Les réseaux : les principes
Comment ça marche ?

Généralités
TCP/IP

Fabrice Harrouet
École Nationale d'Ingénieurs de Brest
harrouet@enib.fr
http://www.enib.fr/~harrouet/

Des réseaux ...

- **Leur rôle**
  - Interconnecter des machines informatiques
  - Mettre en commun des ressources
    - Des informations
    - De la puissance de calcul
- **Leurs propriétés usuelles**
  - Débit
  - Temps de réponse
  - Distance de liaison
  - Mode de transmission
  - Qualité de service ...

Propriétés

- **Le débit**
  - Quantité d'information transmise pendant une durée
  - Généralement exprimé en *bits par seconde*
    - *Kb/s*, *Mb/s*, *Gb/s*
    - Pas des *bytes* mais des *bits* !
  - Éventuellement une autre unité : les *bauds*
    - Nombre de *signes* par seconde
    - Inclue les *bits* de contrôle (parité, start, stop ...)
  - Ne caractérise que le support physique
    - Ce qui est indiqué *"sur la boîte"*
    - Très différent de la quantité de données *"utiles"* !

Propriétés

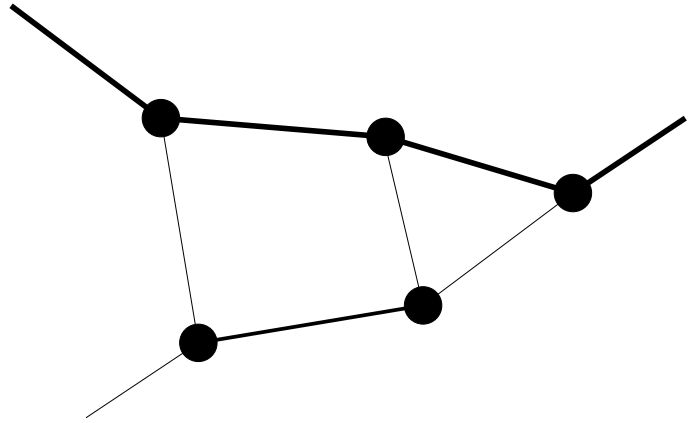
- **Le temps de réponse**
  - Temps écoulé entre l'envoi et la réception d'une information
  - Pas nécessairement lié au débit
- **La distance de liaison**
  - Distance qui sépare les dispositifs
- **Mode de transmission**
  - point-à-point (*peer-to-peer*/*unicast*) → destinataire unique
  - diffusion (*broadcast*) → destinataires *a priori* inconnus
  - diffusion restreinte (*multicast*) → destinataires enregistrés
  - Connecté : un canal de communication
    → fiable mais coûteux
  - Non connecté : une information furtive
    → peu fiable mais économique

Ordres de grandeur

- **Réseau local :** *LAN (Local Area Network)*
  - 1m, 100m ... (bureau, immeuble, campus)
  - *Ethernet* (10Mb/s), *Fast Eth.* (100Mb/s), *Gigabit Eth.* (1Gb/s)
  - *WiFi/802.11b* (11Mb/s), *WiFi/802.11g* (54Mb/s)
  - Temps de réponse pour des machines très proches $\simeq 10\mu$s, 100μs
- **Réseau métropolitain :** *MAN (Metropolitan Area Network)*
  - 1km, 10km ... (campus, technopôle, ville), Mb/s $\rightarrow$ Gb/s
- **Réseau longue distance :** *WAN (Wide Area Network)*
  - 10km, 100km, 1000km ... (pays, continent), 100Mb/s $\rightarrow$ Gb/s
  - Temps de réponse pour un satellite $\simeq 320$ ms
- **Internet : l'ensemble de tous ces réseaux**
  - Nécessité de pouvoir les interconnecter

La commutation

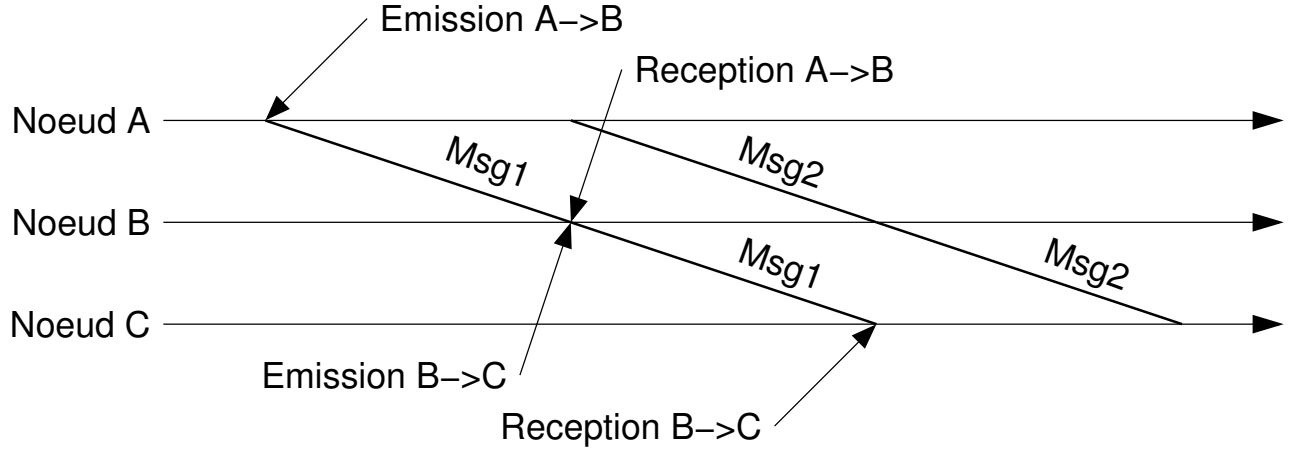
- **Transmettre l'information de nœuds en nœuds**
- **Commutation de circuits**
  - La plus ancienne (cf opérateurs téléphoniques)
  - La liaison est réservée pour une paire d'interlocuteurs
    - Inutilisable pour autre chose
    - Tout le débit est disponible
    - Nécessite énormément de liaisons



La commutation

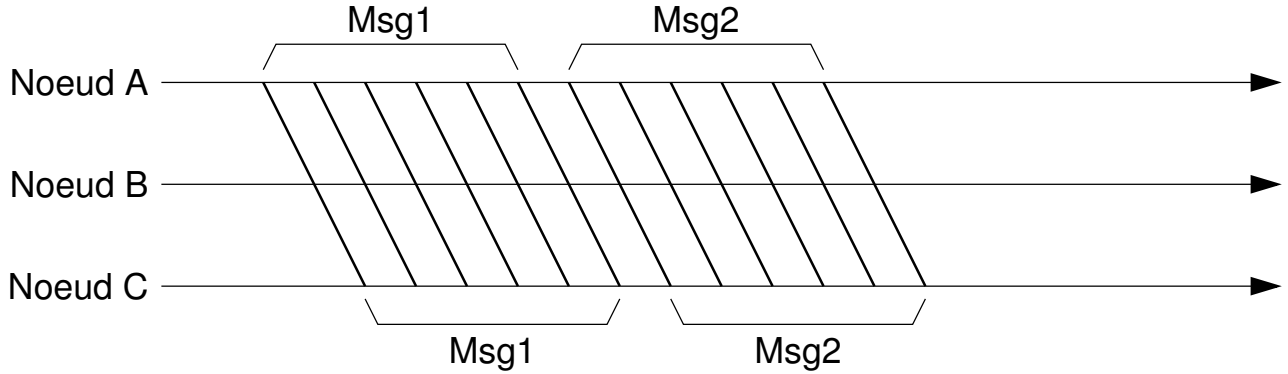
▷ Commutation de messages

- ◇ Multiplexer les liaisons pour en diminuer le nombre
- ◇ La liaison est réservée pendant la transmission d'un message
  - ○ Un fichier, une ligne de commande ...
  - ○ Ré-émission totale si problème de transmission
  - ○ Un nœud attend le message complet avant de le faire suivre
    → long temps de réponse



La commutation

- **Commutation de paquets, de trames, de cellules**
  - Subdiviser le message pour diminuer le temps de réponse
    - Paquets/trames : taille variable
    - Cellules : taille fixe
    - Ré-émission partielle si problème de transmission
    - Attention au rapport volume données utiles / enveloppe (voir l'encapsulation)

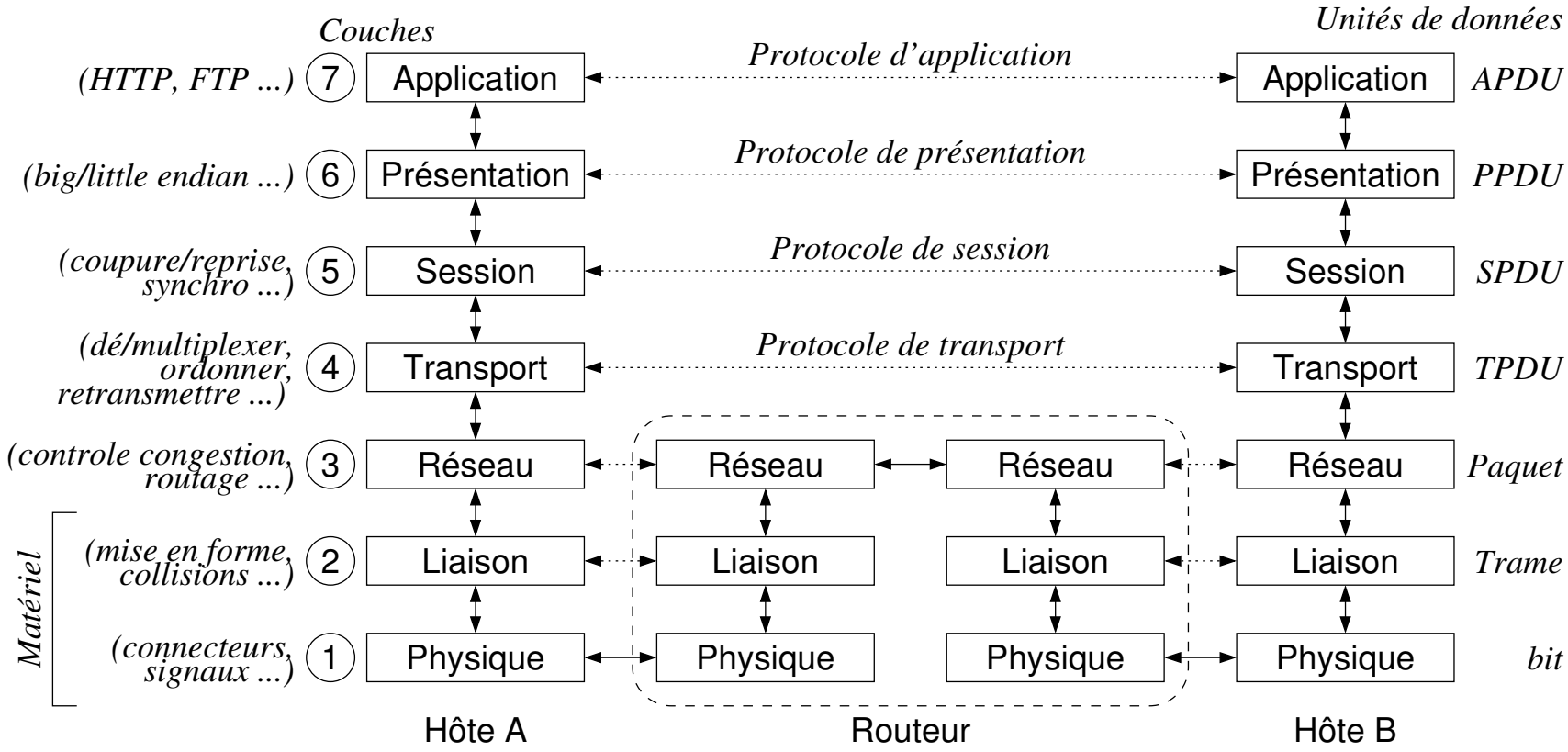


Le modèle en couches

- **Le modèle** *OSI* **(***Open Systems Interconnection***)**
  - Défini par l'*International Standard Organisation*
  - Couches numérotées de 1 à 7
  - À chacune d'elles doit correspondre un protocole
    - Plusieurs protocoles possibles pour chaque couche
      ex : *TCP* ou *UDP* en couche 4
    - Certains sont interdépendants
      ex : *TCP/UDP* en couche 4 nécessitent *IP* en couche 3
  - Protocoles *de bout-en-bout*
    - Une couche dissimule les détails des couches inférieures
  - Couches difficiles à distinguer et matérialiser dans la pratique
    - Modèle utilisé uniquement *"dans les livres"*
    - Dans la pratique on utilise un modèle à 4 couches (*TCP/IP*)

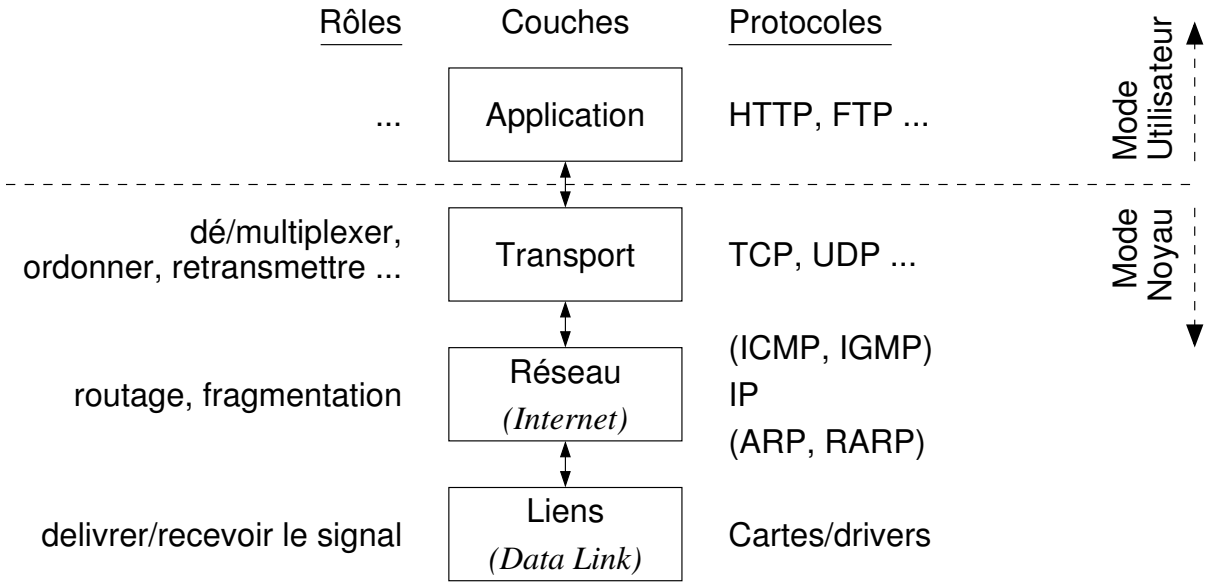
Le modèle en couches

▷ Le modèle *OSI*



Le modèle en couches

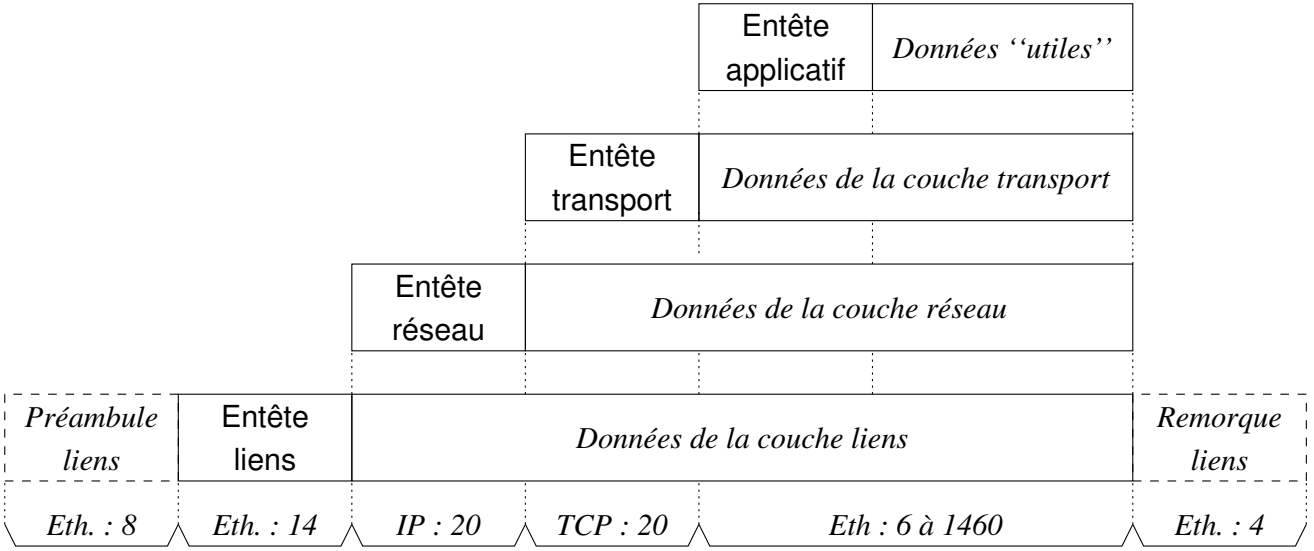
- **Le modèle** *TCP/IP*
  - Modèle à 4 couches *"identifiables"*
  - Inspiré du modèle *DoD* (*Department of Defense*)



Le modèle en couches

▷ Encapsulation

- ◇ Chaque couche *"encapsule"* la couche supérieure
- ◇ Chaque couche possède ses propres structures de données
- ◇ Permet d'acheminer, de router, de démultiplexer ...
  (cf l'enveloppe postale)



Exemple : TCP/IP/Ethernet, de 1.4% (1/72) à 95% (1460/1526) de données applicatives (sans fragmentation)

La couche liens de *TCP/IP*

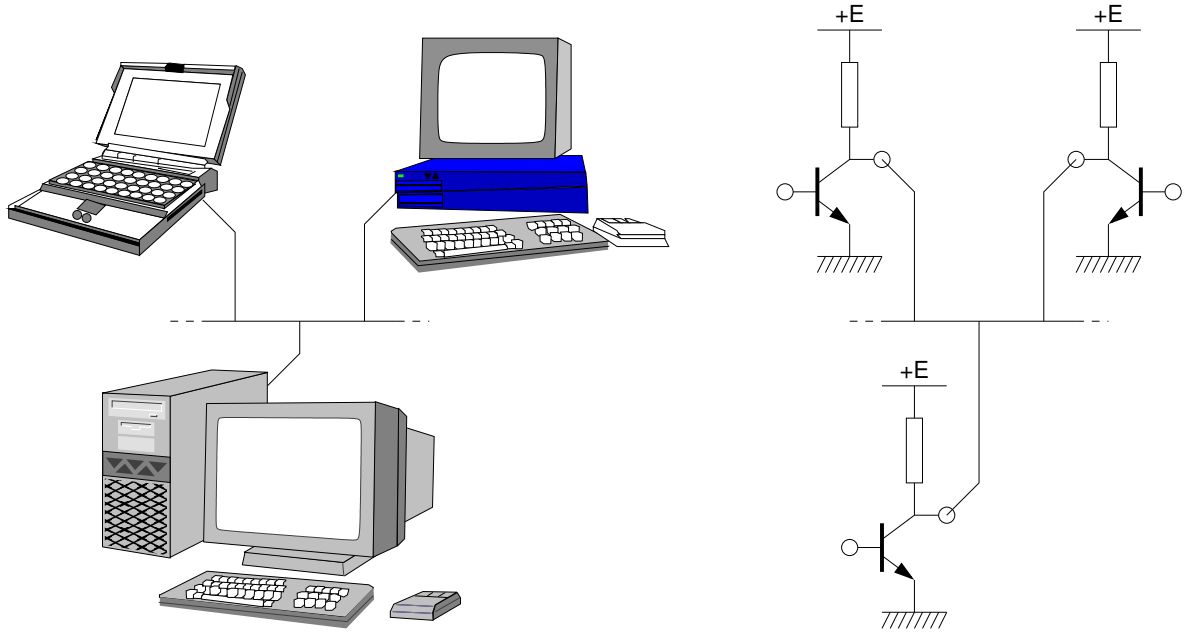
- **Transmettre physiquement un signal entre les dispositifs**
  - Couche physique (1) du modèle *OSI*
  - Dépendant du support physique (médium) et des cartes
- **Structurer les signaux transmis**
  - Couche liaison (2) du modèle *OSI*
  - Notion de *trame* (vocabulaire !)
- **Cas particulier étudié ici :**
  - Théoriquement une infinité de solutions
  - Dans la pratique, essentiellement *Ethernet* et *802.?*

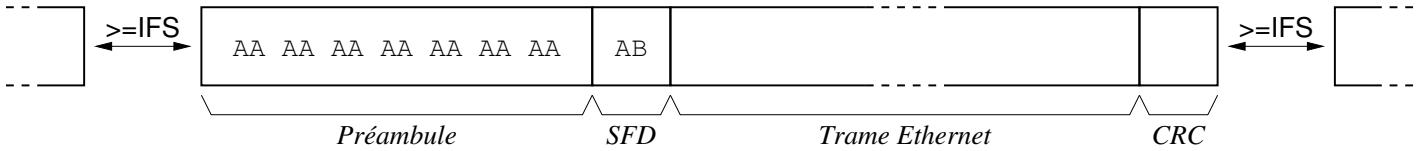
La couche liens de *TCP/IP*

- **La couche physique d'***Ethernet*
  - Codage *Manchester* : limiter la sensibilité aux parasites
    - États haut/bas → fronts montants/descendants
    - Nécessite une fréquence double
  - *CSMA/CD* (*Carrier Sense Multiple Access / Collision Detection*)
    - Écouter si le médium est libre
    - Transmettre le signal au médium **tout en écoutant**
    - S'il y a *collision*, la trame est corrompue
      → attendre aléatoirement avant de ré-essayer
  - Dénomination usuelle : *DébitBase-médium*
    - *10Base-T* : 10Mb/s, paire torsadée, 100m max.
    - *100Base-T* : 100Mb/s, paire torsadée non blindée, 20m max.
    - ...

La couche liens de *TCP/IP*

▷ La couche physique d'*Ethernet*



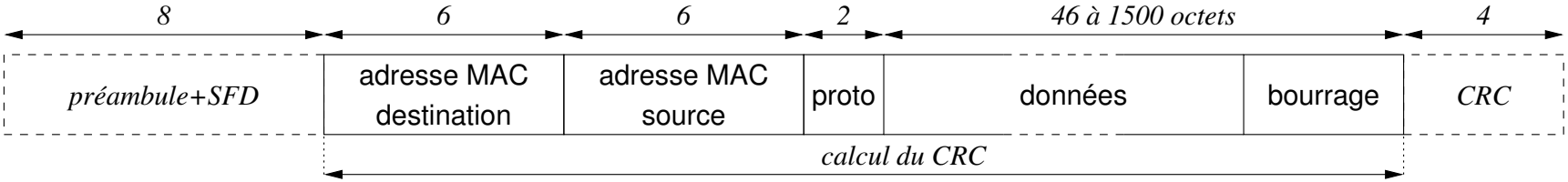


IFS: Inter Frame Spacing, 9.6µs

(Starting Frame Delimitor)

La couche liens de *TCP/IP*

- **La trame *Ethernet*, protocole *MAC* (*Medium Access Control*)**
  - Adresse *MAC* : propriété de la carte réseau
    - Unique, fixée en dur par le fabricant
    - Code fabriquant (3 octets), numéro de carte (3 octets) (*Organizationally Unique Identifier*, *IEEE*, fichier `OUI.txt`)
  - Proto : type des données, comment les interpréter ?
    - $\geq$ 0x0800 : *IP* (0x0800), *ARP* (0x0806), *RARP* (0x8035) ...
  - Bourrage (*padding*) : la trame doit faire 60 octets minimum
    - Trame de durée minimale 51.2μs (10Mb/s)
  - Données limitées à *MTU* (*Maximum Transmission Unit*) : $\leq$ 1500 octets



La couche liens de *TCP/IP*

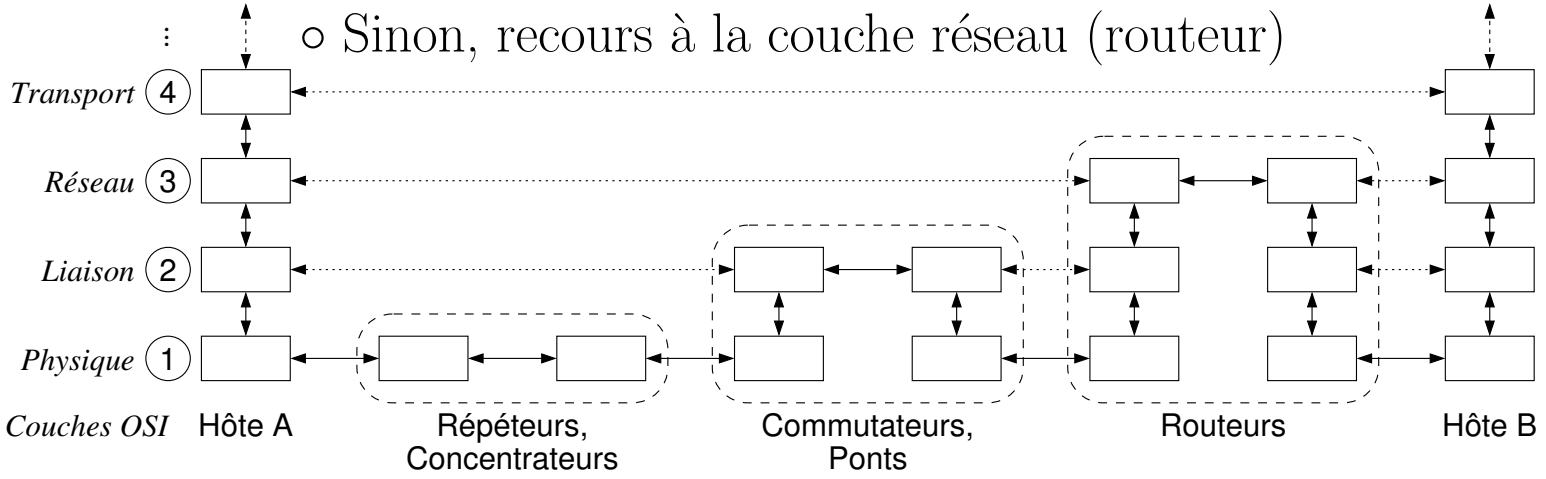
- **Émission d'une trame** *Ethernet*
  - Adresse *MAC* destinataire, proto et données fournis
  - Inscription de l'adresse *MAC* de la carte comme source
    - On peut la forcer avec une *raw socket* ! (ou par configuration)
  - Calcul du *CRC*
- **Réception d'une trame** *Ethernet*
  - Vérification du *CRC*
  - Vérification de l'adresse *MAC* destination
    - Adresse de la carte qui écoute
    - Adresse de diffusion (`FF:FF:FF:FF:FF:FF`)
    - Tout, avec une *raw socket* en mode *promiscuous* !
  - Remontée de la trame en vue de son traitement

La couche liens de *TCP/IP*

- **Échanges de trames** *Ethernet*
  - Les machines d'un même *brin* peuvent communiquer
  - Allonger les brins avec des répéteurs
    - Simple remise en forme du signal
  - Relier plusieurs brins par un concentrateur (*hub*)
    - Ce qui arrive sur un *port* est répété sur tous les autres
    - Le traffic monopolise tous les brins !
  - Relier plusieurs brins par un commutateur (*switch*)
    - Il maintient une table (adresse MAC/port)
    - Adresse source $\rightarrow$ mise à jour de la table
    - Adresse destination $\rightarrow$ choix du port d'emission
    - Cloisonnement : pas de traffic sur tous les brins

La couche liens de *TCP/IP*

- **Échanges de trames entre réseaux hétérogènes**
  - Utilisation d'un *pont* (*bridge*)
    - Machine disposant de plusieurs types d'interfaces
  - Les protocoles doivent être compatibles
    - L'information déterminante est l'adresse *MAC*
    - Exemple : *Ethernet* et *Token Ring*
    - Sinon, recours à la couche réseau (routeur)



La couche liens de TCP/IP

- **Bilan/limitations**
  - Limité à des procédés compatibles (MAC)
  - Connaître les adresses MAC de tous les dispositifs !
  - Procédé *"non fiable"*
    - Détection de collision $\rightarrow$ émission valide
    - Vérification du CRC $\rightarrow$ réception valide
      (mais pas de retransmission en cas de problème !)
    - On ne sait pas qu'une trame n'a pas été reçue !

La couche réseau de *TCP/IP*

- **Utilise le protocole *IP* (*Internet Protocol*)**
  - Échanges de *paquets*/*datagrammes* (vocabulaire !)
  - En relation avec les protocoles *ARP*, *RARP*, *ICMP* et *IGMP*
- **Abstraire l'identification des dispositifs**
  - Attribution d'une adresse *IP*
  - Pouvoir changer de carte sans prévenir tout *Internet* !
- **Permettre le routage**
  - Communiquer à travers des réseaux hétérogènes
  - Dissimuler les détails de connexion
- **Gérer la fragmentation**
  - Respecter les *MTU* de la couche liens
  - Découper les *datagrammes* en *paquets* et les réassembler

La couche réseau de *TCP/IP*

▷ L'adresse *IP*

- ◇ Attribuée à une interface réseau d'une machine
  → plusieurs adresses *IP* si plusieurs interfaces
- ◇ Constituée de quatre octets (*IPv4*) (ex : `192.168.20.236`)
- ◇ Associée à un masque de sous-réseau (ex : `255.255.240.0`)
- ◇ (*IP*`&`masque) → adresse de réseau (ex : `192.168.16.0`)
- ◇ (*IP*`|~`masque) → adresse de diffusion (ex : `192.168.31.255`)
- ◇ Désignation usuelle d'un réseau :
  → adresse/largeur de masque (ex : `192.168.16.0/20`)

Adresse IP	*192* 11000000	• *168* 10101000	• *20* 00010100	• *236* 11101100
Masque de sous–réseau	*255* 11111111	• *255* 11111111	• *240* 11110000	• *0* 00000000
Adresse de réseau	*192* 11000000	• *168* 10101000	• *16* 00010000	• *0* 00000000
Adresse de diffusion	*192* 11000000	• *168* 10101000	• *31* 00011111	• *255* 11111111

La couche réseau de *TCP/IP*

▷ La commande `ifconfig` (`ipconfig` sous *MDO*)

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:04:75:DA:CF:7A
          inet addr:192.168.20.236  Bcast:192.168.31.255  Mask:255.255.240.0
          UP BROADCAST NOTRAILERS RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:29425810 errors:0 dropped:0 overruns:1 frame:0
          TX packets:5159651 errors:0 dropped:0 overruns:0 carrier:22
          collisions:371719 txqueuelen:100
          RX bytes:2830141606 (2699.0 Mb)  TX bytes:2566938591 (2448.0 Mb)
          Interrupt:11 Base address:0xc000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:464512 errors:0 dropped:0 overruns:0 frame:0
          TX packets:464512 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:478999862 (456.8 Mb)  TX bytes:478999862 (456.8 Mb)

vmnet8    Link encap:Ethernet  HWaddr 00:50:56:C0:00:08
          inet addr:172.16.51.1  Bcast:172.16.51.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1896 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
#
```

La couche réseau de *TCP/IP*

- **L'attribution des réseaux, des adresses** *IP*
  - Monde : *IANA* (`www.iana.org`), Europe : *RIPE* (`www.ripe.net`)
  - Numéros de machines au choix à l'intérieur des réseaux
- **Les classes d'adresses** *IP*
  - Classe A : premier bit à `0` (`0.X.X.X` à `127.X.X.X`)
    - Masque de sous-réseau implicite : `255.0.0.0`
    - Au maximum 16777214 adresses *IP* distinctes
  - Classe B : deux premiers bits à `10` (`128.0.X.X` à `191.255.X.X`)
    - Masque de sous-réseau implicite : `255.255.0.0`
    - Au maximum 65534 adresses *IP* distinctes
  - Classe C : trois premiers bits à `110` (`192.0.0.X` à `239.255.255.X`)
    - Masque de sous-réseau implicite : `255.255.255.0`
    - Au maximum 254 adresses *IP* distinctes

La couche réseau de *TCP/IP*

- **Les classes d'adresses** *IP*
  - Classe D : quatre premiers bits à `1110`
    - Adresses de diffusion restreinte (*multicast*)
  - Classe E : cinq premiers bits à `11110`
    - Usage réservé
- *CIDR* **(***Classless InterDomain Routing***)**
  - Préciser explicitement un masque de sous-réseaux
  - Factoriser les entrées des tables de routage (*supernetting*)
    ex : `192.168.20.0` et `192.168.21.0` → `192.168.0.0/16`
  - Découper en sous-réseaux internes (sécurité, administration)
    ex : `172.16.0.0` → `172.16.10.0/24` et `172.16.20.0/24`
    (découper un classe B en 256 sous-réseaux de 254 machines)

La couche réseau de *TCP/IP*

▷ Configuration d'une interface

```
# ifconfig eth1 172.16.200.12
# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 00:E0:18:F3:D3:DF
          inet addr:172.16.200.12  Bcast:172.16.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:227 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:150706 (147.1 Kb)
          Interrupt:5

# ifconfig eth1 172.16.200.12 broadcast 172.16.200.255 netmask 255.255.255.0
# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 00:E0:18:F3:D3:DF
          inet addr:172.16.200.12  Bcast:172.16.200.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:227 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:150706 (147.1 Kb)
          Interrupt:5
#
```

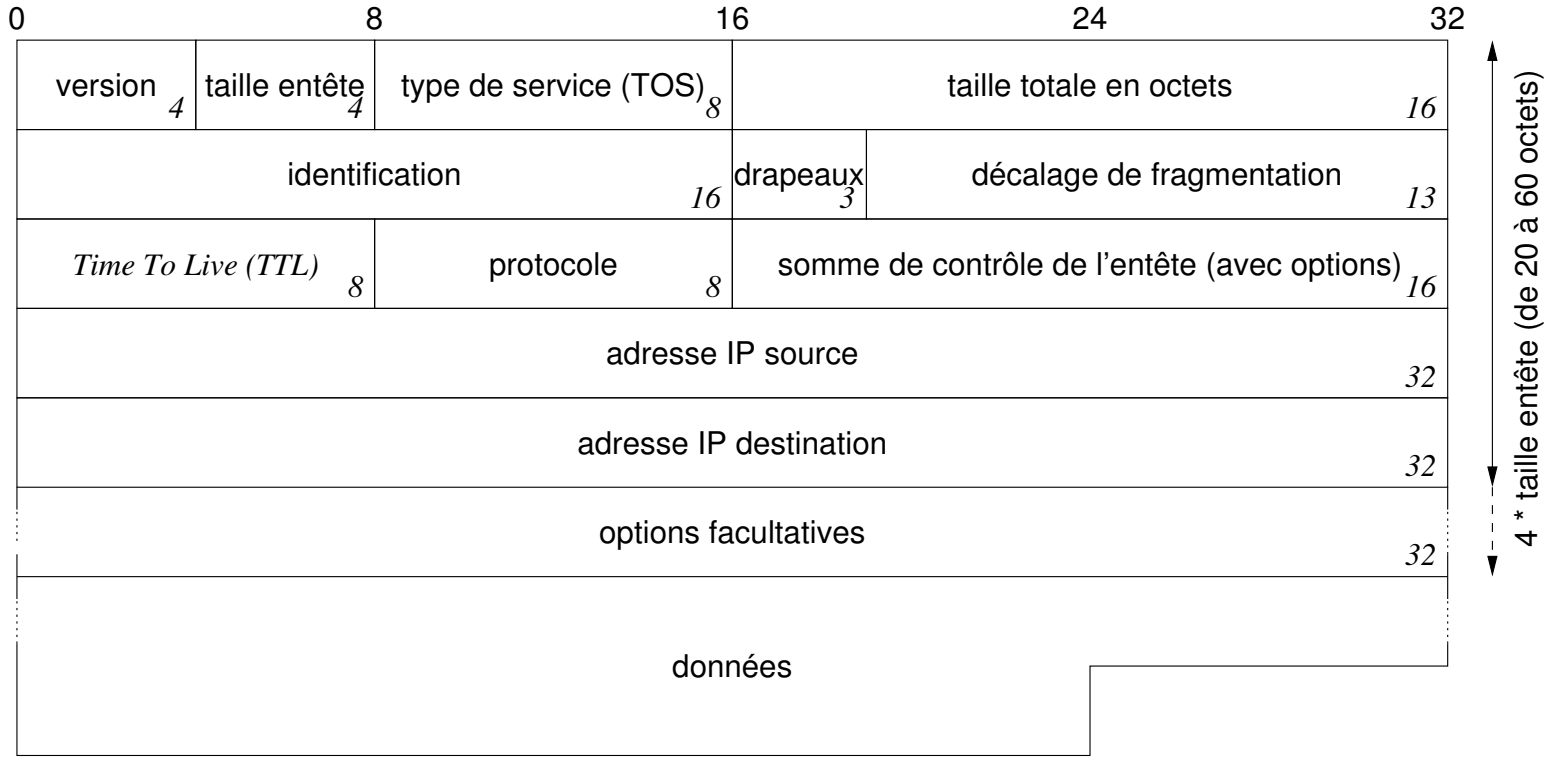
La couche réseau de *TCP/IP*

- **Les adresses *IP* particulières**
  - `127.0.0.0/8` : interface de rebouclage
    - `127.0.0.1` ≡ `localhost`
    - Associée au périphérique virtuel *loopback*
  - `0.0.0.0/255.255.255.255` : src/dest pour autoconfiguration
    - *BOOTP*, *DHCP* ...
  - Adresses de réseaux privés
    - Par *"convention"* non routées sur *Internet* (juste en interne)
    - `10.0.0.0/8` : 1 seul réseau de classe A
    - `172.16.0.0/12` : 16 réseaux de classe B
    - `192.168.0.0/16` : 256 réseaux de classe C

La couche réseau de *TCP/IP*

▷ **Le** *paquet IP*

◇ Il constitue la partie *données* de la trame *Ethernet*



La couche réseau de *TCP/IP*

- **Le** *paquet IP*
  - *version* : 4 pour *IPv4*, 6 pour *IPv6* (non traité ici)
  - *taille entête* : nombre de mots de 32 bits de l'entête (de 5 à 15)
  - *TOS* : 3 bits ignorés, 4 bits significatifs, 1 bit nul
    - Minimiser le délai, le débit, la fiabilité, le coût monétaire ?
    - Choix exclusif, utilisé (ou non !) par les routeurs
  - *taille totale* : taille en octets du paquet (entête + données)
  - *identification* : numéro de datagramme (**++** à chaque envoi)
  - *drapeaux*, *décalage* : voir la fragmentation
  - *TTL* : voir le routage
  - *protocole* : type des données, comment les interpréter ?
    - *ICMP* (1), *IGMP* (2), *TCP* (6), *UDP* (17) ...
  - *somme de contrôle*, *source/destination* : ...
  - *options* : mots de 32 bits (infos de routage ...)

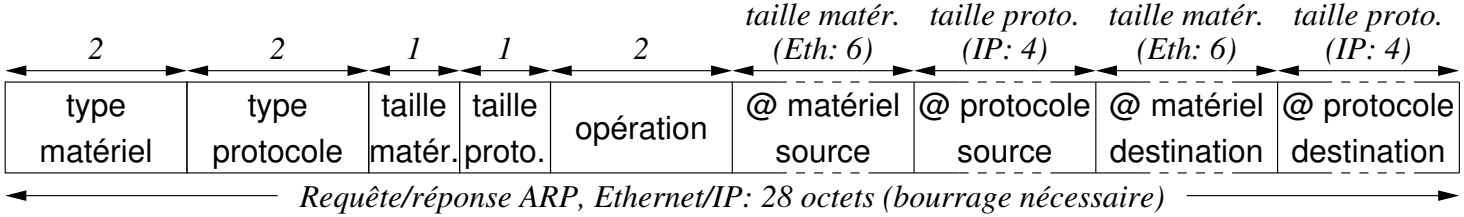
La couche réseau de *TCP/IP*

- **Relation adresse *IP* → adresse *MAC***
  - Protocole *ARP* (*Address Resolution Protocol*)
    - Au dessous d'*IP* mais étroitement lié, proto.*Ethernet*=`0x0806`
  - Les dispositifs *réseau* ont tous une table de résolution *ARP*
    - Paires (adresse *IP*, adresse *MAC*)
    - Mise à jour dynamique + *"oubli"* périodique
    - Mise à jour statique à la demande
- **Envoi d'un paquet vers une adresse *IP* (∉ table *ARP*)**
  - Émission d'une requête *ARP* (*who-has*) pour l'adresse *IP* (trame *Ethernet*, destination=*broadcast MAC*)
  - Toutes les machines du réseau local reçoivent la requête
  - Celle qui a l'adresse *IP* demandée répond à l'émetteur
  - L'émetteur connait maintenant l'adresse *MAC* recherchée

La couche réseau de *TCP/IP*

▷ **Contenu d'une requête/réponse** *ARP*

- ◇ *type matériel* : *Ethernet* (1) ...
- ◇ *type protocole* : *IP* (`0x0800`) ...
- ◇ *taille adresse matérielle* : 6 pour *Ethernet*
- ◇ *taille adresse protocole* : 4 pour *IP*
- ◇ *opération* : requête/réponse *ARP* (1/2), *RARP* (3/4)
- ◇ *adresses source/destination matériel/protocole* :
  - ○ Requête : *MAC1*, *IP1*, `00:00:00:00:00:00`, *IP2*?
  - ○ Réponse : *MAC2*, *IP2*, *MAC1*, *IP1*
  - ○ nb : adresses *MAC* redondantes avec la trame *Ethernet*



La couche réseau de *TCP/IP*

- **Relation adresse** *MAC* $\rightarrow$ **adresse** *IP*
  - Protocole *RARP* (*Reverse Address Resolution Protocol*)
    - Au dessous d'*IP* mais étroitement lié, proto.*Ethernet*=`0x8035`
  - Le contenu de la trame est identique à *ARP*
  - Utilisé principalement pour la configuration automatique
    - *DHCP* : obtenir une adresse *IP* et toute la configuration réseau depuis un serveur dédié
    - *BOOTP* : démarrer sans disque, par le réseau
    - ...

La couche réseau de *TCP/IP*

- **Le routage**
  - Faire parvenir les paquets *IP*
    - Les adresses *MAC* ne suffisent pas → repose sur les adresses *IP*
    - Procédé saut-à-saut (*hop-by-hop*) → si je ne sais pas, je délègue
    - Procédé *"non fiable"*, les paquets peuvent ne pas arriver !
  - Les dispositifs *réseau* ont tous une table de routage
    - Où envoyer un paquet destiné à telle adresse *IP* ?
    - Destination : adresse d'une machine ou d'un réseau
    - Passerelle : une machine qui saura acheminer le paquet
    - Interface : la carte réseau à utiliser
    - Du plus spécifique au plus général

La couche réseau de *TCP/IP*

▷ Exemple de table de routage

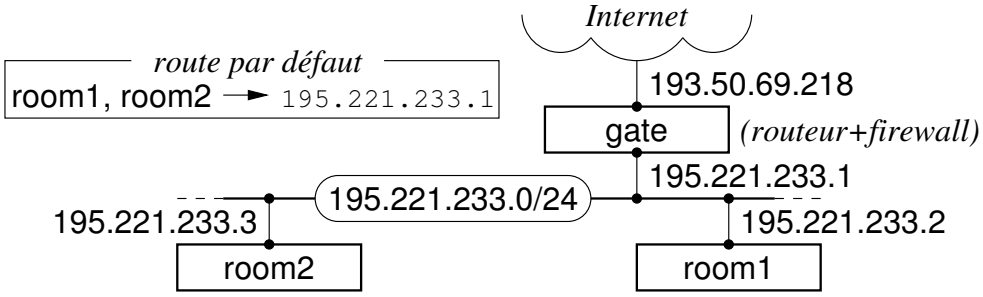
```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:04:75:DA:CF:7A
          inet addr:192.168.20.236  Bcast:192.168.31.255  Mask:255.255.240.0
          ...
eth1      Link encap:Ethernet  HWaddr 00:E0:18:F3:D3:DF
          inet addr:172.16.200.12  Bcast:172.16.200.255  Mask:255.255.255.0
          ...
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          ...
vmnet8    Link encap:Ethernet  HWaddr 00:50:56:C0:00:08
          inet addr:172.16.51.1  Bcast:172.16.51.255  Mask:255.255.255.0
          ...
# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.51.0     0.0.0.0         255.255.255.0   U     0      0        0 vmnet8
172.16.200.0    0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.168.16.0    0.0.0.0         255.255.240.0   U     0      0        0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         192.168.16.1    0.0.0.0         UG    0      0        0 eth0
#
```

La couche réseau de *TCP/IP*

- **Les routeurs (passerelles)**
  - Dispositif de la couche *réseau* (niveau 3 *OSI*)
  - Dispose de plusieurs interfaces
    - Sait faire passer les paquets de l'une à l'autre
    - *TTL*-- (64 $\rightarrow$ 0), paquet détruit si *TTL* nul (*ICMP*), $\rightarrow$ permet de stopper les boucles
  - *Paquet* routé entrant : destination *IP* $\neq$ destination *MAC*
  - *Paquet* routé sortant : source *IP* $\neq$ source *MAC*
  - Routage statique : table de routage définie *"en dur"*
    - Convient à un usage local
  - Routage dynamique : mise à jour de la table de routage
    - Protocoles dédiés *RIP*, *EGP*, *BGP* ...
    - Nécessaire pour le routage d'*Internet*

La couche réseau de *TCP/IP*

- **Exemple d'utilisation de routeurs, installation simple**
  - Attribution d'une adresse externe fixe
  - Attribution d'un réseau accessible depuis l'extérieur
  - Nb : si connexion par *modem* (Fournisseur d'Accès à Internet)
    - Attribution d'une seule adresse externe (non-fixe en général) !
    - Remplacer `195.221.233.0/24` par un sous-réseau privé
    - Utiliser *NAT* (*Network Address Translation*) sur `gate`, voir le cours *Firewall*
    - Rien n'est accessible depuis l'extérieur (sauf éventuellement si l'adresse externe est connue)



La couche réseau de *TCP/IP*

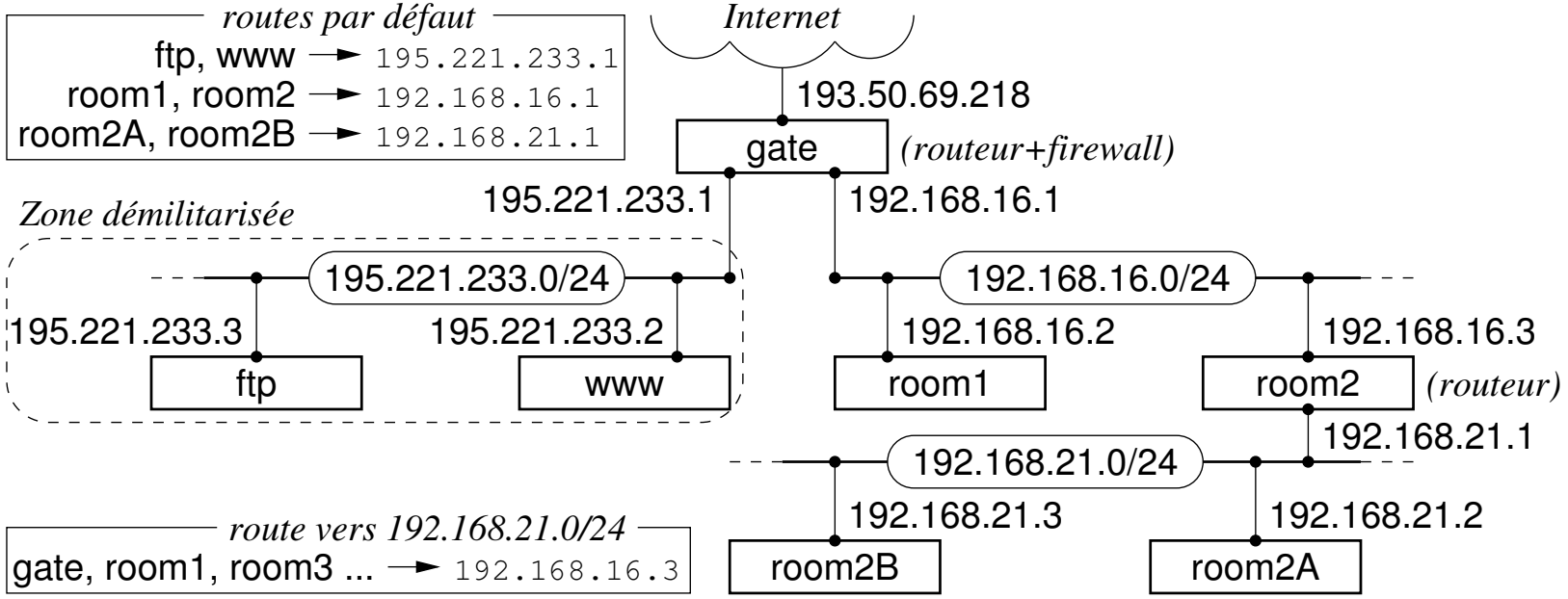
▷ **Exemple d'utilisation de routeurs, installation plus classique**

◇ Réseau `195.221.233.0/24` accessible depuis l'extérieur

○ *DMZ (De-Militarized Zone)*

◇ Réseaux `192.168.X.0/24` privés

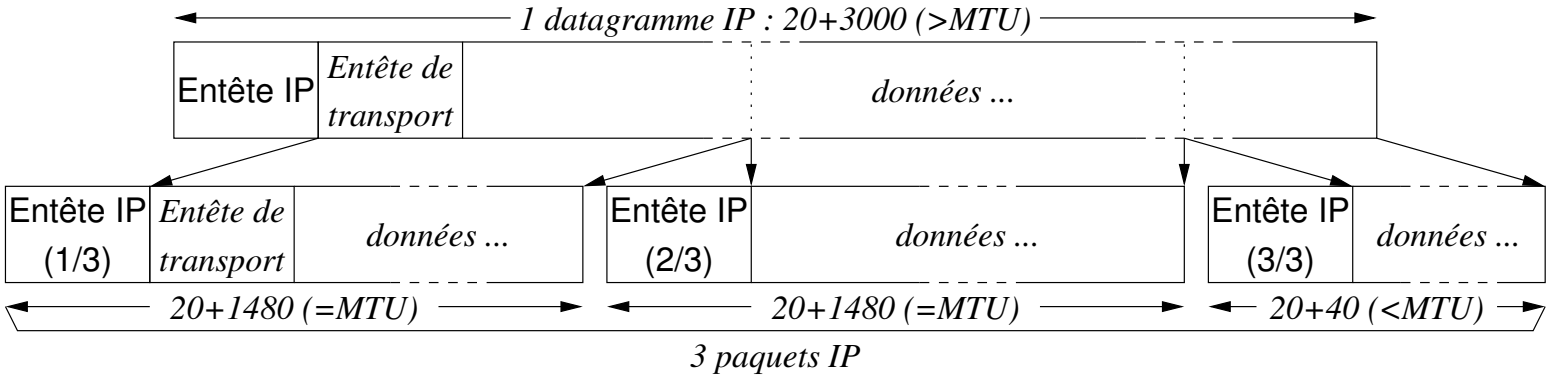
○ *NAT (Network Address Translation)*, voir le cours *Firewall*



La couche réseau de *TCP/IP*

▷ La fragmentation

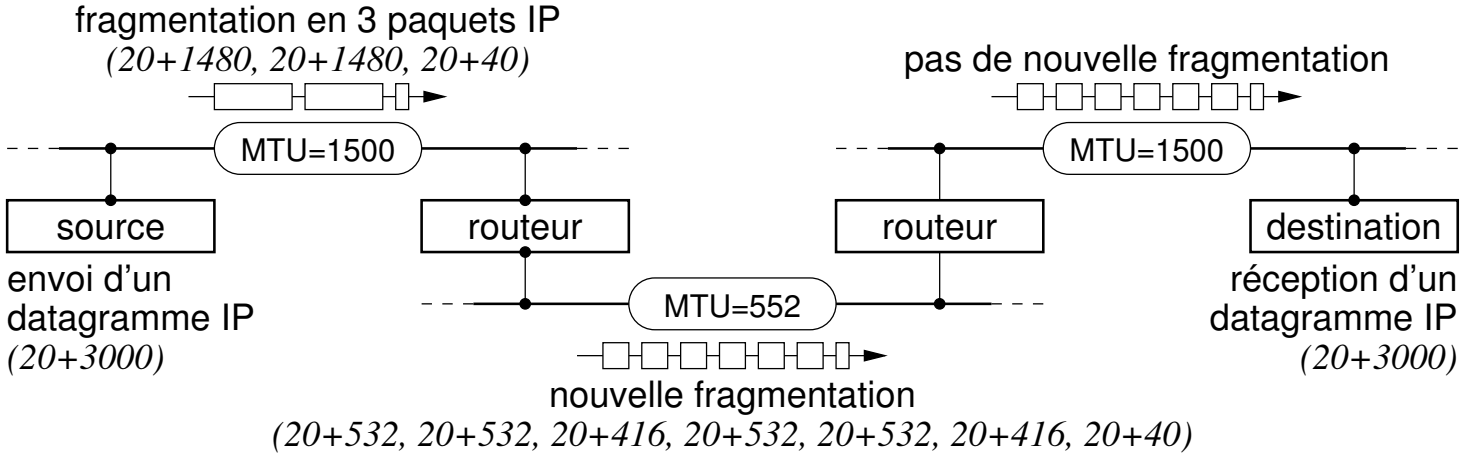
- ◇ Respecter les *MTU* de la couche liens
- ◇ Découper les *datagrammes IP* en *paquets IP* et les réassembler
  - ○ *datagramme* : unité *"logique"* du point de vue de la couche réseau
  - ○ *paquet* : unité *"physique"* du point de vue de la couche liens
- ◇ Communication de *"bout-en-bout"* au niveau réseau
  - ○ La fragmentation, le réassemblage sont transparents



La couche réseau de *TCP/IP*

▷ La fragmentation

- ◇ Un *paquet IP* peut être re-fragmenté
  - ○ *MTU* intermédiaire < *MTU* d'origine
- ◇ Réassemblage réalisé par le destinataire uniquement
  - ○ Un *paquet* perdu → retransmettre tout le *datagramme* (nb : on ne le sait pas à ce niveau, *IP* est *"non fiable"*)



La couche réseau de *TCP/IP*

- **La fragmentation : les champs de l'entête *IP* (suite)**
  - *identification* : constant pour chaque fragment d'un *datagramme*
  - *drapeaux* : 1 bit à 1, 2 bits significatifs
    - *"don't fragment"* : échec (*ICMP*) si besoin de fragmenter
    - *"more fragments"* : d'autres fragments suivent
  - *décalage* : position des données du *paquet* dans le *datagramme*
- **Réassemblage des *paquets* en *datagrammes***
  - Si *décalage* non nul, vérifier qu'on a reçu les *paquets* précédents
  - Si *"more fragments"*, vérifier qu'on a reçu les *paquets* suivants
  - Mêmes *source/identification*, enchaînement *taille/décalage* ok, pas de *"more fragments"* pour le dernier
    $\rightarrow$ *datagramme* complet
  - *Timeout* pour oublier les *datagrammes* incomplets

La couche réseau de *TCP/IP*

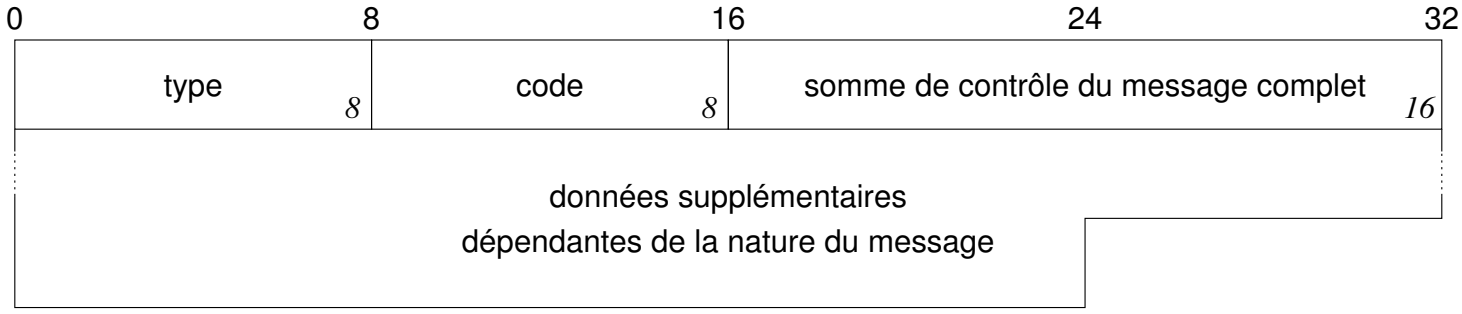
▷ **Les messages** *ICMP* **(***Internet Control Message Protocol***)**

◇ Au dessus d'*IP* mais étroitement lié, proto.*IP*=1

◇ Un *"jeu"* de messages diversifié (erreurs, mise au point ...)

◇ Format de message très dépendant du type (pas de détails ici)

type=8	code=0	: *"echo request"*
type=0	code=0	: *"echo reply"*
type=3	code=1	: *"no route to host"*
type=3	code=3	: *"connection refused"*
type=3	code=4	: fragmentation nécessaire mais interdite
type=5	code=0-3	: redirection, une meilleure route existe
type=11	code=0,1	: le champ *TTL* a atteint 0
...	...	...



La couche réseau de *TCP/IP*

- **Les messages *ICMP*, le programme `ping`**
  - Tester si une adresse *IP* est joignable (machine ou diffusion)
  - Envoie un *ICMP "echo request"* et attend l'*ICMP "echo reply"*
  - `-R` pour noter les adresses de sortie (dans les options *IP*, 9 max.)

```
# ping 195.221.233.9
PING 195.221.233.9 (195.221.233.9) 56(84) bytes of data.
64 bytes from 195.221.233.9: icmp_seq=1 ttl=254 time=2.37 ms
64 bytes from 195.221.233.9: icmp_seq=2 ttl=254 time=1.20 ms
--- 195.221.233.9 ping statistics ---                              <---[Ctrl-C]
2 packets transmitted, 2 received, 0% packet loss, time 1005ms
rtt min/avg/max/mdev = 1.201/1.788/2.376/0.589 ms
# ping -R 195.221.233.9
PING 195.221.233.9 (195.221.233.9) 56(124) bytes of data.
64 bytes from 195.221.233.9: icmp_seq=1 ttl=254 time=3.11 ms
RR:     192.168.20.236
        195.221.233.1
        195.221.233.9
        195.221.233.9
        192.168.16.1
        192.168.20.236
64 bytes from 195.221.233.9: icmp_seq=2 ttl=254 time=2.72 ms    (same route)
--- 195.221.233.9 ping statistics ---                              <---[Ctrl-C]
2 packets transmitted, 2 received, 0% packet loss, time 1012ms
rtt min/avg/max/mdev = 2.723/2.920/3.118/0.204 ms
```

La couche réseau de *TCP/IP*

- **Les messages *ICMP*, le programme `traceroute`**
  - Lister les routeurs qui permettent d'atteindre une machine cible
  - Envoi d'un datagramme *IP* vers la cible avec un *TTL* de 1, 2, 3 ...
    - Routeur qui met *TTL* à 0 envoie un *ICMP* (type=11) à la source
    - L'adresse *IP* d'entrée du routeur est dans l'*ICMP* en question
    - Incrémenter *TTL* jusqu'à atteindre la cible
  - `-I` pour envoyer un *ICMP "echo request"*
    (par défaut *UDP*/33434 risque d'être filtré par les *firewalls*)

```
# traceroute -In 192.44.75.206
traceroute to 192.44.75.206 (192.44.75.206), 30 hops max, 38 byte packets
 1  192.168.16.1  1.241 ms  1.252 ms  1.361 ms
 2  192.168.128.20  4.301 ms  3.292 ms  3.326 ms
 3  193.50.69.217  3.272 ms  3.122 ms  3.142 ms
 4  193.48.78.29  3.904 ms  3.735 ms  3.755 ms
 5  193.48.78.18  4.996 ms  4.964 ms  4.993 ms
 6  193.50.69.90  5.672 ms  5.674 ms  5.355 ms
 7  192.44.75.206  6.018 ms  4.592 ms  4.422 ms
#
```

La couche réseau de *TCP/IP*

- **Bilan/limitations de *IP***
  - *IP* est lié à d'autres protocoles (*ARP*, *RARP*, *ICMP*, *IGMP*)
  - Permet de joindre une machine quelconque sur *Internet*
    - Quels que soient les procédés de la couche liens
    - Même si on ne sait pas exactement par où passer
    - La fragmentation et le réassemblage sont transparents (s'il est délivré, un datagramme *IP* reste un datagramme)
    - Comment connaît-on l'adresse *IP* de la cible ? (*DNS*)
  - Permet la diffusion au sein d'un réseau local
    - Adresse *IP* destination : adresse de diffusion du réseau
    - Nécessairement contenu dans une trame diffusée
  - Protocole *"non fiable"*
    - Quelques indications d'échec (*ICMP*)
    - On ne sait pas qu'un datagramme *IP* n'a pas été reçu

La couche transport de *TCP/IP*

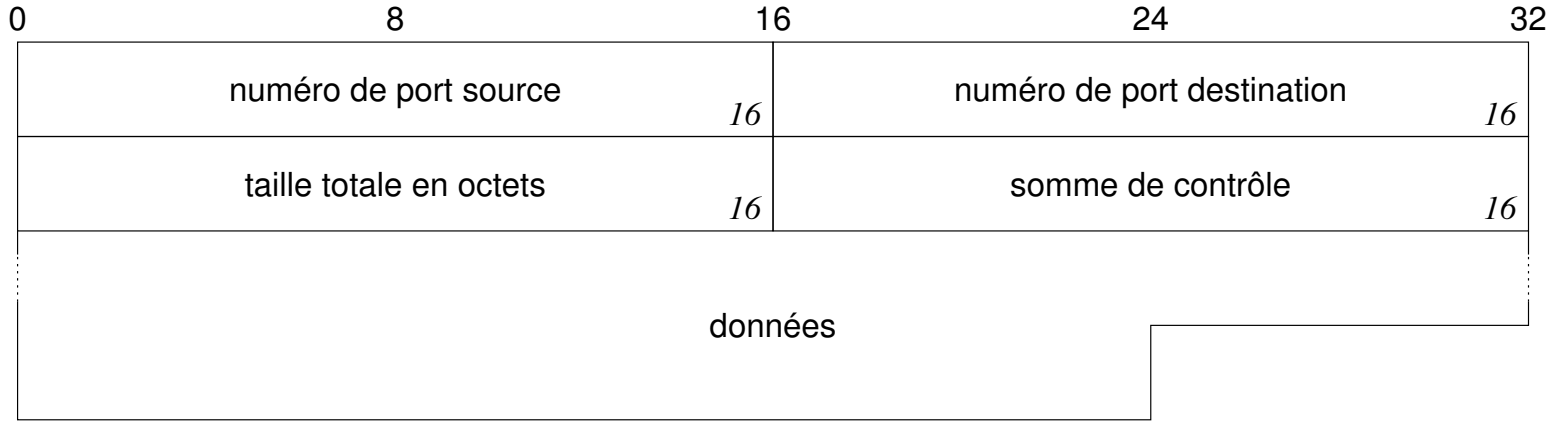
- **Deux protocoles de transport sont principalement utilisés**
  - Utilisables par les applications non privilégiées
  - *UDP* : mode non connecté
  - *TCP* : mode connecté
- **Dé/multiplexer selon un** *numéro de port*
  - Ne pas s'adresser à la machine dans sa globalité
  - S'adresser à un service particulier de cette machine
  - Numéro de port : entier de 16 bits
    - *"Well Known Ports"* : $0 \rightarrow 1023$ (privilèges requis)
    - *"Registered Ports"* : $1024 \rightarrow 49151$
    - *"Dynamic and/or Private Ports"* : $49152 \rightarrow 65535$
    - Voir `www.iana.org` (ou `/etc/services`)

La couche transport de *TCP/IP*

- **Le protocole** *UDP* **(***User Datagram Protocol***)**
  - Permet d'échanger des *datagrammes* (vocabulaire !)
  - Dé/multiplexage selon les *numéros de port*
  - Somme de contrôle calculée sur l'ensemble du datagramme (*IP* : uniquement sur les entêtes des paquets)
  - Protocole *"non fiable"*
    - Si *IP* ne délivre pas un datagramme, *UDP* n'en sait rien
  - $UDP \simeq IP$ + numéros de ports + somme de contrôle
    - Fonctionne en mode non connecté
    - Peu coûteux en traffic

La couche transport de *TCP/IP*

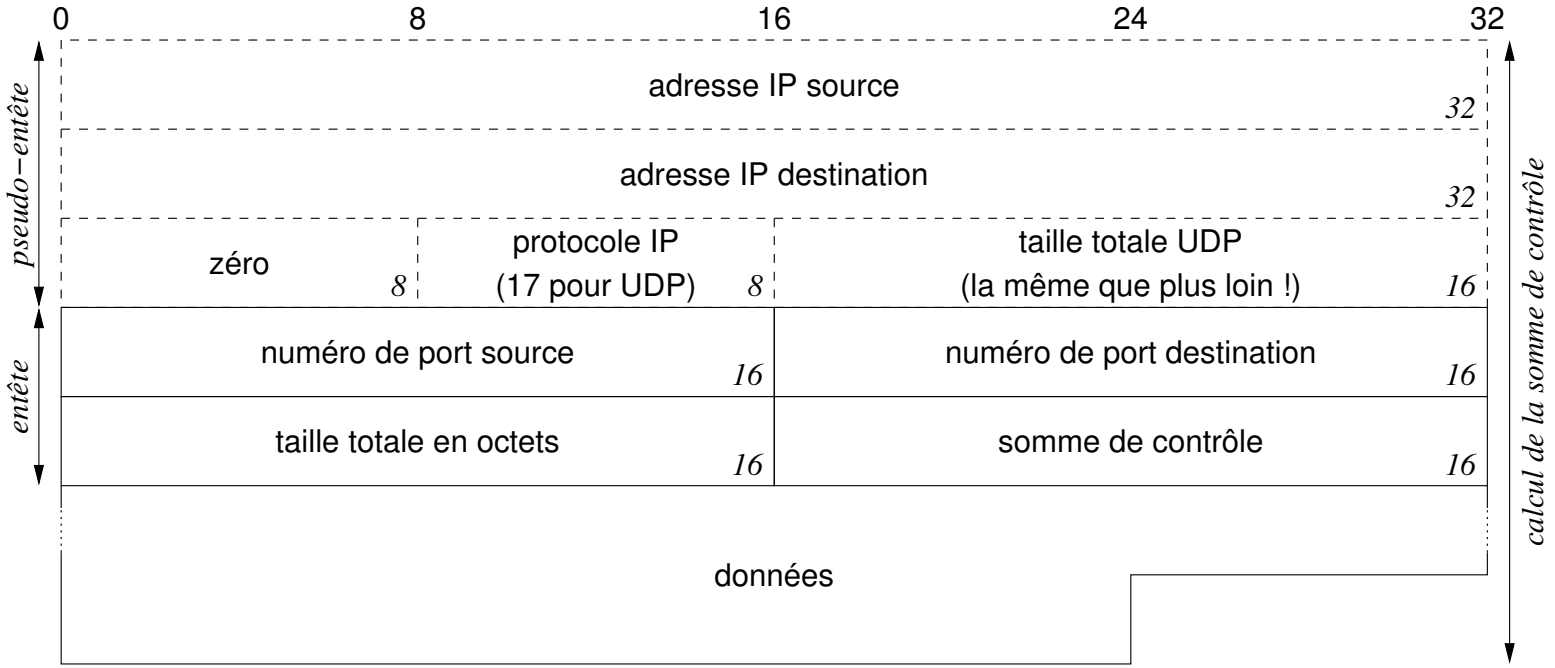
- **Le** *datagramme UDP*
  - Il constitue la partie *données* du datagramme *IP*
  - *ports* : ...
  - *taille* : entête (8) + données
  - *somme de contrôle* : *"pseudo"*-entête + entête + données
    - Inutilisée si nulle (facultative)



La couche transport de *TCP/IP*

▷ **Le *"pseudo"*-entête *UDP***

- ◇ N'existe pas ! Infos extraites du datagramme *IP*
- ◇ Juste utilisé dans la calcul de la somme de contrôle
- ◇ Permet des vérifications redontantes (erreurs de routage ...)



La couche transport de *TCP/IP*

- **Taille maximale des données d'un datagramme** *UDP*
  - Limite théorique imposée par *IP*
    - Taille du datagramme *IP* sur 16 bits
    - Entête *IP* de 20 à 60 octets
    - Entête *UDP* 8 octets
    - Il devrait donc rester 65507 octets de données *UDP*
  - En pratique, limite imposée par la taille des tampons
    - Généralement dimmensionnés pour 8192 octets de données
    - Ajustable par des appels systèmes (API *socket*)
  - Les datagrammes *IP* **doivent** supporter 576 octets de données
    - Beaucoup d'appli. se limitent à 512 octets de données *UDP*
    - Limitation volontaire, démarche très prudente

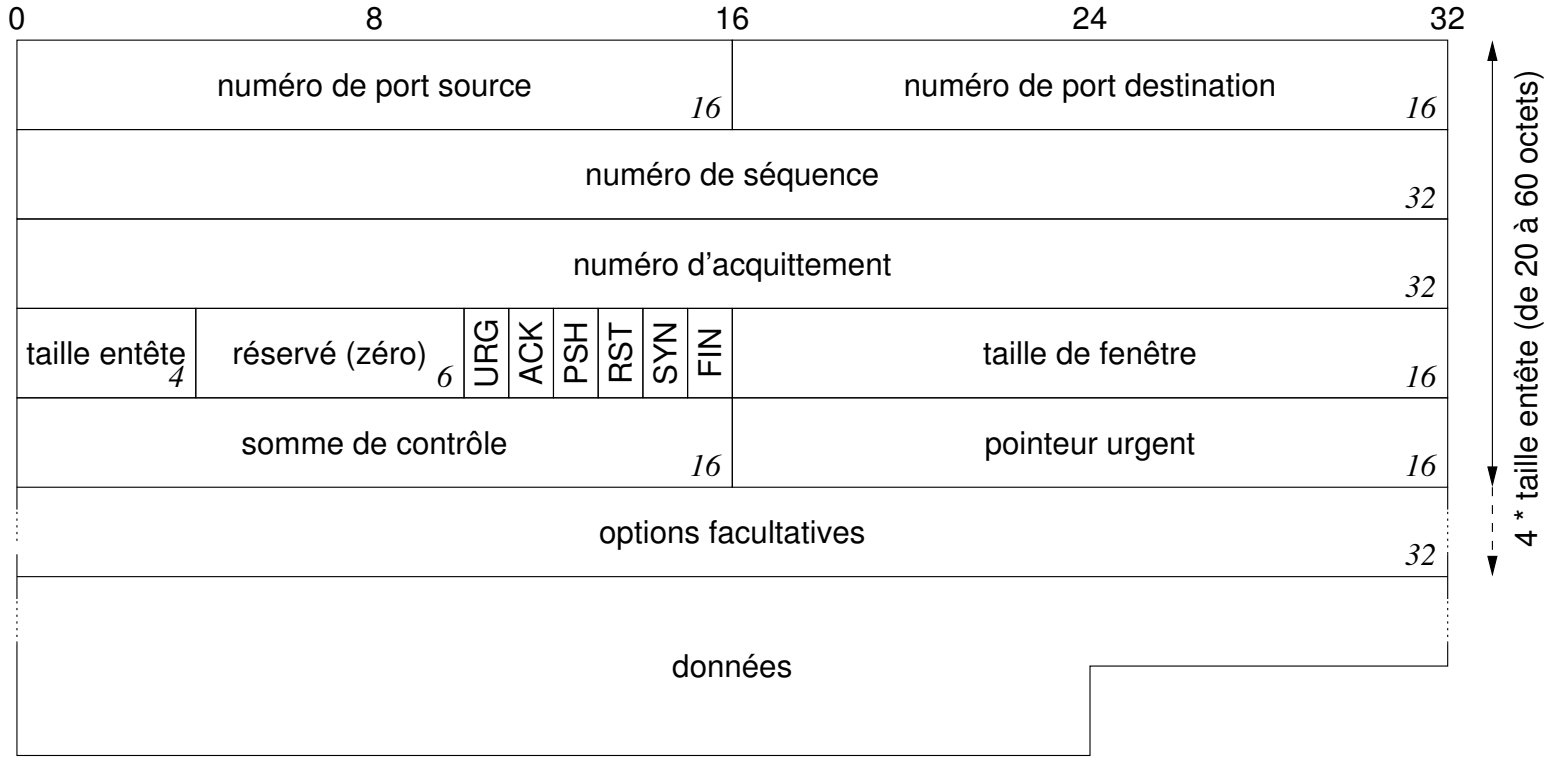
La couche transport de *TCP/IP*

- **Le protocole** *TCP* **(***Transmission Control Protocol***)**
  - Orienté connexion
    - Ouverture/fermeture explicite de la connexion
    - Transmission bidirectionnelle de flots d'octets (*byte stream*) (peut s'apparenter à la communication par tube ...)
    - Le flot est découpé en *segments TCP* (vocabulaire !)
  - Protocole *"fiable"*
    - Chaque *segment* transmis a un numéro de séquence
    - Mécanisme d'acquitement pour chaque *segment* transmis
    - Retransmission si nécessaire (*timeouts*)
    - Échec explicite si problème grave

La couche transport de *TCP/IP*

▷ **Le** *segment TCP*

◇ Il constitue la partie *données* du datagramme *IP*



La couche transport de *TCP/IP*

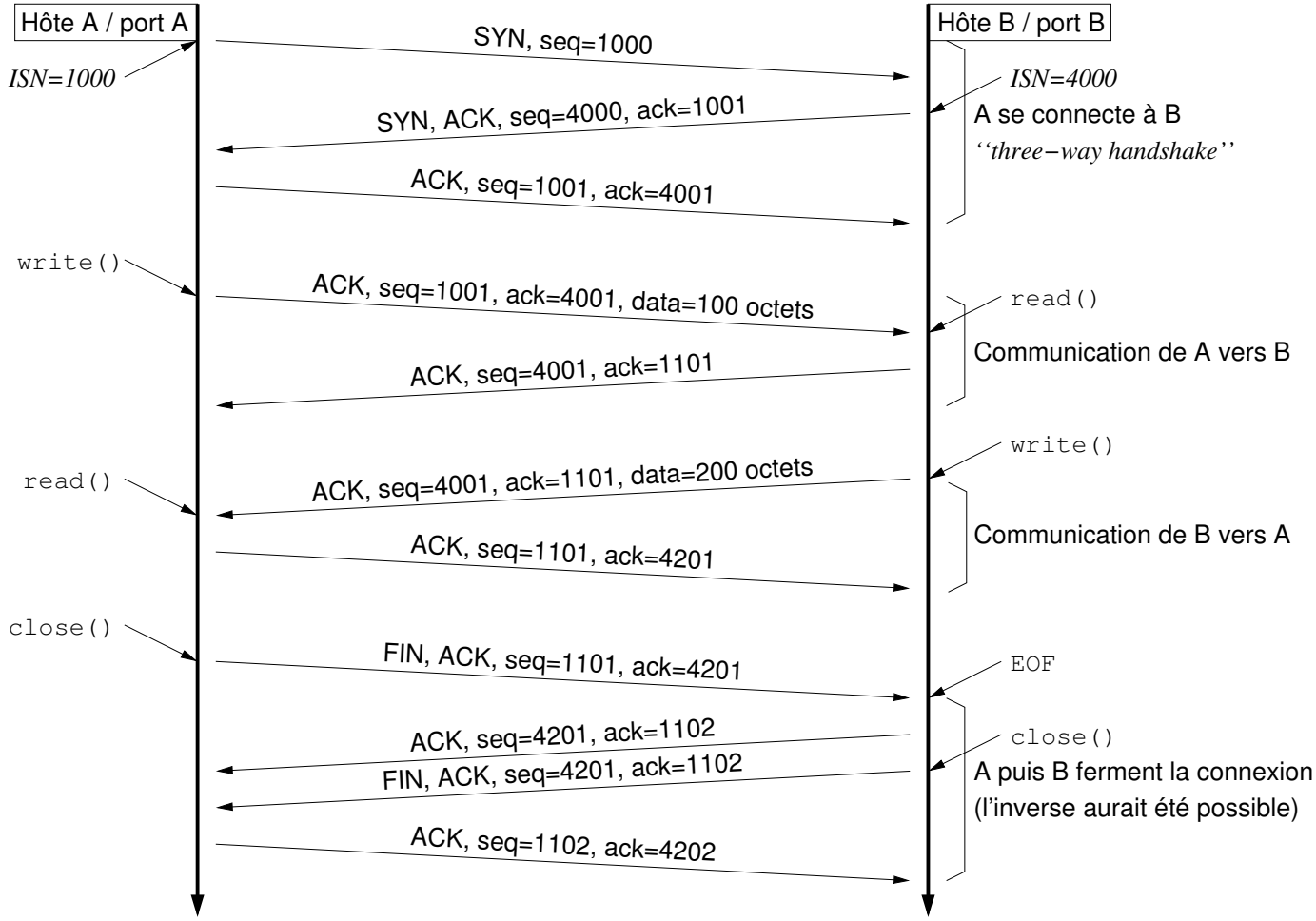
- **Le** *segment TCP*
  - *ports*, *séquence*, *acquittement* : ...
  - *taille entête* : nombre de mots de 32 bits de l'entête (de 5 à 15)
  - *URG* : le champ *pointeur urgent* est valide
  - *ACK* : le champ *acquittement* est valide
  - *PSH* : passer à l'application au plus tôt (tampons non pleins)
  - *RST* : demande de réinitialisation de la connexion
  - *SYN* : demande de synchronisation de *séquence* (init.)
  - *FIN* : fin de transmission
  - *fenêtre* : retard prévu pour l'acquittement (non traité ici)
  - *somme de contrôle* : *"pseudo"*-entête + entête + données
    - Comme pour *UDP* mais obligatoire
    - Proto.*IP*=6, taille calculée depuis le datagramme *IP*
  - *pointeur urgent* : données à traiter d'urgence (non traité ici)

La couche transport de *TCP/IP*

- **Numéros de *séquence* et d'*acquittement* *TCP***
  - Init. d'une connexion → choix d'un *ISN* (*Initial Sequence Number*)
    - Algorithme propre au système
    - Utilisé ensuite comme compteur de données
  - L'acquittement d'un segment reprend ce numéro
    - On lui ajoute le volume de données du segment reçu
    - ++ si le bit *SYN* ou *FIN* est positionné
    - Détermine le prochain numéro de séquence attendu
  - Chaque extrémité a son propre numéro de séquence
    - Permet une communication bidirectionnelle
  - Les segments échangés doivent **toujours** être acquittés
  - Génère plus de traffic que de simples datagrammes
  - Retransmission après un *timeout* si pas d'acquittement
    - Risque de *"doublons"* si transmission lente, *TCP* les élimine

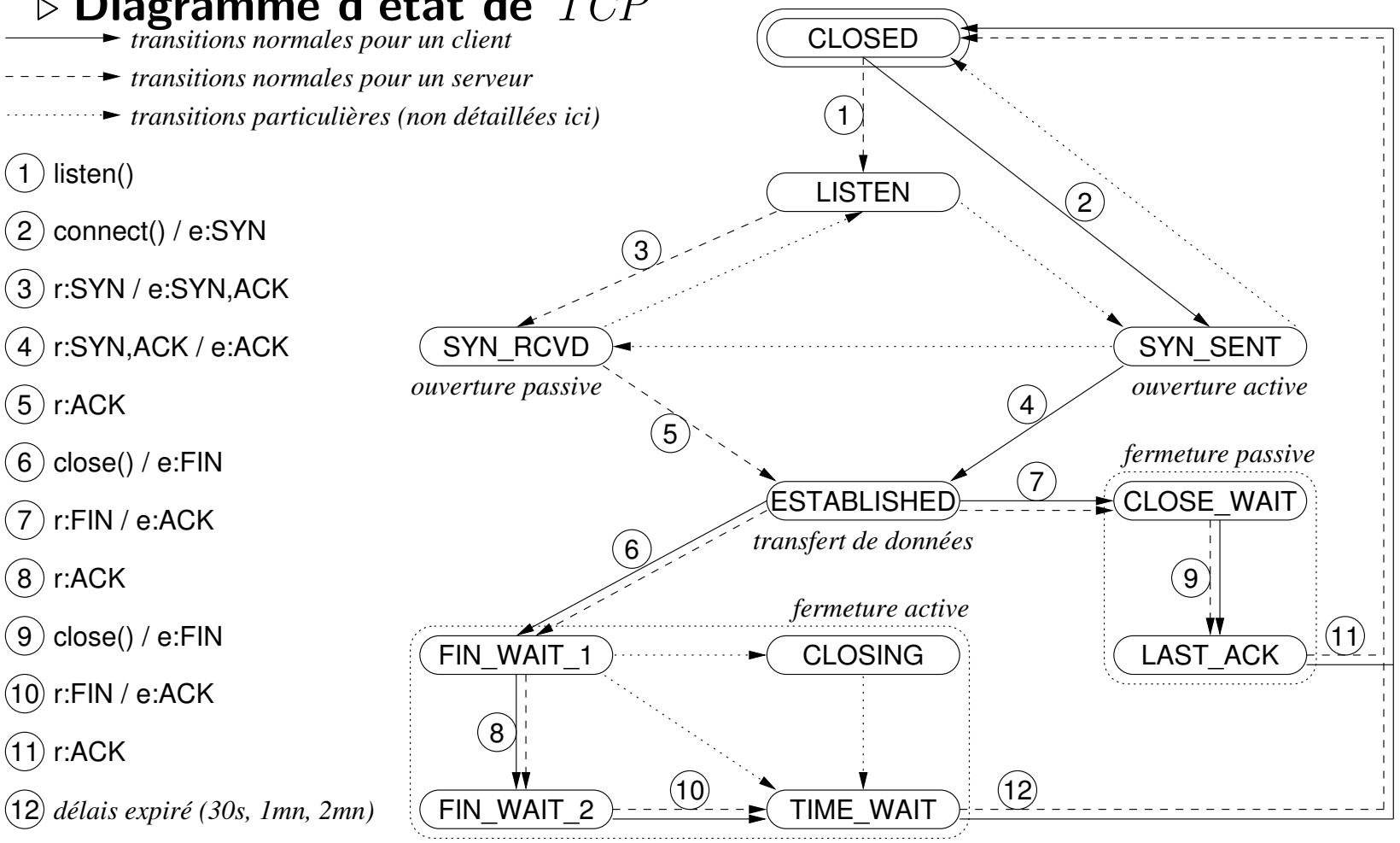
La couche transport de *TCP/IP*

▷ Diagramme temporel d'une communication *TCP*



La couche transport de *TCP/IP*

▷ Diagramme d'état de *TCP*



La couche transport de *TCP/IP*

- **Ouverture d'une connexion** *TCP*
  - Envoi de *SYN* (ouverture active)
  - Réponse avec *SYN* également (ouverture passive)
- **Tentative de connexion** *TCP* **à un port inexistant**
  - Aucun processus *"n'écoute"* sur ce port
  - Un *RST* est envoyé en réponse au *SYN*
- **Fermeture normale d'une connexion** *TCP*
  - Le tampon d'écriture est envoyé, puis *FIN*
- **Avortement d'une connexion** *TCP*
  - Envoi de *RST*, le tampon d'écriture est perdu
- **Timer** *keep alive* **en cas de matériel débranché, arrêté**
  - Connexion ouverte jusqu'à *FIN* ou *RST*
  - Envoi de temps en temps de segments vides (*ACK* attendu)

La couche transport de *TCP/IP*

- **Bilan/limitations de *UDP* et *TCP***
  - Protocoles accessibles aux processus non privilégiés
  - Choix de l'un ou l'autre selon les besoins applicatifs
    - *TCP* : fiable, flot bidirectionnel, coûteux
    - *UDP* : non-fiable, orienté messages, peu coûteux
      → données furtives à validité limitée (pertes acceptables)
    - Diffusion sur un réseau local uniquement en *UDP*
      → diffusion *IP* + filtrage selon numéro de port
  - Mise en œuvre plus détaillée dans le cours sur les *sockets*
  - Il existe d'autres protocoles de transport (*IP over IP* ...)

La couche application de *TCP/IP*

▷ **Une infinité d'applications . . .**

◇ Certaines sont normalisées, services *"de base"*

○ `telnet`, `ssh`, `dns`, `nfs`, `smtp` . . .

◇ Des processus (serveurs) *"écoutent"* sur des ports particuliers

○ `http` : 80/TCP, `dns` : 53/TCP/UDP, `X11` : 6000/TCP/UDP . . .

◇ Des processus (clients) établissent une connexion (*TCP*) ou envoient des messages (*UDP*) en vue d'obtenir le service

○ Données formatées selon un protocole applicatif

◇ Sous *UNIX*, possibilité d'utiliser `inetd`/`xinetd`

○ Évite de charger en mémoire une multitude de serveurs qui écoutent

○ `inetd`/`xinetd` écoute un ensemble de ports *TCP/UDP*
→ passe la main dès qu'il y a une connexion/un message

La couche application de *TCP/IP*

▷ La commande `netstat`

◇ Observer l'état des connexions

```
# netstat -pantu
Active Internet connections (servers and established)
Proto   Local Address        Foreign Address   State         PID/Program name
tcp     0.0.0.0:929          0.0.0.0:*         LISTEN        1174/rpc.mountd
tcp     0.0.0.0:935          0.0.0.0:*         LISTEN        1177/rpc.statd
tcp     0.0.0.0:139          0.0.0.0:*         LISTEN        1211/smbd
tcp     0.0.0.0:973          0.0.0.0:*         LISTEN        1220/amd
tcp     0.0.0.0:111          0.0.0.0:*         LISTEN        1143/rpc.portmap
tcp     0.0.0.0:6000         0.0.0.0:*         LISTEN        1231/X
tcp     0.0.0.0:21           0.0.0.0:*         LISTEN        1218/proftpd: (acce
tcp     0.0.0.0:22           0.0.0.0:*         LISTEN        1152/sshd
tcp     0.0.0.0:631          0.0.0.0:*         LISTEN        1191/cupsd
tcp     127.0.0.1:43189      127.0.0.1:139     ESTABLISHED   15451/smbclient
tcp     127.0.0.1:21         127.0.0.1:43187   ESTABLISHED   15411/proftpd: harr
tcp     127.0.0.1:139        127.0.0.1:43189   ESTABLISHED   15452/smbd
tcp     127.0.0.1:43187      127.0.0.1:21      ESTABLISHED   15410/ftp
...
```

La couche application de *TCP/IP*

▷ La commande `netstat`

```
...
udp     0.0.0.0:32768         0.0.0.0:*                   -
udp     0.0.0.0:2049          0.0.0.0:*                   -
udp     172.16.200.12:137     0.0.0.0:*                   1213/nmbd
udp     172.16.51.1:137       0.0.0.0:*                   1213/nmbd
udp     192.168.20.236:137    0.0.0.0:*                   1213/nmbd
udp     0.0.0.0:137           0.0.0.0:*                   1213/nmbd
udp     172.16.200.12:138     0.0.0.0:*                   1213/nmbd
udp     172.16.51.1:138       0.0.0.0:*                   1213/nmbd
udp     192.168.20.236:138    0.0.0.0:*                   1213/nmbd
udp     0.0.0.0:138           0.0.0.0:*                   1213/nmbd
udp     0.0.0.0:926           0.0.0.0:*                   1174/rpc.mountd
udp     0.0.0.0:800           0.0.0.0:*                   -
udp     0.0.0.0:929           0.0.0.0:*                   1177/rpc.statd
udp     0.0.0.0:932           0.0.0.0:*                   1177/rpc.statd
udp     127.0.0.1:32952       0.0.0.0:*                   15452/smbd
udp     0.0.0.0:68            0.0.0.0:*                   27292/dhcpcd
udp     0.0.0.0:974           0.0.0.0:*                   1220/amd
udp     0.0.0.0:111           0.0.0.0:*                   1143/rpc.portmap
udp     0.0.0.0:1022          0.0.0.0:*                   1220/amd
udp     0.0.0.0:1023          0.0.0.0:*                   1220/amd
#
```

La couche application de *TCP/IP*

- **Le système de noms de domaines** *DNS* **(***Domain Name System***)**
  - Adresses *IP* difficiles à manipuler pour les utilisateurs
    - On préfère utiliser des mots qui évoquent quelque chose
    - Ce n'est pas un problème technique mais humain !
    - *TCP/IP* en ignore tout → utilisé par les applications
  - Un *nom de domaine* : liste de mots séparés 2 à 2 par un point
    - Chaque mot contient 63 caractères au maximum
    - Pas de distinction majuscule/minuscule
    - Un *FQDN* désigne une machine dans un réseau (*Fully Qualified Domain Name*)
    - ex : `www.enib.fr`, machine `www` dans le réseau `enib.fr`
    - Les points relatent la structure hiérarchique des domaines (`www` dans le domaine `enib`, lui-même dans le domaine `fr`)

La couche application de *TCP/IP*

- **Nom de domaine → adresse** *IP*
  - Fonction `gethostbyname()` de la `libc`
  - Consulter le fichier `/etc/hosts` (`C:\WINDOWS\HOSTS` sous *MDO*)
    - Associer une adresse *IP* à un (ou des) nom de domaine
    - Mise à jour manuelle envisageable pour un parc très réduit !
  - Interroger un serveur *DNS* (voir le fichier `/etc/resolv.conf`)
    - Interroger la machine désignée sur le port 53/UDP/TCP
    - Serveur *DNS* secondaire si le primaire est en panne
    - Compléter implicitement le nom de domaine si pas de point

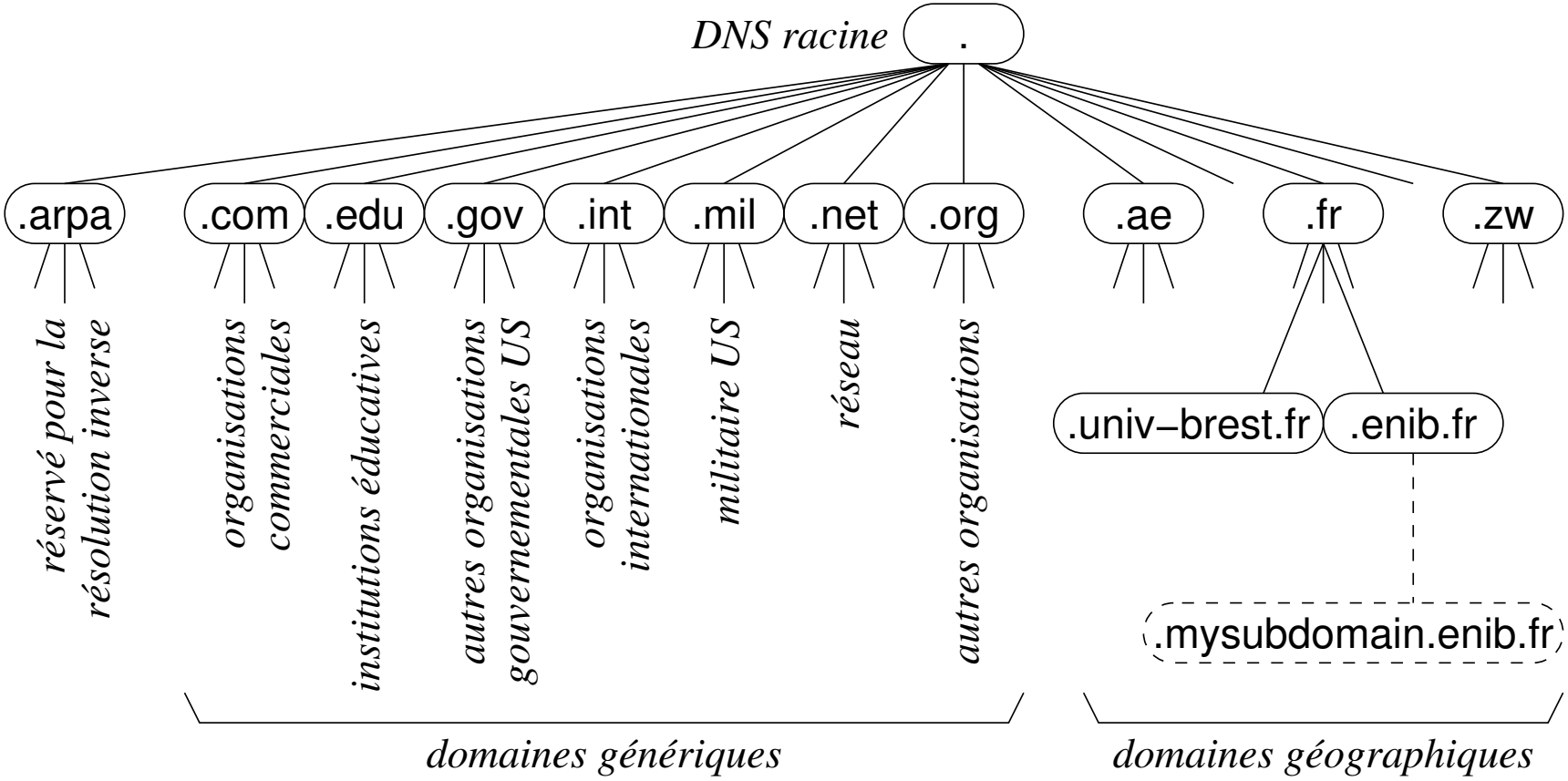
```
# cat /etc/hosts
127.0.0.1         localhost
192.168.20.236    nowin nowin.enib.fr
192.168.21.221    winout winout.enib.fr
# cat /etc/resolv.conf
nameserver 192.168.18.4
nameserver 192.168.18.3
search enib.fr
#
```

La couche application de *TCP/IP*

- **Le serveur** *DNS*
  - Maintient une liste semblable à `/etc/hosts`
    - Uniquement pour les machines de la zone d'autorité
  - Autres noms de domaine ? Demander à un *DNS racine*
    - ex : `.enib.fr` → `.`
  - Le *DNS racine* connaît les *DNS* des zones inférieures, etc
    - ex : `.` → `.fr` → `.enib.fr`, `.univ-brest.fr` ...
  - Il s'agit d'une base de données distribuée
    - Chaque serveur *DNS* ne maintient que des informations partielles
    - L'arborescence des *DNS* donne accès à l'ensemble
    - Les serveurs ont un cache avec une durée d'expiration (éviter de refaire la recherche complète à chaque fois)

La couche application de *TCP/IP*

▷ L'arborescence des serveurs *DNS*



La couche application de *TCP/IP*

▷ **Exemple : rapatrier la page** `http://www.opengl.org`

→ Requête *ARP* pour le *DNS* local
← Réponse *ARP* pour le *DNS* local
→ Requête *DNS* pour `www.opengl.org`
(*DNS.enib.fr* ↔ routeur ↔ . . . *Internet*. . . ↔ routeur ↔ *DNS.opengl.org*)
← Réponse *DNS* pour `www.opengl.org`
→ Requête *ARP* pour le routeur
← Réponse *ARP* pour le routeur
→ *SYN* vers `www.opengl.org`/80 via le routeur (. . .)
← (. . .) *SYN/ACK* depuis `www.opengl.org` via le routeur
→ *ACK* vers `www.opengl.org` via le routeur (. . .)
→ Requête *HTTP* vers `www.opengl.org` via le routeur (. . .)
← (. . .) *ACK* depuis `www.opengl.org` via le routeur
← (. . .) Réponse *HTTP* depuis `www.opengl.org` via le routeur
→ *ACK* vers `www.opengl.org` via le routeur (. . .)
← (. . .) *FIN/ACK* depuis `www.opengl.org` via le routeur
→ *ACK* vers `www.opengl.org` via le routeur (. . .)
→ *FIN/ACK* vers `www.opengl.org` via le routeur (. . .)
← (. . .) *ACK* depuis `www.opengl.org` via le routeur

Annexe : Vocabulaire pour les unités de données

- **Couche liens,** *Ethernet*, *802.2/802.3*
  - Une trame
- **Couche réseau,** *IP*
  - Du point de vue de la couche inférieure : un paquet
  - Du point de vue de la couche supérieure : un datagramme
- **Couche transport,** *UDP*
  - Du point de vue de la couche inférieure : un datagramme
  - Du point de vue de la couche supérieure : un message
- **Couche transport,** *TCP*
  - Du point de vue de la couche inférieure : un segment
  - Du point de vue de la couche supérieure : un flot

Annexe : Somme de contrôle sur 16 bits

- **Utilisé dans les entêtes** *IP*, *UDP* **et** *TCP*
  - Champ *somme de contrôle* $\in$ données du calcul !!!
    - Il **faut** mettre ce champ à zéro **avant** d'effectuer le calcul
  - Volume de données concerné pas nécessairement pair

```
unsigned short
checksum16(const void * addr,
           unsigned short size)
{
unsigned long sum=0;                          // somme sur 32 bits si retenues
while(size>1)
  {
  sum+=*(((const unsigned short *)addr)++);   // additionner des entiers
  size-=sizeof(unsigned short);               // de 16 bits
  }
if(size)                                      // un octet supplementaire ?
  {                                           // poids fort d'un entier
  sum+=*((const unsigned char *)addr);        // de 16 bits
  }
while(sum>>16)                                // ``replier'' la somme sur
  {                                           // 32 bits pour obtenir un
  sum=(sum&0x0000FFFF)+(sum>>16);             // entier de 16 bits
  }
return(~sum);                                 // le complement de ce calcul
}
```

Annexe : Quelques *RFC* (*Request For Comments*)

- **Descriptions détaillées de ce qui doit être normalisé**
  - On peut les obtenir sur `http://www.ietf.org/rfc.html`
- **Couche liens,** *Ethernet*, *802.2*/*802.3*
  - `RFC 894`, `RFC 1042`
- **Couche réseau,** *IP*, *ARP*, *RARP*, *ICMP*
  - `RFC 791`, `RFC 826`, `RFC 903`, `RFC 792`
- **Couche transport,** *UDP*, *TCP*
  - `RFC 768`, `RFC 793`
- **Couche application,** *DNS*
  - `RFC 1034`, `RFC 1035`