RESEAUX : Protocoles

1 Introduction

Les nombreuses applications précédentes fonctionnent toutes grâce à la notion de protocoles de communication.

Ce chapitre va étudier la **famille de protocoles TCP/IP** issus du monde Unix et largement répandus.

Ils ont été développés (dès 1970) sur l'initiative du département américain de la défense (*DoD Department of Defense*) par des chercheurs travaillant autour de DARPA *(Defense Advanced Research Project Agency*).

L'exemple le plus illustre en est le réseau ***Internet*** mis en service dans les années 80 et qui continue de s'étendre (plusieurs millions d'ordinateurs aujourd'hui).

D'abord réservé au milieu académique, ce réseau mondial a vite intéressé le monde des affaires.
Son immense succès est aussi lié à la technologie **WWW** *(World Wide Web*) qui autorise un non spécialiste à *surfer on the net.*

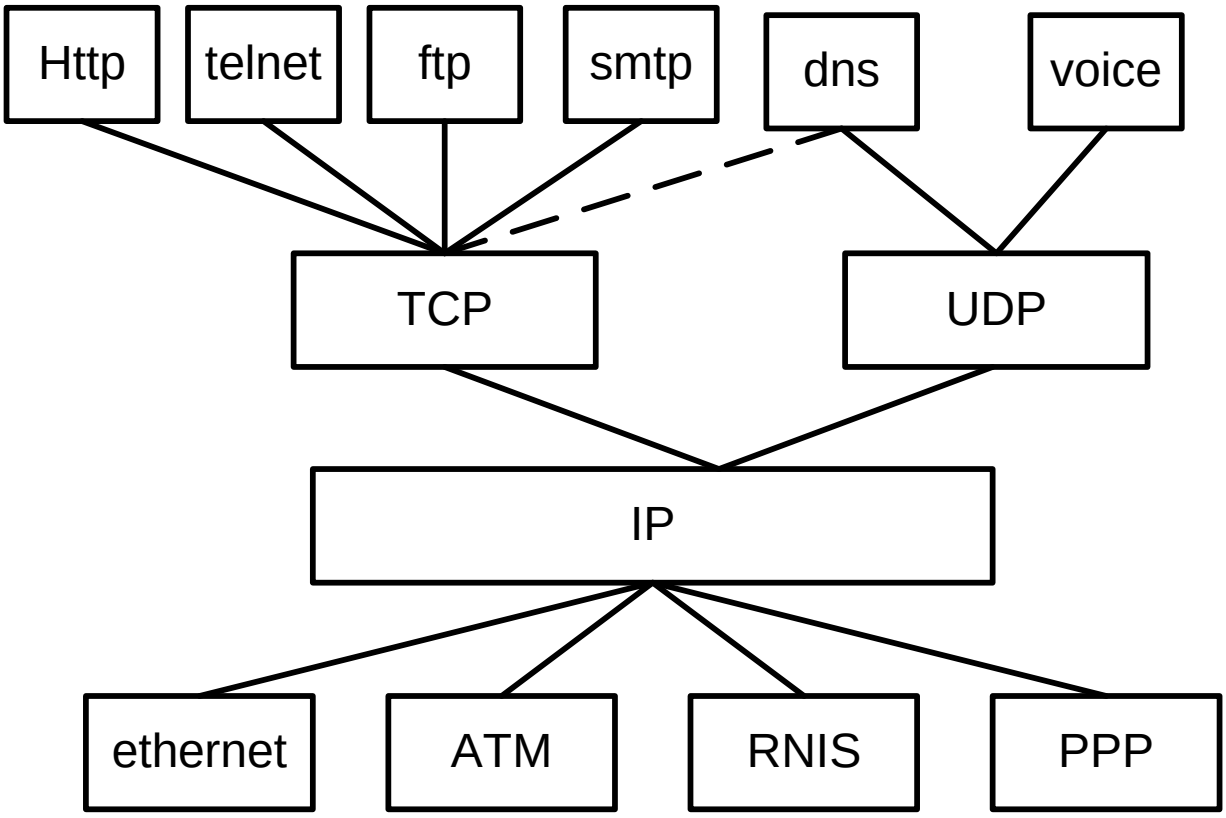
Avec l'*internet* commercial, ces protocoles sont livrés en standard dans les postes de travail actuels.

Les RFC (*Requests For Comment*), sortes de rapports techniques, décrivent ces différents protocoles.

a) Objectifs de l'architecture TCP/IP :

- Indépendance de la technologie du réseau
- Indépendance de l'ordinateur
- Protocoles d'applications

b) Famille de protocoles TCP/IP



- Protocoles d'applications : telnet, ftp, smtp, …
- Technologie du réseau : *ethernet*, ATM, RNIS, …

2 *Internet Protocol* (IP)

La figure précédente montre bien l'importance de cette couche qui cache le détail du réseau.

a) Adresse IP

adresse IP (32 bits) = partie ***Network*** + partie ***Host***
adresses source et destination

b) Classe A

0 Network	Host	Host	Host

adresses : 1.HH.HH.HH → 127.HH.HH.HH
max : 2^{24} *hosts* par *network*

c) Classe B

10 Network	Network	Host	Host

adresses : 128.NN.HH.HH → 191.NN.HH.HH
max : 2^{16} *hosts* par *network*

L'Uni. de Genève dispose d'une adresse 129.194.HH.HH

d) Classe C

110 Net.	Network	Network	Host

adresses : 192.NN.NN.HH → 223.NN.NN.HH
max : 2^{8} *hosts* par *network*

e) Adresse unique au niveau mondial

La partie *Network* est attribuée par l'office central **NIC** (*Network Information Center*) afin d'identifier chaque réseau de manière **unique.**

f) Administration des adresses IP

La configuration d'un noeud IP est encore souvent une opération manuelle : certain paramètre comme l'adresse IP, ... , exige une intervention humaine.

Des sources de problème sont dès lors possibles :

- adresse incorrecte (hors de l'intervalle alloué)
- adresse dupliquée

g) DHCP (*Dynamic Host Configuration Protocol*)

Lors du *boot*, un nœud va demander une adresse IP au serveur (DHCP).

Le protocole DHCP est également utilisé par un poste de travail distant *(remote access)* qui établit une connexion avec son fournisseur d'accès *internet (ISP Internet Service Provider).*

h) Protocole sans connexion

Cette couche fournit un **service non fiable** (comme *ethernet*) basé sur l'échange de datagrammes.

Il n'existe en effet aucune garantie pour que le datagramme IP arrive à destination.

De plus, chaque datagramme est géré indépendamment de tous les autres (sans numéro de séquence)

La gestion du contrôle d'erreur est minimale : lorsqu'un problème survient, IP rejette le datagramme et essaie de renvoyer un message ICMP (*Internet Control Message Protocol*) à la source.

Toute la fiabilité requise doit être assurée par la couche supérieure.

Ce service est qualifié de ***best effort***

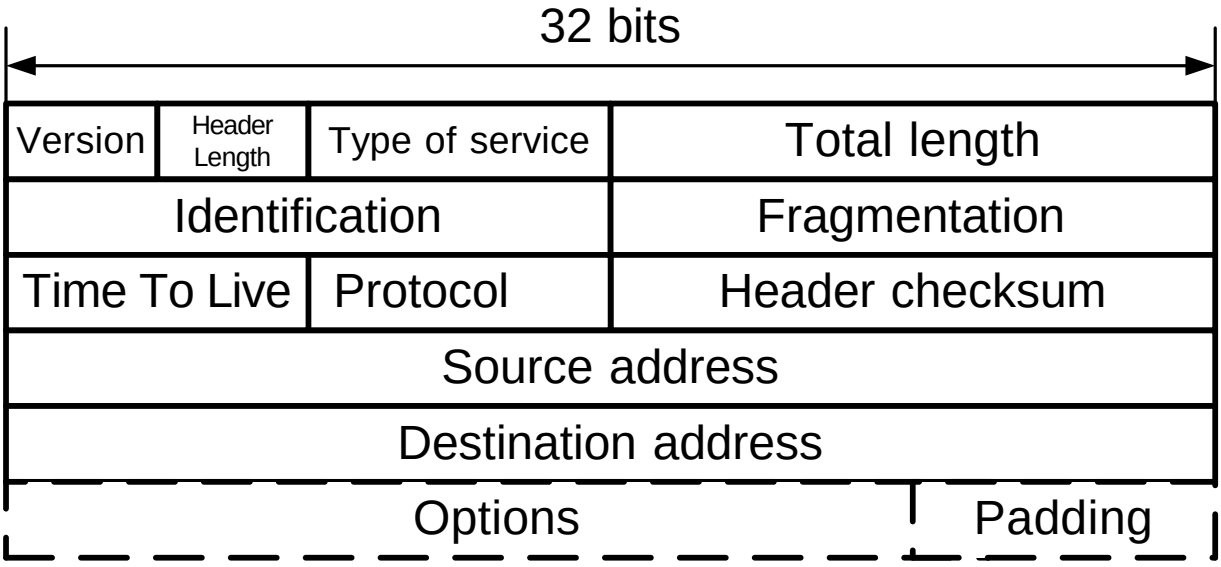
i) Couches supérieures

Deux chemins relient cette couche IP à la couche application.

TCP (*Transmission Control Protocol*) : un chemin sécurisé par un protocole orienté connexion pour la majorité des applications (telnet, ftp, smtp, ...)

UDP (*User Datagram Protocol*) : un chemin non-sécurisé pour des applications comme *voice over ip*, ...

j) Principaux champs de l'en-tête IP



La **version** courante du protocole est la version 4 et c'est pourquoi IP est parfois appelé IPv4.

La **longueur d'en-tête** (*header length*) est le nombre de mots de 32 bits figurant dans l'en-tête.
La valeur habituelle est de 20 octets signifiant l'absence d'option.

Dans le champ **type de service** (*TOS - type of service*), seuls 4 bits peuvent être utilisés pour :

- minimiser le délai
- maximalise le débit
- maximalise la fiabilité
- minimise le coût monétaire

Un seul bit parmi ces 4 bits peut être positionné à 1.
Les 4 bits mis à 0 signifient un service normal.

Les RFC 1340 et 1349 indiquent comment ces bits doivent être positionnés par les diverses applications :

- minimiser le délai telnet
- maximalise le débit ftp
- maximalise la fiabilité snmp (messagerie)
- minimise le coût monétaire nntp (news)

Le champ **longueur totale** (*total length*) contient la taille totale du datagramme IP.
Avec ce champ de 16 bits, la taille maximale d'un datagramme IP est donc de 65535 octets.
Bien que cette couche IP soit capable de fragmentation, la plupart des ordinateurs la pratique à un niveau supérieur.

Le champ **Identification** identifie de façon unique chaque datagramme envoyé par une machine qui l'incrémente simplement de un.

Le champ **TTL** (*time to live*) donne une limite supérieure au nombre de routeurs qu'un datagramme peut traverser.
Il limite ainsi la durée de vie du datagramme.
Il est initialisé à une certaine valeur par l'expéditeur (32 ou 64) et est décrémenté de un par chaque routeur traversé.
Lorsque ce champ atteint la valeur 0, le datagramme est rejeté et l'expéditeur est prévenu par un message ICMP.

Le champ ***protocol*** identifie le protocole de couche supérieure (TCP :6, UDP :17, ICMP :1, ...)

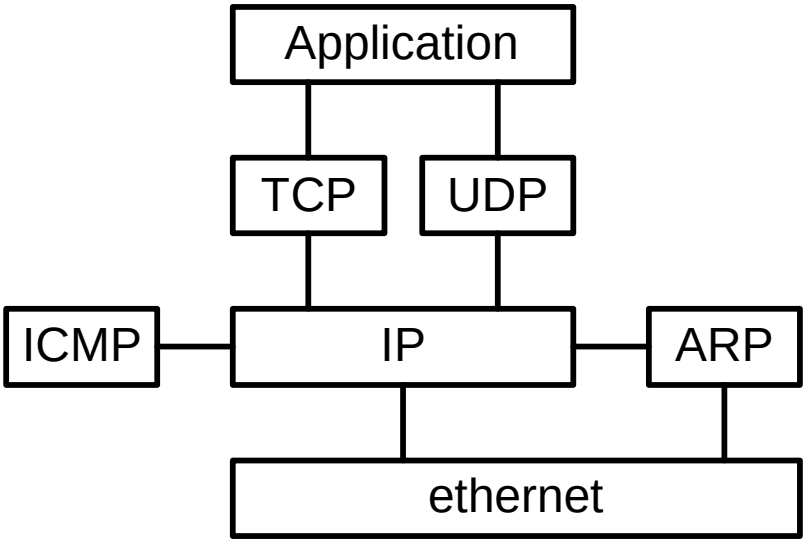
Chaque datagramme IP contient **l'adresse source** (32 bits) et **l'adresse destination** (32 bits).

k) Analyse de protocole IP entre AST2 et Alpha :

```
Internet Protocol
   Version (MSB 4 bits): 4
   Header length (LSB 4 bits): 5 (32-bit word)
   Service type: 0x00
      000. .... = 0 - Routine
      ...0 .... = Normal delay
      .... 0... = Normal throughput
      .... .0.. = Normal reliability
   Total length: 43 (Octets)
   Fragment ID: 41480
   Flags summary: 0x40
      0... .... = Reserved
      .1.. .... = Do not fragment
      ..0. .... = Last fragment
      Fragment offset(LSB 13 bits): 0 (0x00)
   Time to live: 32 seconds/hops
   IP protocol type: TCP (0x06)
   Checksum: 0x4474
   IP address 129.194.184.201 → 129.194.184.2
   No option
```

3 *Address Resolution Protocol* (ARP)

Le protocole **ARP** (*Address Resolution Protocol*) gère la relation entre **adresse logique (IP)** et **adresse physique (*ethernet*)** :



La couche IP, recevant une requête d'une des interfaces supérieures (TCP, UDP, ICMP), maintient une table de correspondance **(cache ARP)** entre adresses logique et physique :

1. Couche IP reçoit une requête d'une des interfaces supérieures
2. Adresse *ethernet* corespondante est-elle présente dans le **cache** ?
   Si non
   requête **ARP** vers toutes les stations (diffusion)
   réponse de la station
   mise à jour du **cache**
3. Envoyer la trame

Ex 1 Comment la couche *Ethernet* fait-elle pour envoyer le trafic sur le bon protocole (IP ou ARP) ?

Ex 2 Pourquoi certains ordinateurs, comme Solaris ou NT4, effectuent une requête ARP de leur propre adresse IP lors du *boot* ?

a) Analyse de protocole ARP entre AST2 et SUN :

```
Ethernet
    Destination Address:
    Source Address:
    Type: 0806 (ARP)
Address Resolution Protocol
    Hardware Type: 1 (Ethernet)
    Protocol Type: 800
    Hardware Address Length: 6
    Protocol Address Length: 4
    Operations: ARP Request
    Source Hardware Address: 00-60-8C-F2-15-30
    IP Source Address: 129.194.184.201
    Destination Hardware Address: 00-00-00-00-00-00
    IP Destination Address: 129.194.184.210

Ethernet
    Destination Address:
    Source Address:
    Type: 0806 (ARP)
Address Resolution Protocol
    Hardware Type: 1 (Ethernet)
    Protocol Type: 800
    Hardware Address Length: 6
    Protocol Address Length: 4
    Operations: ARP Response
    Source Hardware Address: 08-00-20-04-26-5E
    IP Source Address: 129.194.184.210
    Destination Hardware Address: 00-60-8C-F2-15-30
    IP Destination Address: 129.194.184.201
```

Ex 3

Ces paquets ARP correspondent au champ DATA du format ethernet.
Déterminer les champs *Destination Address* et *Source Address* correspondants.

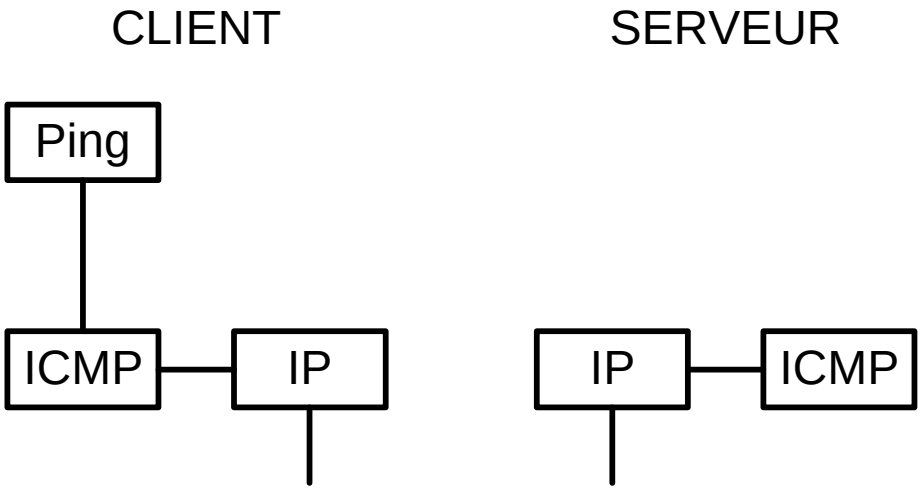
4 *Ping*

Cette application permet de vérifier que la communication avec une autre machine est possible.

Le client envoie un message ICMP : → **echo request**
Le serveur répond par : ← **echo reply**

Ping mesure également le **temps aller et retour**.

La plupart des implémentations de TCP/IP supporte le serveur ping directement dans le noyau :




```
Ping host 130.240.16.39 (ftp.luth.se)

Number      : 6
Timeout     : 15 [s]
Delay       : 1000 [ms]
Packet Size : 700 [byte]
```

```
#     Address          Response Time
1     130.240.16.39    221 ms
2     130.240.16.39    147 ms
3     130.240.16.39    204 ms
4     130.240.16.39    202 ms
5     130.240.16.39    112 ms
6     130.240.16.39    119 ms

Statistics:  Out 6, in 6, loss 0%
             times (min/avg/max) 112/167/221
```

Ce temps aller et retour **RTT** *(Round-Trip Time)* est un élément caractéristique des performances entre client et serveur.

a) Analyse de protocole ICMP entre AST2 et SUN :

```
Internet Control Message Protocol
      Type: 8 - Echo Request
      Code: 0
      Checksum: 0xBA5B
      Identifier: 256
      Sequence Number: 37376
      Optional Data: (32 bytes)

Internet Control Message Protocol
      Type: 0 - Echo Reply
      Code: 0
      Checksum: 0xC25B
      Identifier: 256
      Sequence Number: 37376
      Optional Data: (32 bytes)
```

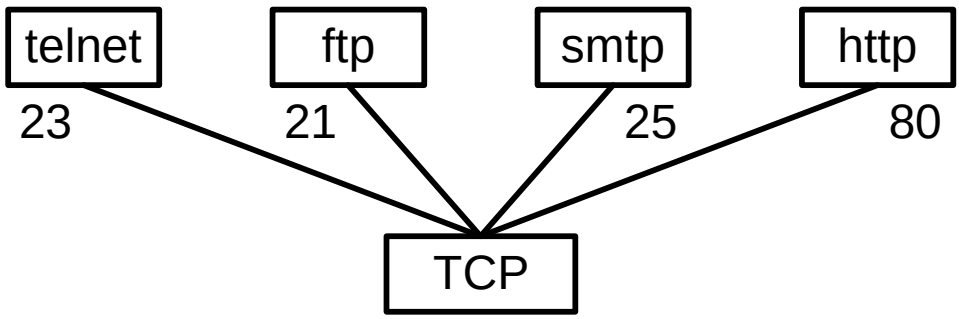
5 *Transmission Control Protocol* (TCP)

a) Principe

Le service offert par cette couche est **orienté connexion.**
Tout échange va donc commencer par une phase d'établissement.
L'unité d'échange est appelé **segment**
Le flux de données sécurisé dispose donc de fonctions de contrôles d'erreur et de flux (numéro de séquence, accusé de réception,...)

b) Port

Ce champ identifie le protocole de couche supérieure :



Des valeurs normalisées *(Well Known Ports)* comprises entre 0 et 1023 donnent accès aux **serveurs** :

- 23 → telnet
- 21 → ftp
- 25 → smtp
- 80 → http

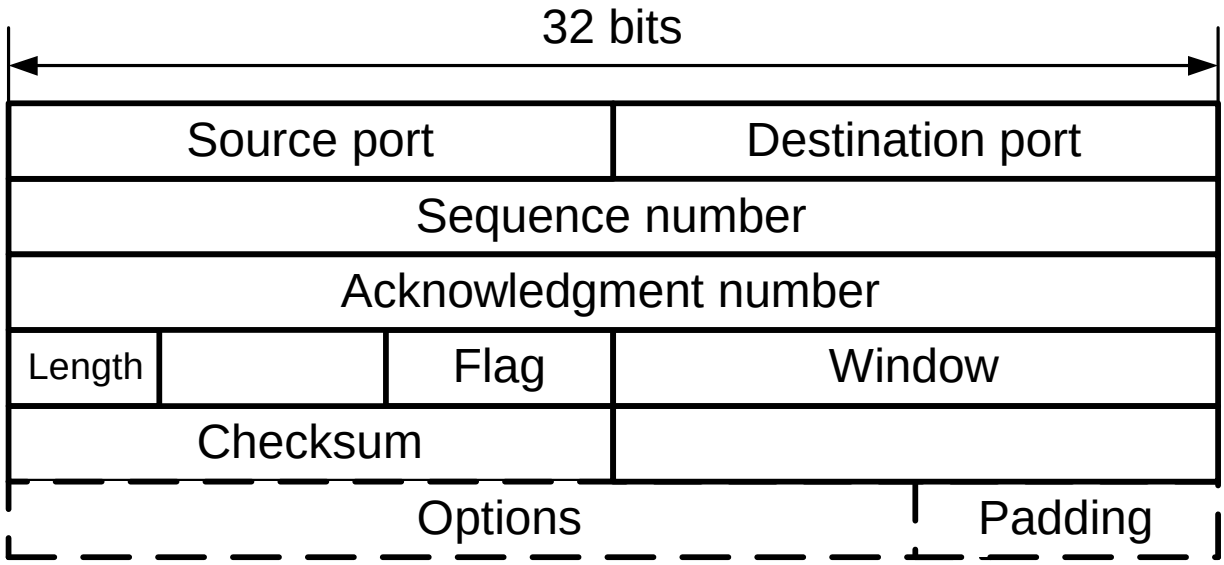
Le **client** attribue dynamiquement les numéros de port dans l'espace 49152 et 65535 *(Dynamic Ports)*

Autre espace défini : *Registered Ports* = 1024 .. 49151

La norme définit le ***socket* = adresse IP + port.**

c) Principaux champs de l'en-tête TCP

La longueur habituelle (sans option) est de 20 octets :



Valeurs du champ *flag* :

- SYN synchronise les numéros de séquence lors de l'établissement
- FIN libération
- PSH (*push*) envoi des données utiles
- ACK accusé de réception
- RST *reset* de la connexion

d) Etablissement

→ SYN demande d'établissement
← SYN, ACK demande acceptée
→ ACK, data

Protocole orienté connexion

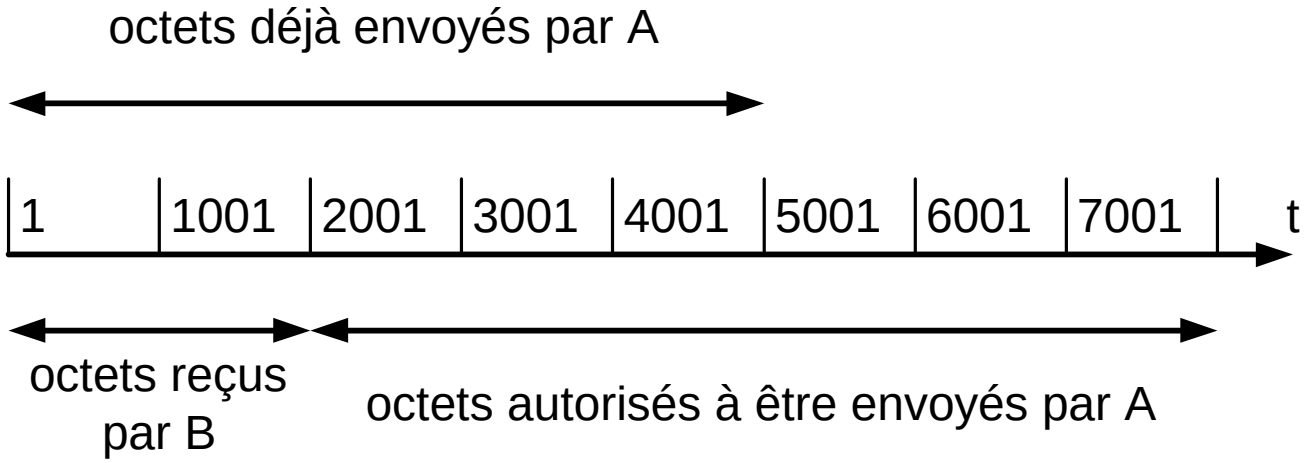
e) Flux de données

Il est orienté octet (*byte stream service*).

Ainsi le numéro de séquence (*sequence number*) marque la position du premier octet du segment envoyé.

De même le champ *Window*, nécessaire à la gestion du contrôle de flux, indique le nombre d'octets que la station peut recevoir :

→	PSH	seq=1	ack=1	win=2000
→	PSH	seq=1001	ack=1	win=2000
→	PSH	seq=2001	ack=1	win=2000
→	PSH	seq=3001	ack=1	win=2000
←	ACK		ack=2000	win=6000
→	PSH	seq=4001	ack=1	win=2000



f) Temporisateurs

Ce protocole orienté connexion est pourvu de temporisateurs :

- Acquittements retardés (superposition aux données)
- *Timer* de retransmission (contrôle de l'acquittement)
- *Timer* persistant (si la fenêtre est fermée)
- ...

g) Analyse de protocole TCP entre AST2 et Alpha :

```
Transmission Control Protocol
    Port 1029 ---> Telnet
    Sequence Number: 17279538
    Acknowledgement Number: 4126887814
    Header Length(MSB 4 bits): 5 (32-bit word)
    Reserved(LSB 4 bits): 0
    Code: 0x18
        RES: 00.. .... = Reserved
        URG: ..0. .... = Urgent Pointer is Invalid
        ACK: ...1 .... = Acknowledgement Field is Valid
        PSH: .... 1... = Push Requested
        RST: .... .0.. = No reset Connection
        SYN: .... ..0. = No synchronize Sequence Number
        FIN: .... ...0 = More Data From Sender
    Window: 8667
    Checksum: 0x0CCA
    Urgent Pointer: 0x0000

Transmission Control Protocol
    Port Telnet ---> 1029
    Sequence Number: 4126887814
    Acknowledgement Number: 17279541
    Header Length(MSB 4 bits): 5 (32-bit word)
    Reserved(LSB 4 bits): 0
    Code: 0x18
        RES: 00.. .... = Reserved
        URG: ..0. .... = Urgent Pointer is Invalid
        ACK: ...1 .... = Acknowledgement Field is Valid
        PSH: .... 1... = Push Requested
        RST: .... .0.. = No reset Connection
        SYN: .... ..0. = No synchronize Sequence Number
        FIN: .... ...0 = More Data From Sender
    Window: 6141
    Checksum: 0x16B1
    Urgent Pointer: 0x0000
```

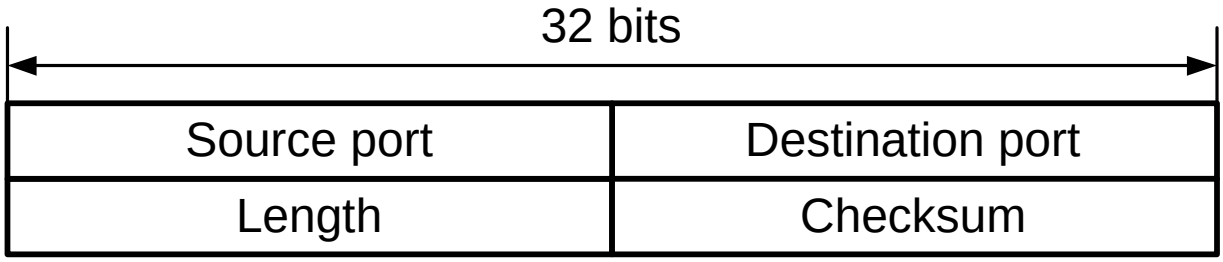
6 *User Datagram Protocol* (UDP)

UDP est un protocole simple orienté **datagramme.**

Il fait ainsi partie des protocoles **sans connexion.**

A l'inverse de TCP, il n'offre aucune garantie de fiabilité (comme IP).

Principaux champs de l'en-tête UDP :



Par souci de commodité, UDP utilise les mêmes valeurs de ports que TCP.

Analyse de protocole UDP entre AST2 et Alpha :

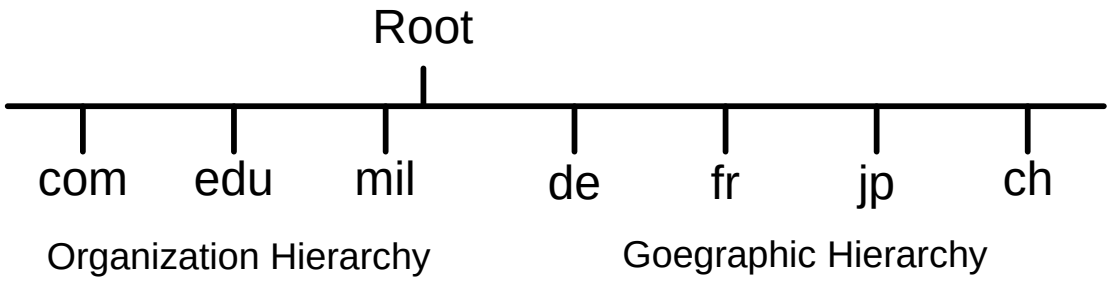
```
User Datagram Protocol
      Port 1046 ---> Domain Name Server
      Total length: 38 (Octets)
      Checksum: 0xD2BC

User Datagram Protocol
      Port Domain Name Server ---> 1046
      Total length: 218 (Octets)
      Checksum: 0x3D6E
```

7 *Domain Name System* (DNS)

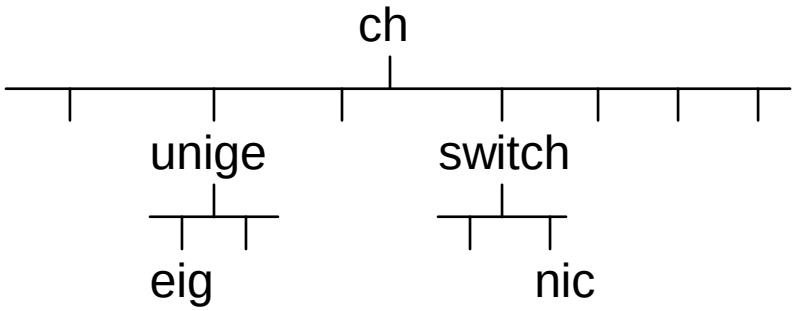
L'utilisateur du réseau *internet* préfère souvent utiliser une adresse facilement mémorisable comme http://**www.cern.ch** ou ping **nic.switch.ch** plutôt qu'une adresse IP comme 130.59.1.40

On parle alors de ***Full Qualified Domain Name*** *(FQDN)*
A l'origine, un seul fichier HOSTS.TXT gérait l'équivalence entre une adresse FQDN et une adresse IP.

Le *Domain name system* est une **base de données distribuée.**



Structure arborescente : eig.unige.ch
(host=eig domain=unige.ch)



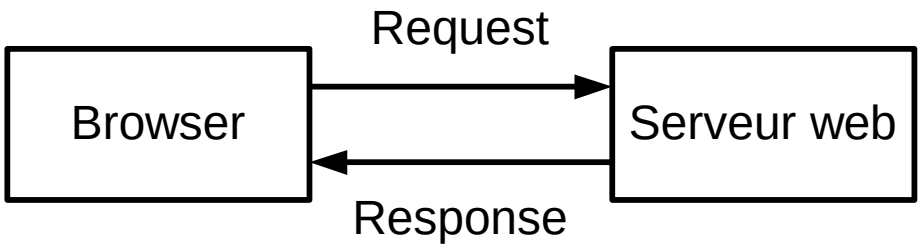
Possibilité de déléguer l'autorité : Université gère unige.ch
labo de TD gère td.unige.ch

L'échange est du type **client** *(resolver)* - **serveur** *(name server).*
Lien entre père et fils par leur adresse ip et leur domaine.
Mémoire cache dans chaque serveur.

8 Analyse de protocoles - application web

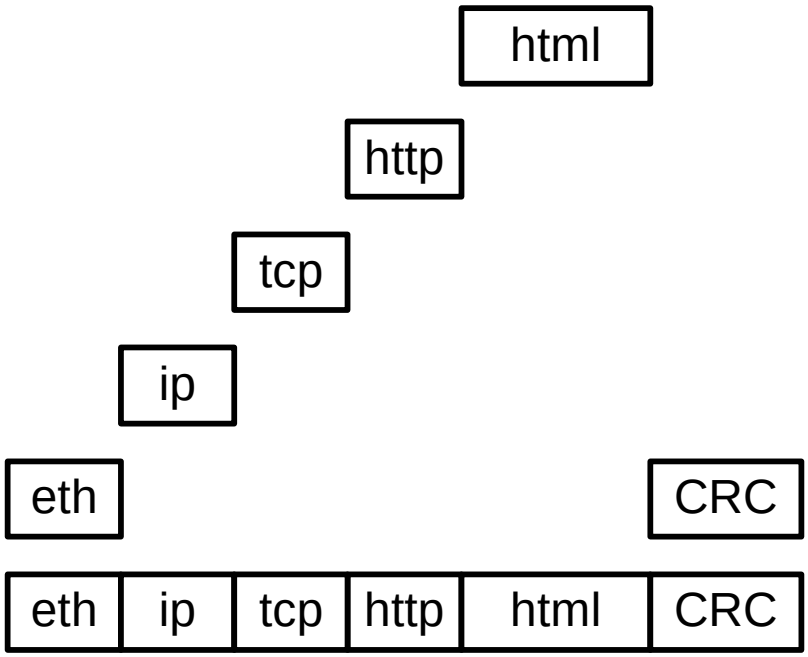
Relation de type **client** – **serveur**

Protocole HTTP *(HyperText Transfer Protocol)* utilisé entre navigateur *(browser)* et serveur *web* :



→ Get http://www.td.unige.ch/default.html
← Response

Protocol stack

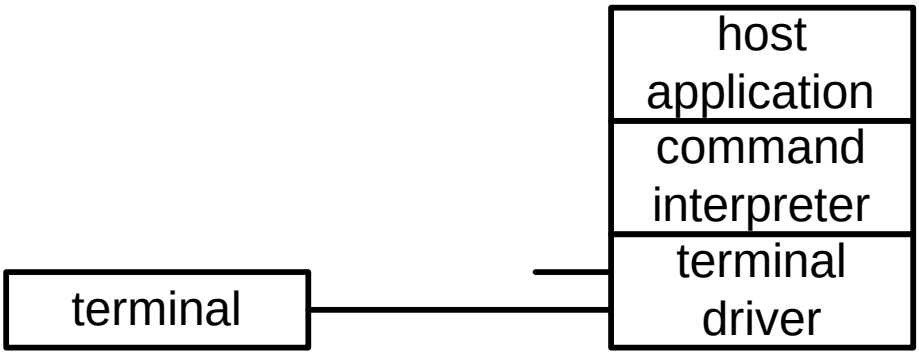


Ex 4 Donner une représentation synthétique (couches) des paquets 4 et 5 en précisant la valeur des principaux champs

9 *Telnet*

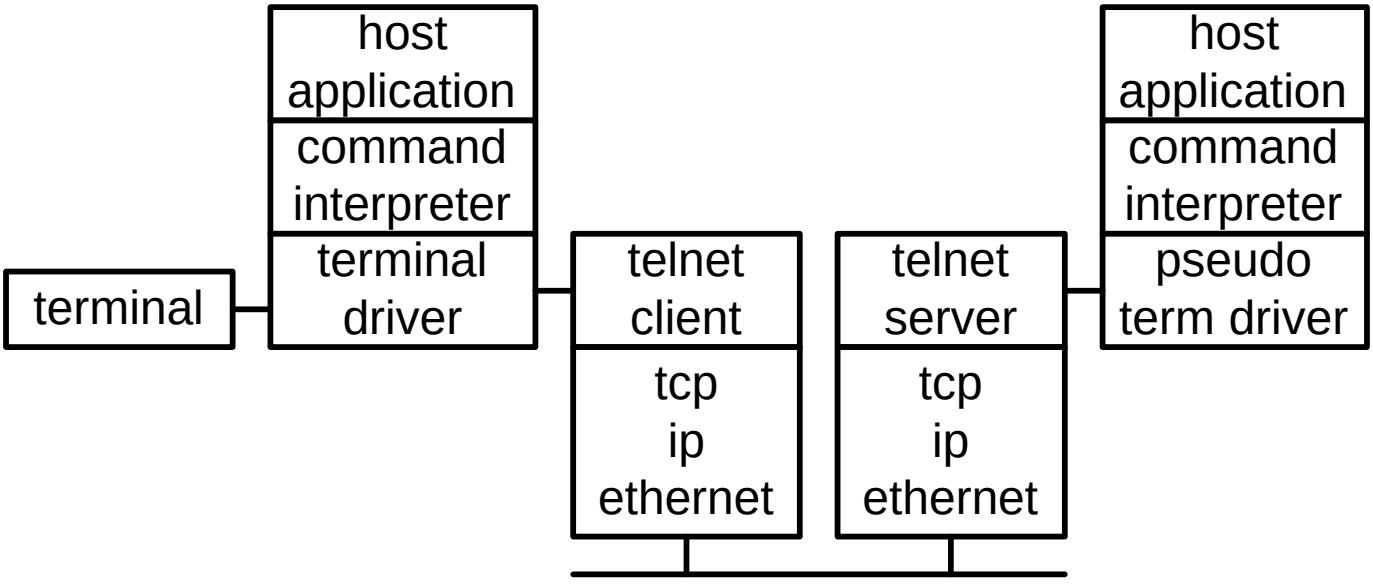
a) Architecture Unix

L'utilisateur sur son terminal, disposant d'un compte utilisateur, peut accéder au serveur par le service *remote login* :



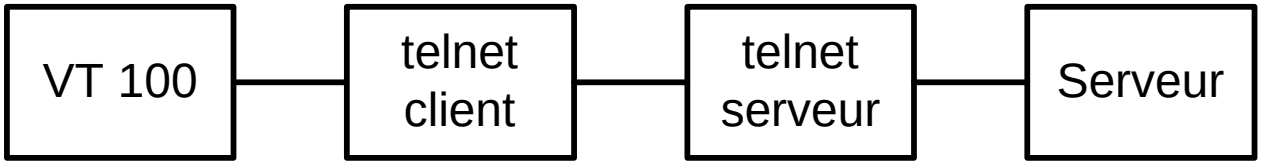
La procédure de liaison terminal - serveur, souvent asynchrone pour des raisons de coût, supporte des terminaux physiques tels VT-100.

b) Telnet dans l'architecture Unix



Côté utilisateur (*client side*) : telnet
Côté serveur (*server side*) : **telnetd** (*daemon* sur port 23)

Telnet a été conçu pour fonctionner entre n'importe quel serveur (c'est-à-dire tout système d'exploitation) et n'importe quel terminal.
Sa spécification définit le plus petit terminal commun, appelé *network virtual terminal* (NVT).
Le NVT est un terminal imaginaire, à partir duquel les deux extrémités convertissent leurs terminaux réels.



c) Commandes *telnet*

Les données utiles sont codées en ASCII.
L'octet 0xff (255) est appelé **IAC** (*interpret as command*).
L'octet suivant correspond à l'octet de commande.
Pour envoyer l'octet de donnée 255, deux octets consécutifs sont émis.

Principales commandes :

- IAC 255 octet de donnée 255
- DONT 254 négociation d'option
- DO 253 négociation d'option
- WONT 252 négociation d'option
- WILL 251 négociation d'option

Principales options :

- echo 1 RFC 857
- suppress go ahead 3 RFC 858
- status 5 RFC 859
- terminal type 24 RFC 1091
- window size 31 RFC 1073
- terminal speed 32 RFC 1079
- remote flow control 33 RFC 1372

d) Négociation de l'option

→ WILL A veut valider l'option
← DO B dit oui

→ WILL A veut valider l'option
← DONT B dit non

→ DO A veut que B valide l'option
← WILL B dit oui

→ DO A veut que B valide l'option
← WONT B dit non

→ WONT A veut invalider l'option
← DONT B doit dire oui

→ DONT A veut que B invalide l'option
← WONT B doit dire oui

10 *File Transfer Protocol* (ftp)

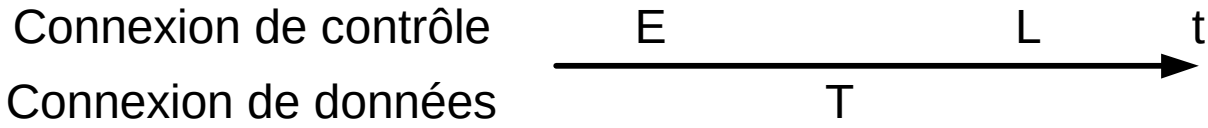
Avec cette application analogue à Kermit, l'utilisateur doit communiquer tantôt avec l'ordinateur **distant** ; tantôt avec l'ordinateur **local**.

Pour un transfert de **A → B :**	**A (local)**	**B (distant)**
		receive
	send	

Le protocole ftp simplifie la procédure en pilotant (put - get) de façon transparente l'ordinateur distant.

Pour un transfert de **A → B :**	**A (local)**	**B (distant)**
	put	

Il utilise en fait **2 connexions TCP :**



Le RNIS (et ATM) font un large usage de ce concept de **données et signalisation séparées !**

Types de fichier supportés : ASCII - binaire

L'utilisateur dispose de cette application depuis :
- une fenêtre terminal en mode commande (fenêtre DOS),
- une interface graphique spécifique,
- son navigateur habituel.

11 Principales RFCs

Protocole	RFC
ARP	826
DHCP	1531
DNS	1034, 1035
ICMP	792
FTP	959
IP	791, 919, 922, 950
IP over ethernet	894
IP over FR	1294
IP over X.25,ISDN	
PPP	1331, 1332
RIP	1058
SLIP	1055
SMTP	821, 822
SNMP	1441-1452
TCP	793
TELNET	854, 855
TFTP	1350
UDP	768
Assigned numbers	1700