forward
```

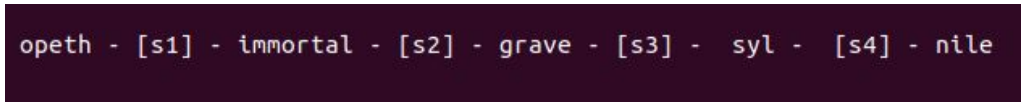
Test de ping entre *opeth* et *grave*

```
opeth$ ping 147.210.0.2
64 bytes from 147.210.0.2: icmp_seq=1 ttl=63 time=0.413 ms
```

```
immortal$ tcpdump -i any
12:06:10.856698 IP 192.168.0.2 > 147.210.0.2: ICMP echo request, id 515, seq 480, length 64
12:06:10.856723 IP 192.168.0.2 > 147.210.0.2: ICMP echo request, id 515, seq 480, length 64
12:06:10.857277 IP 147.210.0.2 > 192.168.0.2: ICMP echo reply, id 515, seq 480, length 64
12:06:10.857285 IP 147.210.0.2 > 192.168.0.2: ICMP echo reply, id 515, seq 480, length 64
```

Routage

Exercice. Considérons le réseau 147.210.0.0/16 avec la configuration suivante. On distingue 4 sous-réseaux interconnectés par les switchs *s1, s2, s3* et *s4*.



- Sur quelle machine faut-il activer le *forward* de paquet IP ?

  Sur *immortal, grave,* et *syl* qui servent de passerelles entre deux sous-réseaux. Ce n'est pas le cas pour *opeth* et *nile* qui n'appartiennent que à un seul sous-réseau.

- Quelle est la route par défaut pour *opeth* ?

  C'est @*immortal* dans le réseau local d'*opeth*. De même pour *nile* avec *syl* comme passerelle.

- Est-ce qu'une route par défaut est suffisante pour *immortal* ? Même question pour *grave*.

  Oui pour *immortal*. En effet, *immortal* peut parler directement à *opeth* et *grave*, mais nécessite d'utiliser *grave* comme passerelle pour parler aux réseaux *s3* et *s4*.

  Non pour *grave*. Dans ce cas précis, il faut ajouter deux routes spécifiques vers les réseaux *s1* et *s4* avec *route add -net* …