M1 Informatique, Réseaux
Découverte des réseaux centrée sur l'internet

Olivier Togni
Université de Bourgogne, IEM/LE2I
Bureau G206
olivier.togni@u-bourgogne.fr

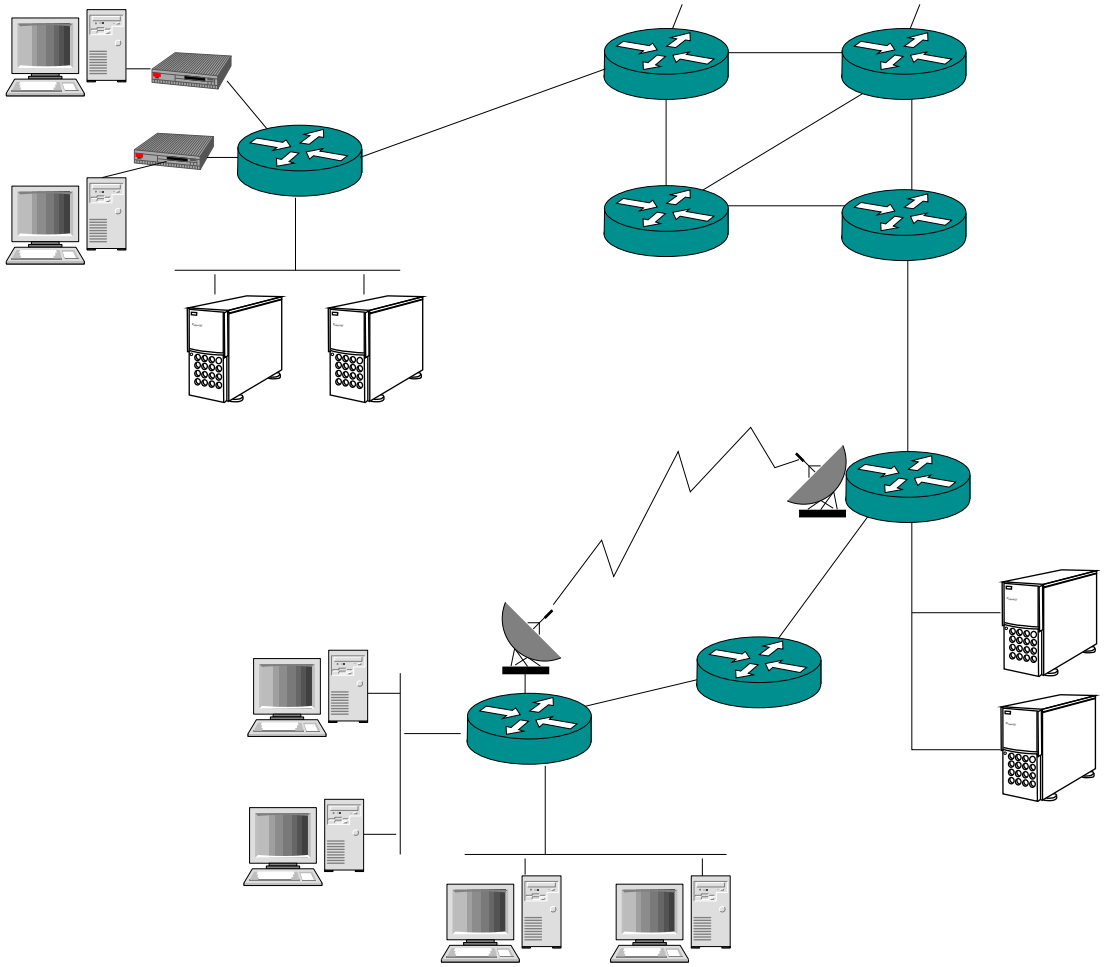
4 septembre 2023

Chapitre 1 : Les réseaux et l'internet : une introduction

1. introduction, historique
2. protocole
3. classifications des réseaux
4. réseaux d'accès, FAI
5. modèle en couche et architecture de l'internet

L'internet ?

Concrètement : réseau d'ordinateurs à l'échelle planétaire



L'internet ?

Fonctionellement : applications distribuées permettant aux terminaux d'échanger des données

- mél,
- naviguation web,
- transfert de données multimédia en flux continu (streaming),
- téléhponie sur IP,
- jeux en réseau,
- partage d'égal à égal,
- ...

Rem : toile (web) n'est pas un réseau à part mais une des nombreuses applications utilisant les services de l'internet

Historique

- 1961-1972 commutation par paquet (MIT, ARPA)
  → 1972 : ARPAnet a 15 nœuds
- 1972-1980 réseaux propriétaires et interfonctionnement de réseaux (ALOHAnet, Telnet, SNA, Cyclades, ...)
- 1980-1990 prolifération des réseaux (BITNET, CSNET, NSFNET,...)
  → 1983 : TCP/IP remplace NCP dans ARPAnet
- 1990- explosion de l'internet
  → Tim Berners-Lee conçoit le web (1989-1991)

Rem : et la France ? réseau commuté (X.25, Frame Relay)
Transpac, Minitel

Définition

Définition

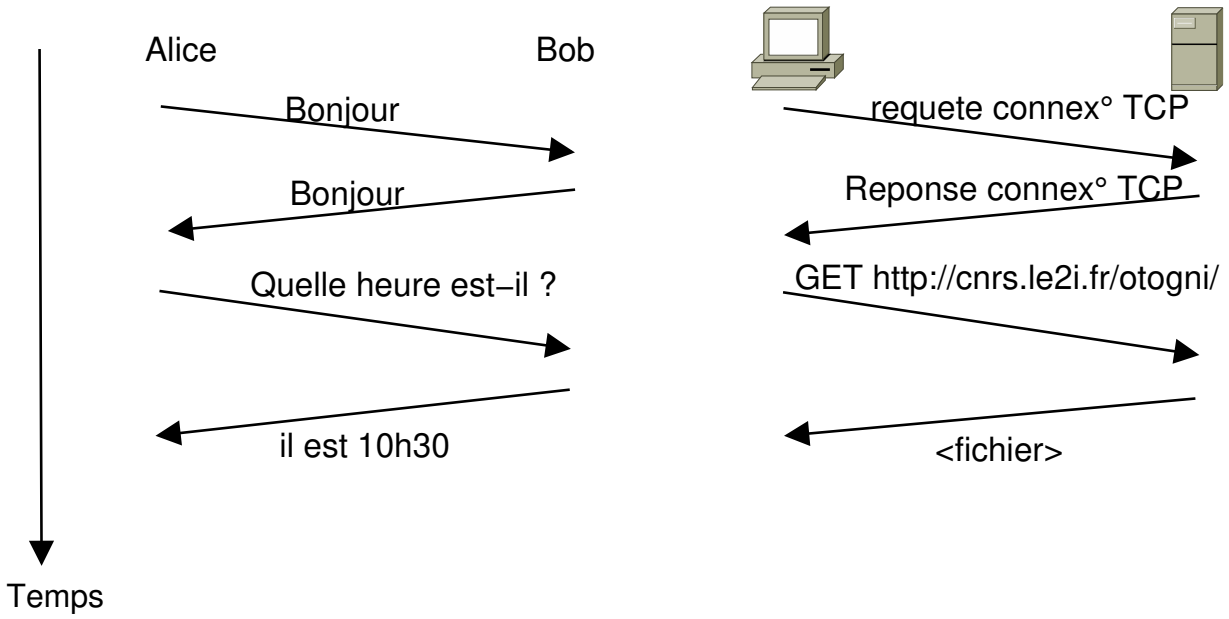
Un protocole définit le format et l'ordre des messages échangés entre deux entités ou plus ainsi que les actions générées au moment de la transmission ou réception d'un message ou autre événement.

Rem : peut être décrit par un automate

IETF (Internet Engeenering Task force) : organisme dont le rôle est de faire évoluer l'Internet (création, expérimentation, applications des normes de l'internet) $\rightarrow$ documents RFC (Request For Comment), actuellement plus de 6000.

Protocoles principaux de l'internet : TCP et IP

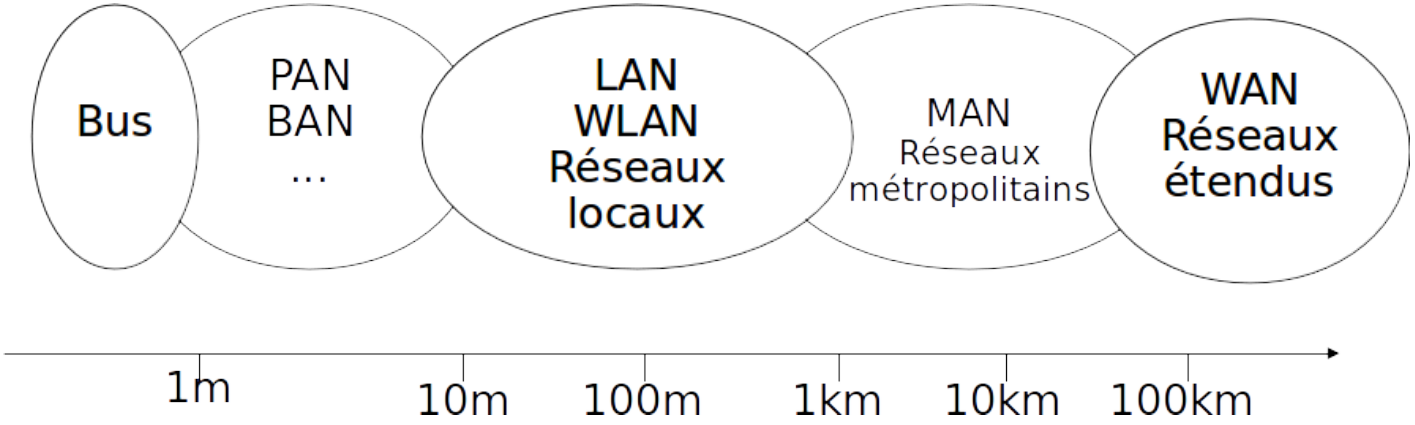
Exemple d'échange



Classifications des réseaux

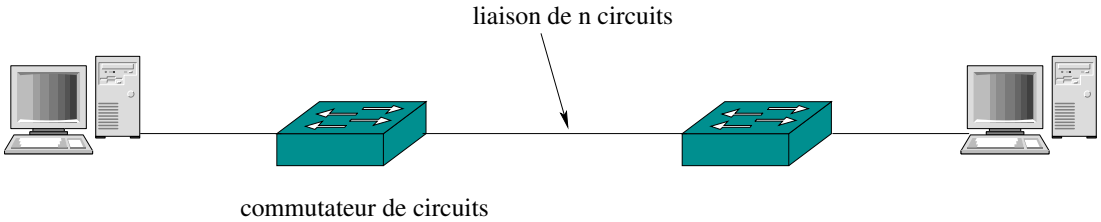
Différentes classifications suivant :

- la taille du réseau (étendue),
- le mode de fonctionnement (commutation de circuits / par paquet),
- le mode de transmission (connecté / non connecté).



Commutation de circuits

Pour le cœur de réseau
Ressources le long d'un chemin pour la communication entre les terminaux sont réservées pour toute la durée de la session
Plusieurs circuits par liaison par multiplexage en fréquence (FDM) ou temporel (TDM)

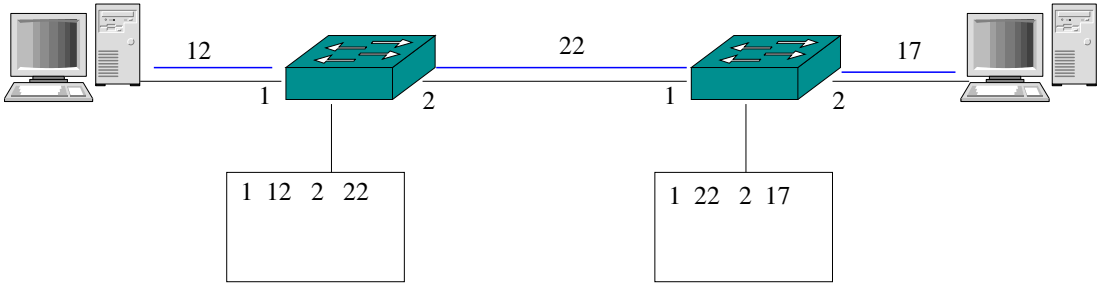


– couteux en ressources (si silence)
– mise en place prend du temps

Circuits virtuels

VC = virtual circuit constitué de

- chemin source / destination,
- n° de VC pour chaque liaison,
- entrée dans table de conversion de VC dans chaque commutateur.



⇒ Chaque commut. garde une trace (état) des connexions le traversant

Commutation par paquet

- Ressources utilisées à la demande (pas de réservation) ⇒ parfois attente pour accès liaison
- Données fragmentées en paquets par l'émetteur. Chaque paquet contient l'adresse de destination (structure hiérarchique).
- Les commutateurs (routeurs) ne gardent pas trace des flux les traversant.
- Routeur = plusieurs interfaces avec chacune un tampon (file d'attente) en sortie ⇒ délai d'attente suivant encombrement + perte possible si tampon saturé

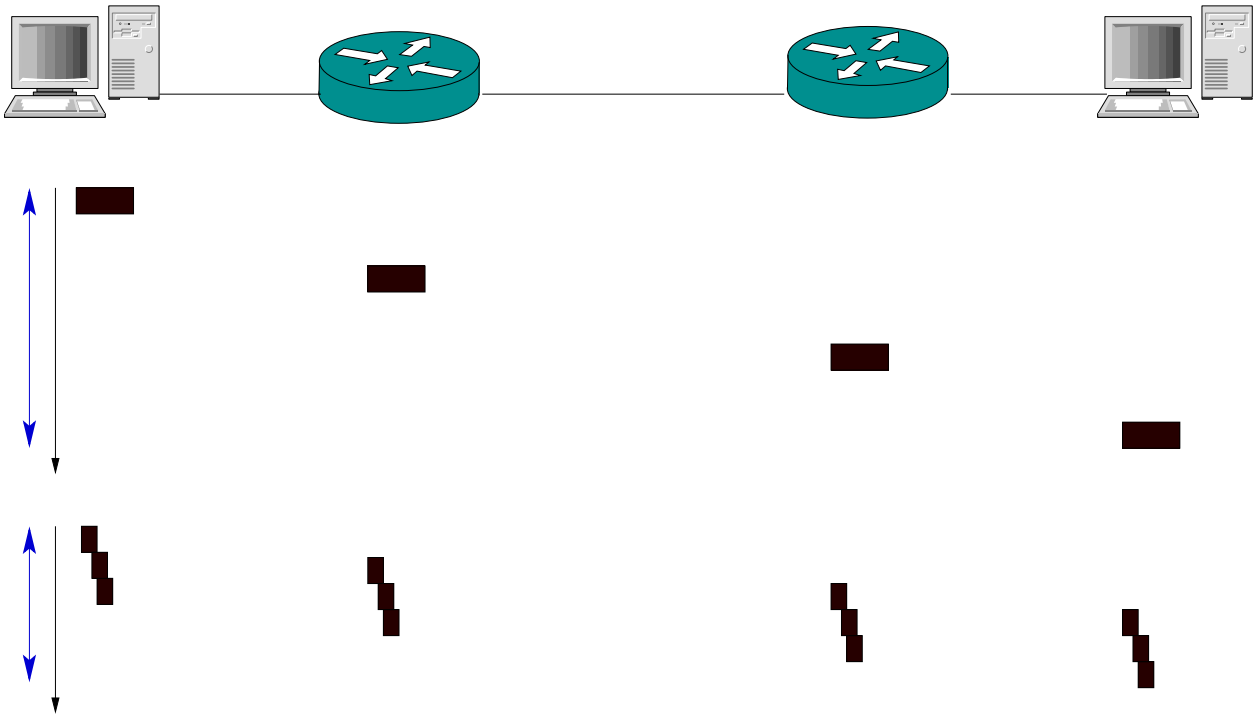
− mal adapté aux services en temps réél
+ meilleur partage, simplicité

Mode connecté / non connecté

- Mode connecté : établissement d'un connexion entre les entités par échange de messages de contrôle (procédure de "poignée de main") avant le transfert des paquets ⇒
  - permet de se préparer à l'envoi ou la réception de paquets
  - associé au service de transfert fiable et contrôle de flux, congestion
  - ex : TCP
- Mode non connecté : envoi de paquet entre entités sans procédure de mise en présence préalable
  - permet envoi plus rapide des données
  - pas de fiabilité : la source ne sait pas quels paquets sont arrivés à destination
  - ex : UDP

Segmentation de messages

Dans un réseau à commutation par paquet, il est préférable que les messages soient segmentés en paquets (plus petits) par la source et reconstitués par le destinataire



Retards et pertes

- temps de traitement = temps pour examiner en-tête et déterminer liaison de sortie $\rightarrow \mu s$ pour les routeurs haut débit
- temps d'attente (en file attente sortie) $\mu s \rightarrow ms$
- temps de transmission = temps pour placer tous les bits sur la liaison = L/R où L = taille du paquet (bits) et R = débit (b/s)
- temps de propagation : dépend du support = distance/vitesse, avec vitesse entre $2.10^8 m/s$ et $3.10^8 m/s$

Exemple de traceroute

```
traceroute to www.cbs.com (64.30.228.49), 30 hops max, 60 byte packets
 1  swr-r235.u-bourgogne.fr (193.52.235.97)  0.921 ms  0.919 ms  1.156 ms
 2  vl193-te0-0-0-0-dijon-rtr-011.noc.renater.fr (193.51.181.218)  3.353 ms  3.355 ms  3.351 ms
 3  te0-3-2-0-lyon1-rtr-001.noc.renater.fr (193.51.177.72)  9.231 ms  9.235 ms  9.230 ms
 4  * * *
 5  xe-4-3-0-100.mrs10.ip4.gtt.net (77.67.90.117)  16.950 ms  16.955 ms  16.951 ms
 6  xe-9-2-4.par22.ip4.gtt.net (89.149.187.166)  19.311 ms  19.268 ms  19.266 ms
 7  as3356.par22.ip4.gtt.net (141.136.103.182)  19.243 ms  19.240 ms  19.249 ms
 8  ae-1-8.bar1.Phoenix1.Level3.net (4.69.133.29)  163.782 ms  164.497 ms  164.135 ms
 9  ae-1-8.bar1.Phoenix1.Level3.net (4.69.133.29)  164.878 ms  163.348 ms  163.977 ms
10  CBS-CORPORA.bar1.Phoenix1.Level3.net (4.53.106.166)  161.593 ms  163.541 ms  162.578 ms
11  ae2-0.io-phx2-ex8216-1.cnet.com (64.30.227.58)  163.266 ms  162.597 ms  166.940 ms
12  * * *
13  * * *
...
30  * * *
```

Réseaux d'accès

Réseau d'accès = liaison entre le terminal et un routeur périphérique

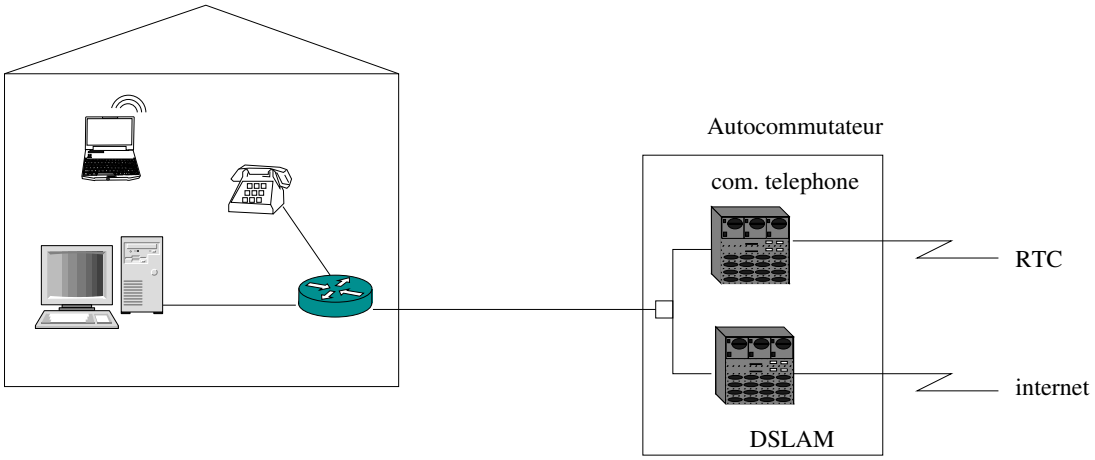
3 modes d'accès :

- résidentiel
  - ADSL,
  - fibre optique + cable coaxial (cable)
- d'entreprise : LAN (le plus souvent Ethernet sur paire torsadée cuivre)
- mobile : WLAN (Wifi, WiMAX, ...)

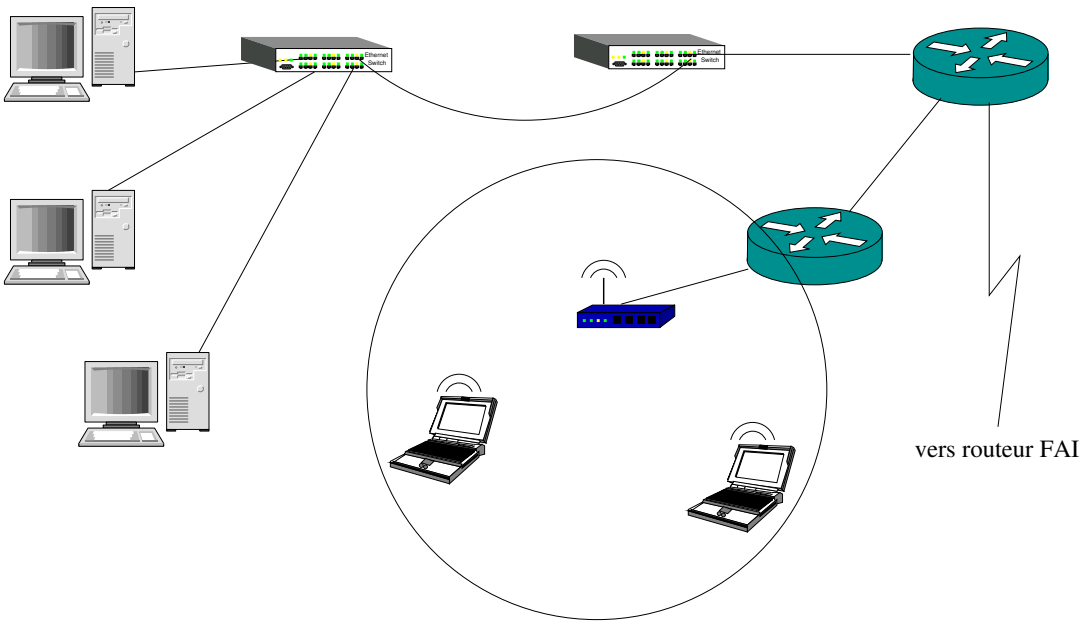
ADSL

Asymetric Digital Suscriber Line fait partie des normes xDSL

Modem ADSL = Multimodulateur sur 255 porteuses (4,3125 kHz, 8,625 kHz, 12,9375 kHz, ...)
Canaux 16-31 : traffic montant
Canaux 33-255 : traffic descendant



LAN et WLAN

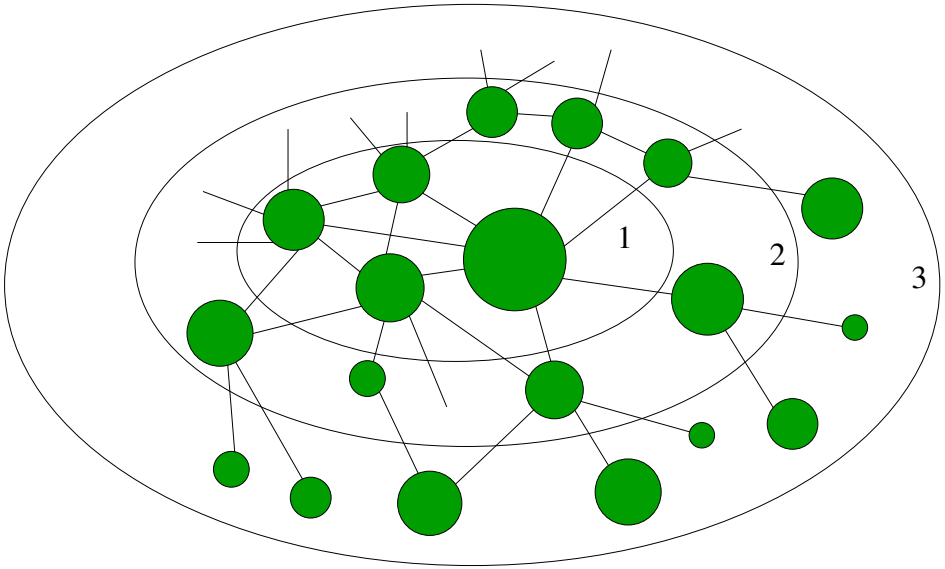


Fournisseurs d'accès internet

Plusieurs niveaux de FAI :

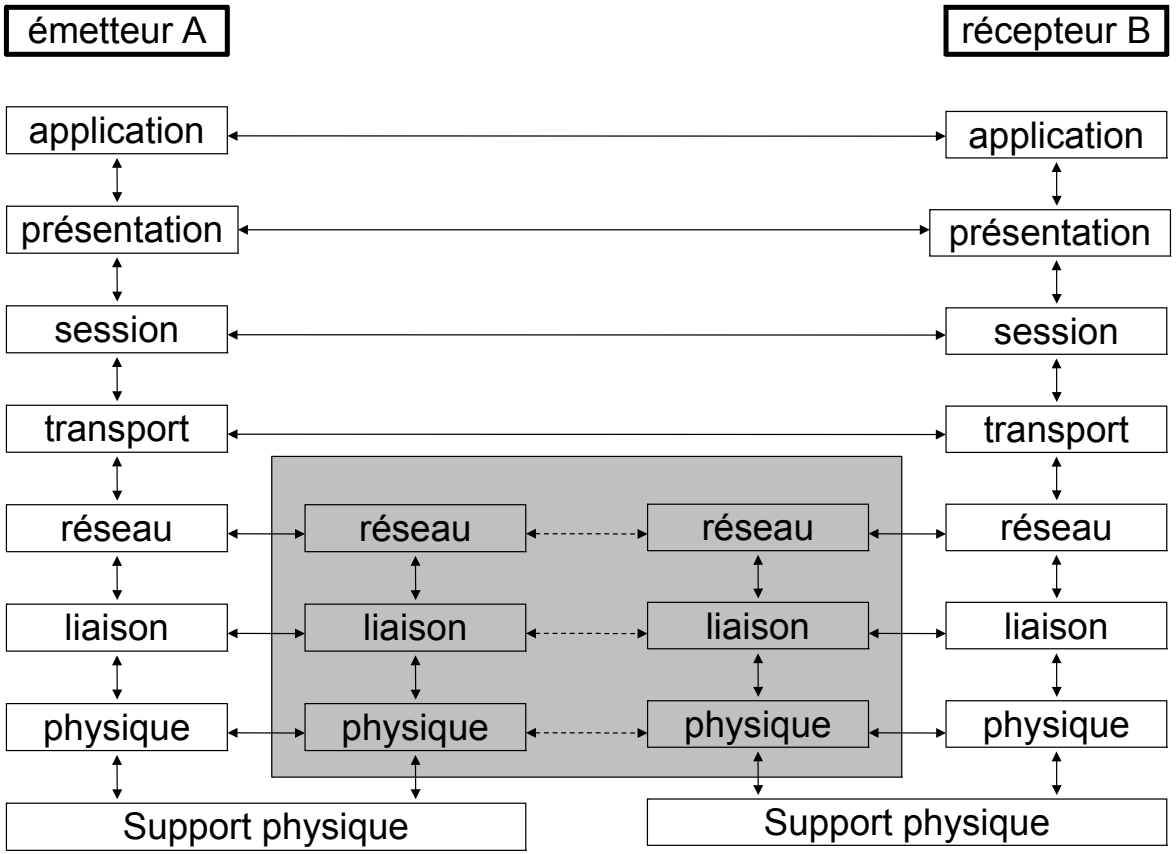
- niveau 1 : couverture internationale (ex . UUNet)
- niveau 2 : couverture nationale

Connectés entre eux par des points de présence (POP), points d'accès au réseau (NAP) ou noeuds d'échange privés



Structure en couches

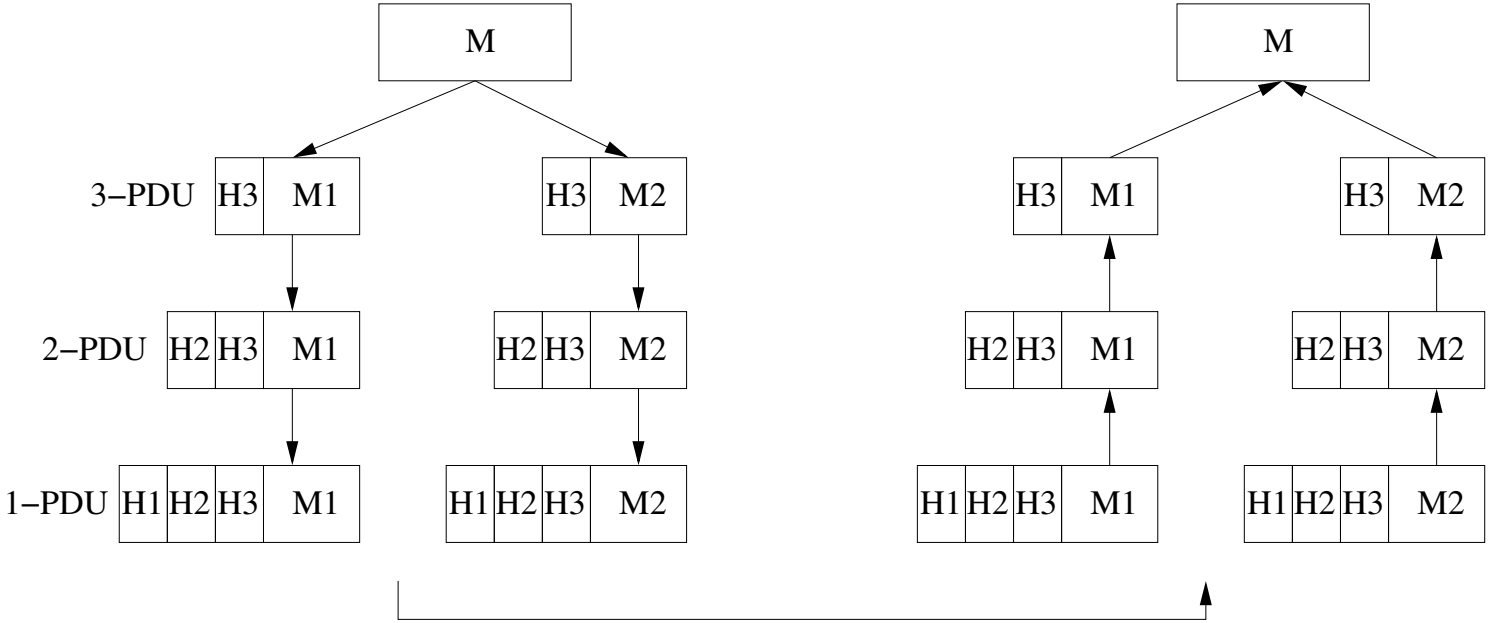
Modèle en couche OSI normalisé par l'ISO en 1997 (ISO 7498)



Communication entre couches

Encapsulation

n-PDU = Packet Data Unit de la couche *n*



Fonctions des couches

- Détection d'erreurs → améliorer fiabilité
- contrôle de flux → éviter saturation d'un poste plus lent avec des PDU
- segmentation et réassemblage → diviser les groupes de données volumineux coté émetteur et les reconstituer coté récepteur
- multiplexage → plusieurs session de niveau supérieur peuvent partager une même connexion de niveau inférieur
- établissement de connexion → procédure de mise en présence avec d'autres postes

Architecture de l'internet

Architecture = couches + protocoles

7 Application
6 Présentation
5 Session
4 Transport
3 Réseau
2 Liaison
1 Physique

Modèle OSI

Application: *SSH, HTTP, Telnet, DNS, ...*	
Transport: *TCP, UDP*	
Inter–réseau: *IP*	*ICMP* *ARP*
Accès au réseau: *Ethernet, TokenRing, FDDI, PPP,...*	

Architecture TCP/IP

Les couches de l'internet 1/2

- Couche Application
  - Responsable de l'exécution des différentes applications réseau
  - Multitude de protocoles (HTTP pour le web, SMTP pour la messagerie électronique, FTP pour le transfert de fichiers, ...)
  - Facile de créer des protocoles personnels de couche application
- Couche Transport
  - transport des messages de la couche Application
  - TCP : service orienté connexion, garantie livraison et contrôle de flux et de saturation
  - UDP : service sans connexion (service minimum)

Les couches de l'internet 2/2

1. Couche Réseau
   - acheminement des datagrammes (IP)
   - routage : plusieurs protocoles intra ou inter domaine
   - gestion des erreurs : ICMP
2. Couche Liaison
   - transmission des paquets IP d'un noœud au nœud suivant
   - service offert dépend du protocole utilisé (Ethernet, PPP, ATM, ...)
3. Couche Physique
   - transmettre suite de bits correspondant à la trame du noœud au nœud suivant
   - dépend du protocole et du support physique

Chapitre 2 : La couche Application

1. Principes des applications réseaux
2. Le web et HTTP
3. Le DNS
4. le courriel

Les éléments de la couche Application

- les processus sur différents terminaux communiquent en s'envoyant des messages
- les applications réseau utilisent des protocoles de niveau application
- l'agent utilisateur est interface entre l'utilisateur et l'application

Ex : le web =

- norme sur le format des documents (HTML)
- navigateurs web (IE, Firefox, chrome, ...)
- serveurs web (Apache, Microsoft, ...)
- protocole HTTP (RFC 2616)

Protocoles de niveau application

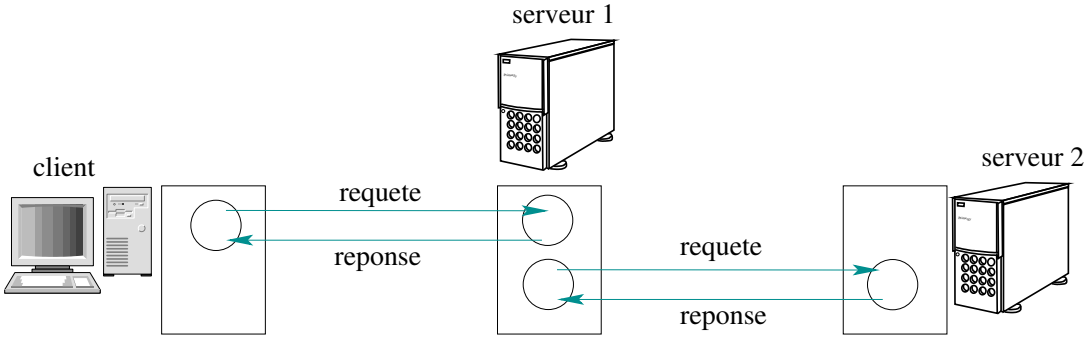
Définissent la façon dont les processus situés sur des systèmes d'exploitation différents échangent mutuellement des messages :

- type de messages échangés (requête, réponse, ...)
- syntaxe (différents champs et délimitation)
- sémantique (sens des informations contenues dans les champs)
- règles utilisées pour savoir quand et comment un processus doit envoyer (ou répondre à) un message

Rem : de nombreux protocoles sont publics (définis par des RFC), d'autres sont du domaine privé (ex. skype)

Architecture client-serveur

- Serveurs : toujours actifs, adresse IP permanante
- clients : pas toujours connectés, adresse IP dynamique, ne communiquent pas directement entre eux



Processus identifiés par couple (adresse IP, n° de port)

Architecture pair à pair

- les pairs communiquent directement entre eux
- chaque pair demande des services et en propose en retour
- serveurs pas toujours actifs, adresse IP changent → gestion difficile
- auto-scalable : les nouveaux pairs apportent de nouveaux services et de nouvelles demandes
- ex. : BitTorrent

Services nécessaires à une application

Application	Perte de données	Débit	Sensibilité au temps
Transfert de fichier	Interdite	Flexible	non
Courrier électronique	Interdite	Flexible	non
Pages web	Interdite	Flexible	non
Fichier av en temps réel	Acceptable	qques kb/s à 10 Mb/s	oui : qques centaine de ms
Fichier av enregistré	Acceptable	qques kb/s à 10 Mb/s	oui : qques s
Jeux interactifs	Acceptable	qques kb/s	oui : qques centaines de ms
Messagerie instantanée	Interdite	Flexible	oui et non

Services fournis par TCP et UDP

TCP : service orienté connexion et transport fiable
UDP : service sans connexion et non fiable

Application	Protocole applicatif	protocole de transport
Courier électronique	SMTP	TCP
Accès terminal distant	Telnet	TCP
Web	HTTP	TCP
Transfert de fichiers	FTP	TCP
serveur de fichier distant	NFS	UDP ou TCP
Multimédia en streaming	souvent propriétaire	UDP ou TCP
Téléphonie sur internet	souvent propriétaire	généralement UDP

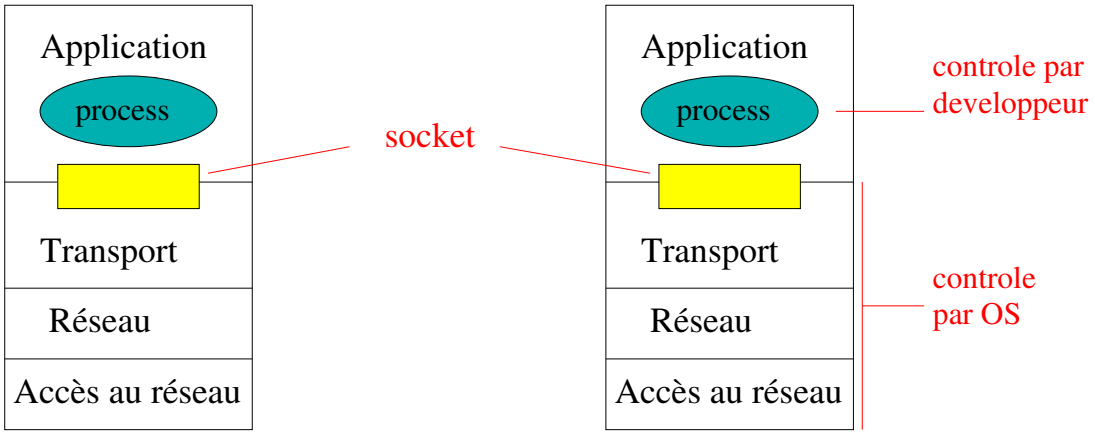
Rem : d'autres protocoles de transport existent, par ex. SCTP

Sécurisation de TCP

- TCP & UDP
  - pas cryptés
  - mots de passe traversent l'internet en clair
- SSL/ TLS (depuis 2001, IETF)
  - fournit une connexion TCP cryptée
  - intégrité des données, authentification de bout en bout
  - au niveau application : les applications utilisent une librairie SSL qui "parle" à TCP
  - ex. HTTPS = HTTP sur TLS

Sockets

Processus envoie/reçoit des messages vers/depuis ses sockets

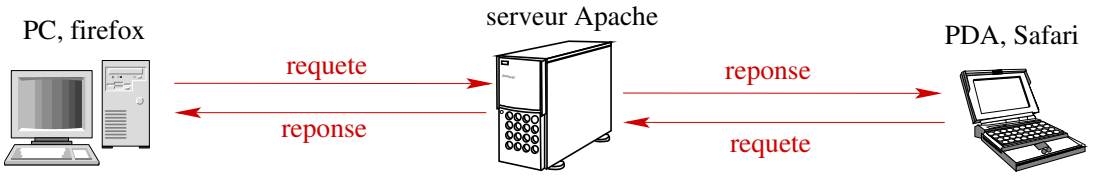


HTTP et le web

- HTTP = HyperText Transfert Protocol
  - HTTP 1.0 (→ 1996, RFC 1945)
  - HTTP 1.1 (1999, compatible avec 1.0 (RFC 2616 + RFC 7230-7237))
  - HTTP 2 (2015, RFC 7540) provient de SPDY de Google
  - HTTP 3 (2022, RFC 9114), utilise QUIC au lieu de TCP
- Page web constituée de plusieurs objets : fichier HTML, image, clip, applet, ...
- objet identifié par une URL (Uniform Ressource Locator) ex : `http://www.monserveur/monsite/photo.png`
- Client web (navigateur) demande, reçoit et affiche les pages web (pôle client de HTTP)
- Serveur web héberge les objets web (pôle serveur de HTTP) et les envoie en réponses aux requêtes

HTTP et le web (suite)

- utilise TCP (sauf HTTP3)
  - le client initie une connexion TCP auprès du serveur (port 80)
  - le serveur accepte la dmeande de connexion du client
  - messages HTTP échangés entre le navigateur (client HTTP) et le serveur Web (serveur HTTP)
  - fermeture connexion TCP
- HTTP est un protocole sans état : le serveur ne maintien pas d'information sur les requêtes précédentes du client



Connexion persistante

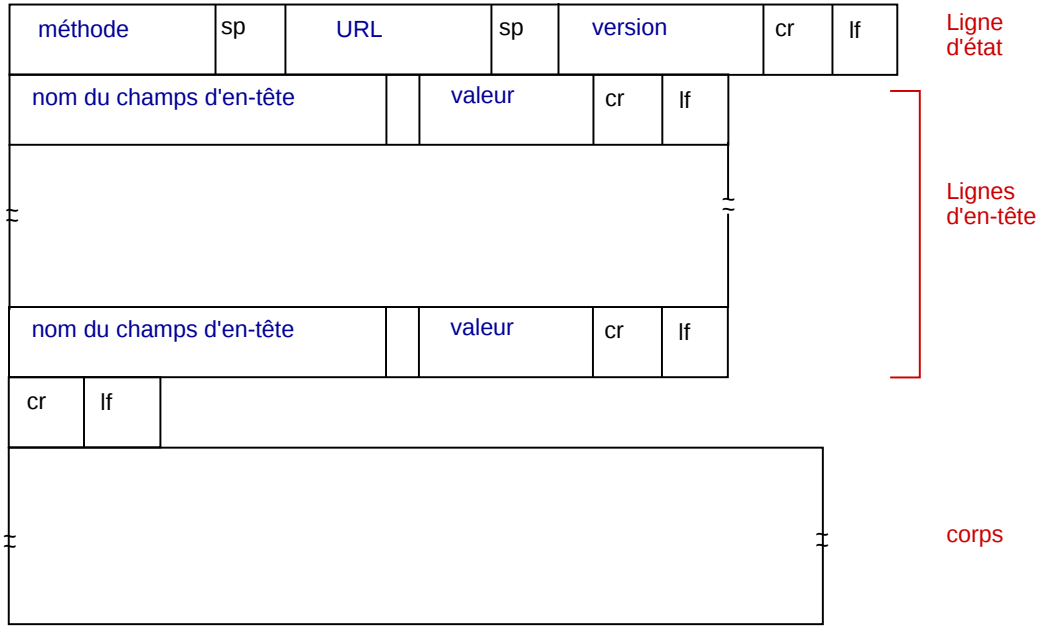
- connexion non persistante : au plus un objet envoyé sur la connexion TCP (puis fermeture)
  ex. : HTTP 1.0 utilise des connexions non persistantes
- connexion persistante : plusieurs objets peuvent être envoyés lors d'une même connexion TCP
  Connexion fermée si inactivité longue
  HTTP 1.1 les deux (par défaut persistante)
- avec pipelinage (envoi requête suivante sans attendre réponse) ou sans

$\Rightarrow$ plus rapide

Messages HTTP

- 2 types de messages : requête ou réponse
- format ASCII ⇒ lisible
- Trois requêtes les plus courantes (9 en HTTP 1.1) :
  - GET : obtenir un objet
  - HEAD : obtenir informations sur objet sans demander l'objet en lui-même
  - POST : transmettre des données pour modifier ou créer un objet

Format des requêtes HTTP




```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

Réponses HTTP

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 09 Aug 2011 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
(data data data data data ...)
```

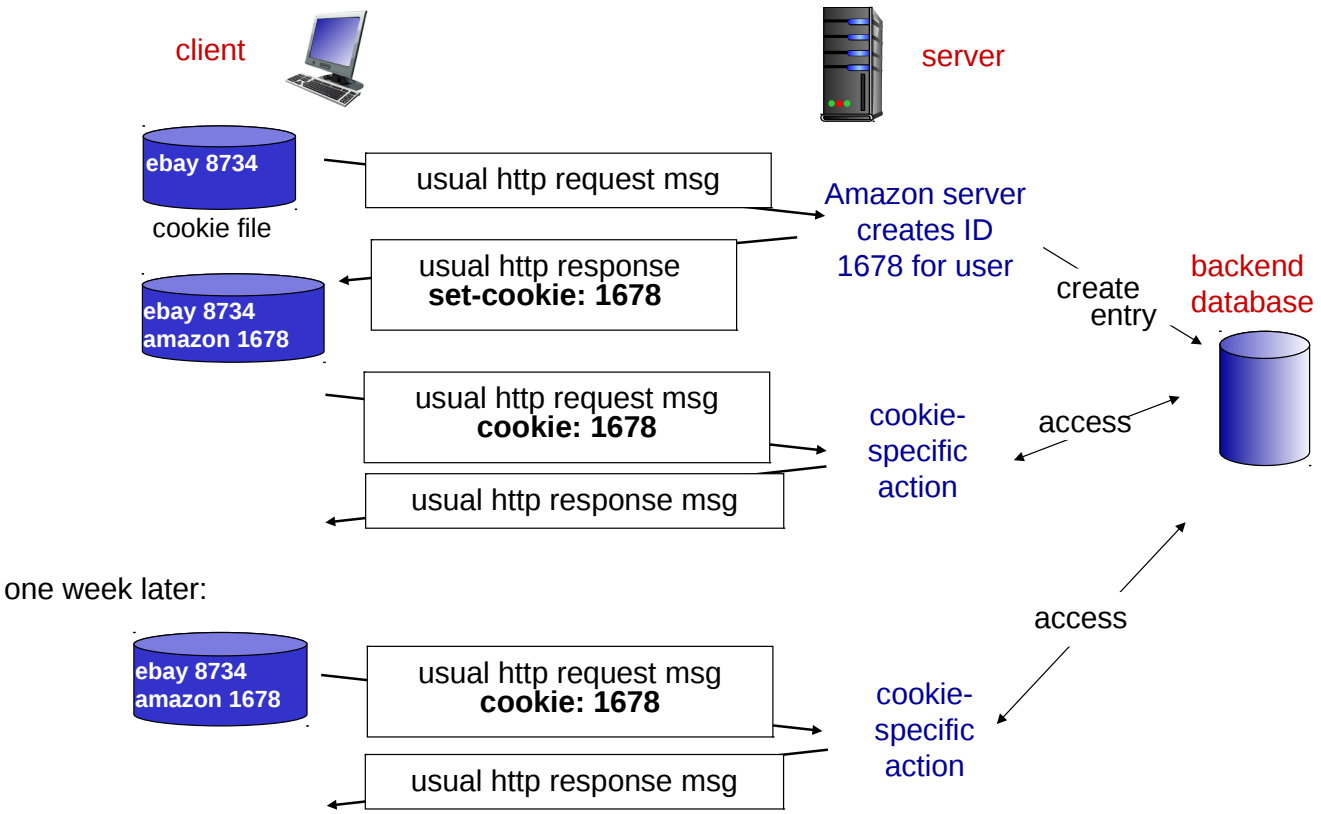
Quelques autres codes :

- 301 Moved Permanently
- 400 Bad Request
- 404 Not Found
- 505 HTTP Version Not Supported

Cookies

- Crées par Netscape pour gérer les états
- permet au serveur de mémoriser des données coté client :
  - clé de session
  - préférences de l'utilisateur sur le site
  - achats réalisés, ...
- Set-Cookie dans réponse du serveur pour demander au client de créer le cookie
- contient plusieurs champs : Nom, Valeur, Domaine, Date péremption, ...

Cookies



(de Kurose et Ross, Computer networking : a top-down approach, 6E, Pearson)

Caches

- permettent de réduire les délais de réponse
- deux types de caches :
  - cache navigateur
  - cache proxy HTTP
- repose sur GET conditionnel (If-modified-since : `<date>`)

Proxy HTTP

- utilise régulièrement des requêtes HEAD pour proposer des fichiers non périmés
- Etag : identifie un objet sur le serveur (si objet modifié ⇒ Etag modifié) utilisé avec `If-match`, `If-none-match`
- négociation de contenu : `Accept-langage:<l1>, <l2>, ...` (pondération possible)

HTTP/2 (RFC 7540) : flux et trames

- nouvel encodage binaire des messages, découpés en trames (*frames* de données ou d'en-tête)
- plusieurs flux (*streams*) bidirectionnels dans la connexion
  - identifiant du flux présent dans chaque en-tête de trame
  - les flux peuvent être priorisés
- chaque message HTTP (requête, réponse) correspond à une ou plusieurs trames
- les trames de plusieurs flux sont entrelacées et réassemblées ensuite

Compression HPACK (RFC 7541) par codage Huffman + liste indexée des en-têtes sur client et serveur

HTTP/2 : démarrage connexion

```
GET / HTTP/1.1
Host: www.example.com
Connection: Upgrade, HTTP2-Settings
Upgrade: h2c
HTTP2-Settings: [base64url encoding of HTTP/2 SETTINGS payload]
```

Si le serveur est un vieux serveur qui ne gère pas HTTP/2, il ignorera le Upgrade :

```
HTTP/1.1 200 OK
Content-Length: 243
Content-Type: text/html
```

Si le serveur est d'accord pour faire de l'HTTP/2, il répondra 101 (Switching Protocols) :

```
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: h2c
```

HTTP/2 : types de trames

Champs `type` sur 1 octet :

- DATA (type 0) : trames de données (ex. pages HTML)
- HEADERS (type 1) trame d'en-têtes HTTP (comprimés par HPACK)
- PRIORITY (type 2) : indique la priorité que l'émetteur donne au flux qui porte cette trame
- RST_STREAM (type 3) : terminaison du flux
- SETTINGS (type 4) : envoi de paramètres
- PUSH_PROMISE (type 5) : indique qu'on va transmettre des données non sollicitées (push)
- PING (type 6) : permet de tester le flux
- GOAWAY (type 7) :fermer proprement une connexion
- WINDOW_UPDATE (type 8) : faire varier la taille de la fenêtre
- CONTINUATION (type 9) : indique la suite d'une trame précédente

Identification des hôtes

- Vie réelle : humain identifiés de plusieurs façons (nom, n° sécu, n° permis conduire, ...)
- internet : hôtes identifiés de deux façons :
  - Nom d'hôtes (ex. cnn.com, ufrsciencestech.u-bourgogne.fr)
    - appréciés par les humains
    - pas ou peu d'info sur localisation
    - longueur variable
  - adresses IP (IPv4 : 32 bits, ex : 193.52.237.96)
    - hiérarchique : réseau + sous-réseau + numéro hôte
  - Translation effectuée par DNS

Domain Name System

DNS (RFC 1034, 1035 + récents)

1. BDD distribuée sur hiérarchie de serveur DNS
2. protocole de niveau application permettant aux hôtes de questionner la BDD

- souvent serveur Unix + logiciel BIND (Berkeley Internet Name Domain)
- utilise UDP sur port 53 (TCP pour recopie BDD entre serveurs)
- point critique de l'architecture TCP/IP ! !

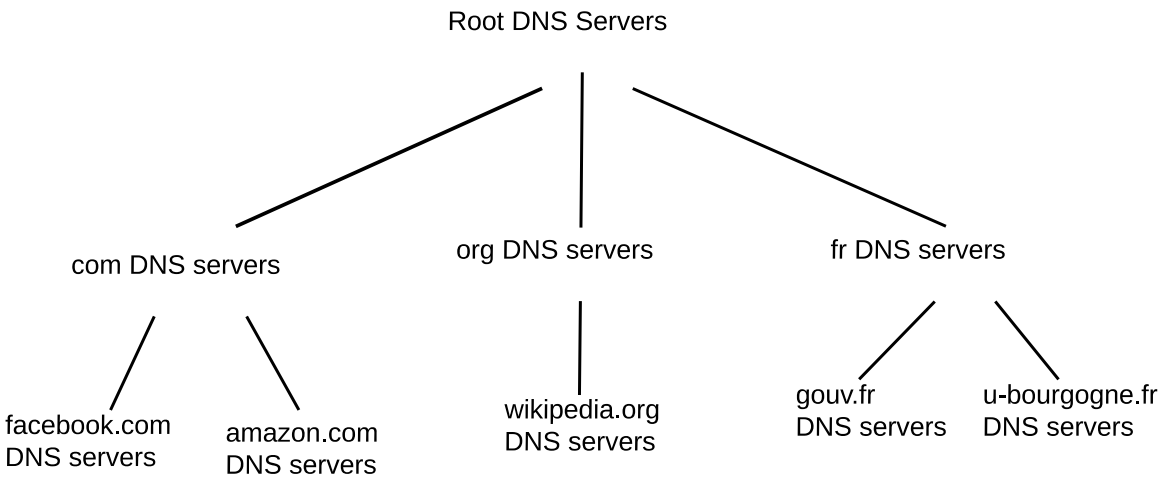
Domain Name System (suite)

Donne aussi d'autres informations comme

- alias d'hôte (1 hôte = nom canonique + alias 1, alias 2, ...)
- alias de mél (ex : smtp.u-bourgogne.fr = zproxy.u-bourgogne.fr)
- équilibrage de charge
  ex : serveurs de google.fr = plusieurs adresses IP
  requête DNS www.google.fr → ensemble d'adresses dans ordre différent à chaque réponse

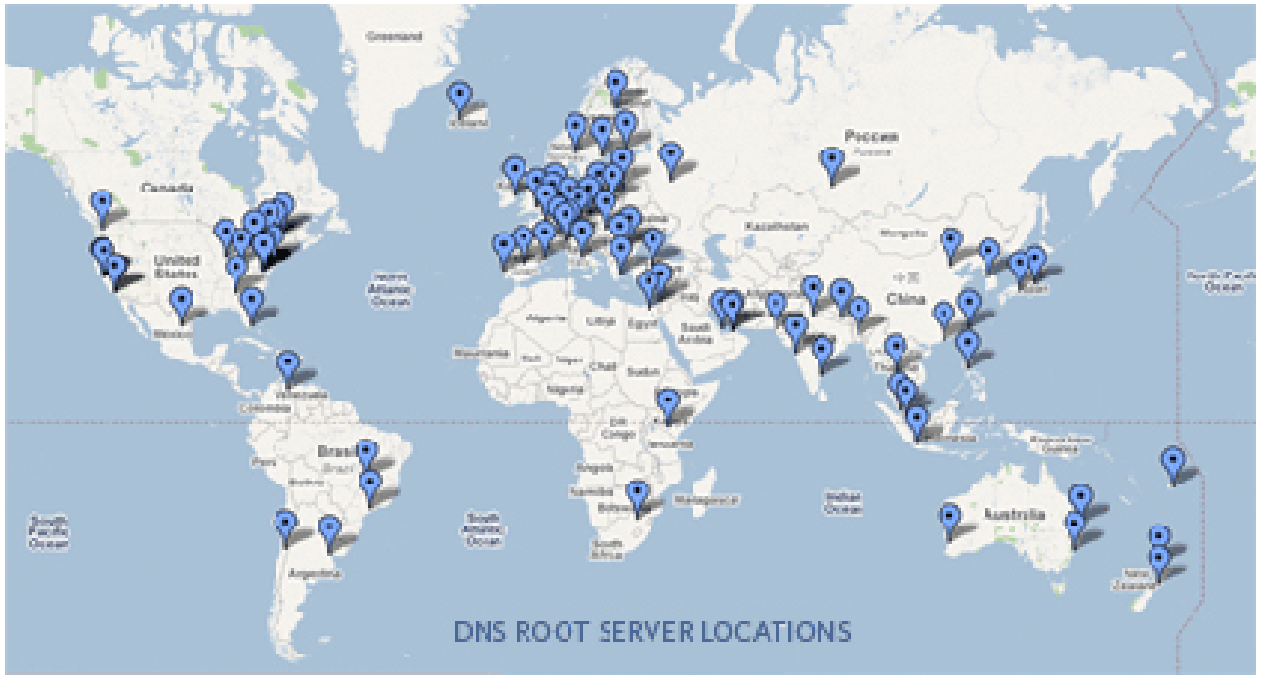
Hiérarchie

Root servers + Top Level Domains (TLD) + Authoritative servers + local servers



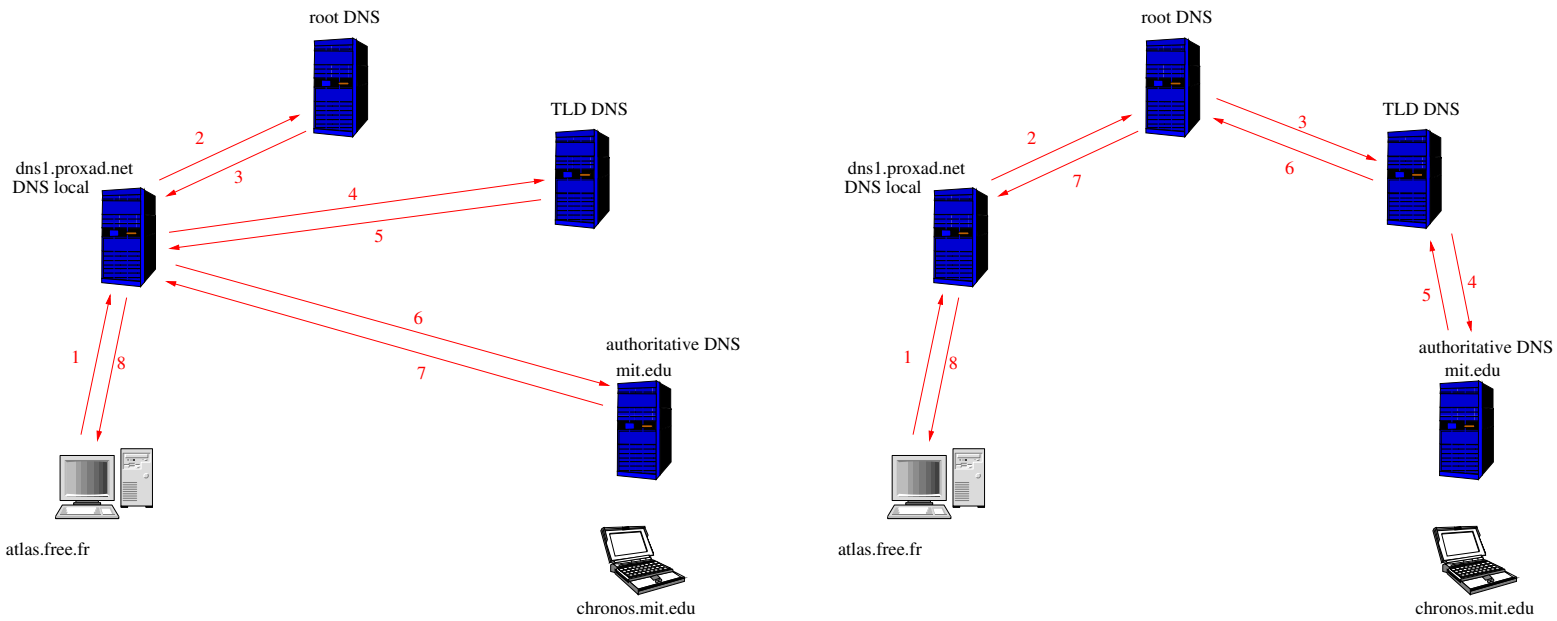
Serveurs racine

13 serveurs racine (root-servers `[a-m].root-servers.net`) dont neuf dupliqués, soit plus de 130 serveurs physiques



(voir `http://www.root-servers.org/`)

Requête itérative/récursive



+ cache (correspondance gardée en mémoire par DNS local)

Enregistrements

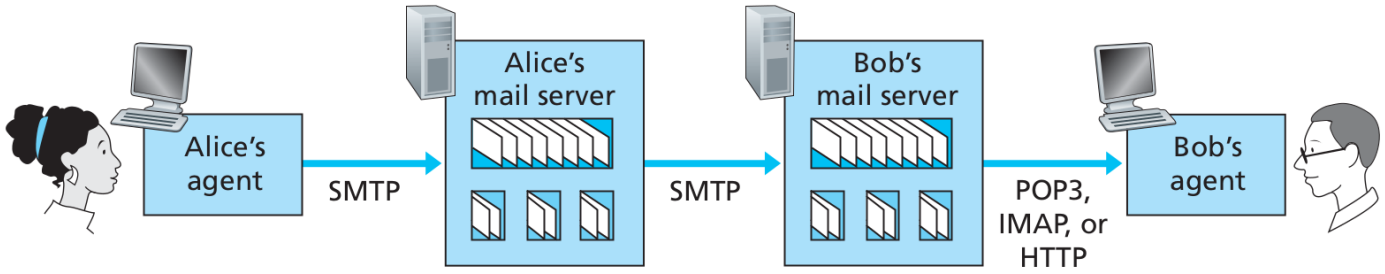
RR = ressource record = (Name, Value, Type, TTL)

- TTL : combien de temps garder la ressource en cache
- Type :

A	Name = hostname, Value = adresse IPv4
AAAA	Name = hostname, Value = adresse IPv6
NS	Name = domain, Value = hostname du serveur ayant authorité
CNAME	Value = cannonical hostname
MX	Value = nom cannonique de serveur mél
PTR	Name = (adresse IP en sens inverse).in-addr.arpa., Value = hostname

Le courriel

- Un des tous premiers services de l'internet (Hotmail : premier web-mél en 1996)
- Communication asynchrone, rapide, peu coûteux, nombreuses améliorations : attachements, hyperliens, texte au format HTML, photos,...
- Repose sur trois composantes : agents utilisateurs, serveurs de mél, protocole SMTP



(de Kurose et Ross, Computer networking : a top-down approach, 6E, Pearson)

Agent utilisateur

- l'envoi de mél pourrait se faire de l'agent utilisateur directement vers le serveur SMTP du récepteur, mais préférable de passer par un serveur SMTP "local"
- plusieurs protocoles d'accès à boite distante : POP3, IMAP, Web-mél
  - POP3 (RFC 1939) est simple mais limité. Trois phases : autorisation, transaction, mise à jour et deux modes : download et delete ou dowload et keep.
  - Internet Mail Access Protocol (IMAP, RFC 3501) est plus complexe mais permet de gérer une hierarchie de dossiers à distance et de ne récupérer qu'une partie d'un mél (en-tête ou partie d'un message MIME).
  - Web-mél : l'envoi depuis le navigateur web vers le serveur SMTP ainsi que l'envoi entre le serveur SMTP et le navigateur du récepteur se font en utilisant HTTP plutôt que SMTP et IMAP ou POP3.

Simple Mail Transfer Protocol (SMTP)

- RFC 5321, mais SMTP date d'avant 1982
- Utilise TCP, partie Client et partie serveur
- Le corps du message doit être codé en ASCII 7-bits (donc les contenus multimédia doivent être encodés avant envoi)
- Dialogue direct de serveur SMTP de l'émetteur avec le serveur SMTP du récepteur. Si srv recept. non actif, attente du coté serv. émet. puis nouvelles tentatives
- Comparaison avec HTTP :
  - Les deux utilisent tous deux des connections TCP persistantes
  - HTTP est un proto. de type "pull" et SMTP de type "push"
  - pas de restriction de codage ASCII 7-bits avec HTTP
  - transfert de texte + image : dans le même message SMTP vs. dans des messages HTTP différents

Simple Mail Transfer Protocol (SMTP)

Exemple de dialogue client (C) - serveur (S) :

> `telnet smtp.u-bourgogne.fr 25`

```
S: 220 zproxy01.u-bourgogne.fr ESMTP Postfix
C: HELO bidule.net
S: 250 zproxy01.u-bourgogne.fr
C: MAIL FROM: <truc@bidule.net>
S: 250 2.1.0 Ok
C: RCPT TO: <e.macron@presidence.gouv.fr>
S: 250 2.1.5 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: Salut vielle branche,
C: ça baigne ?
C: .
S: 250 2.0.0 Ok: queued as 7B14B2D4762
C: QUIT
S: 221 2.0.0 Bye
```

Protocoles de transport

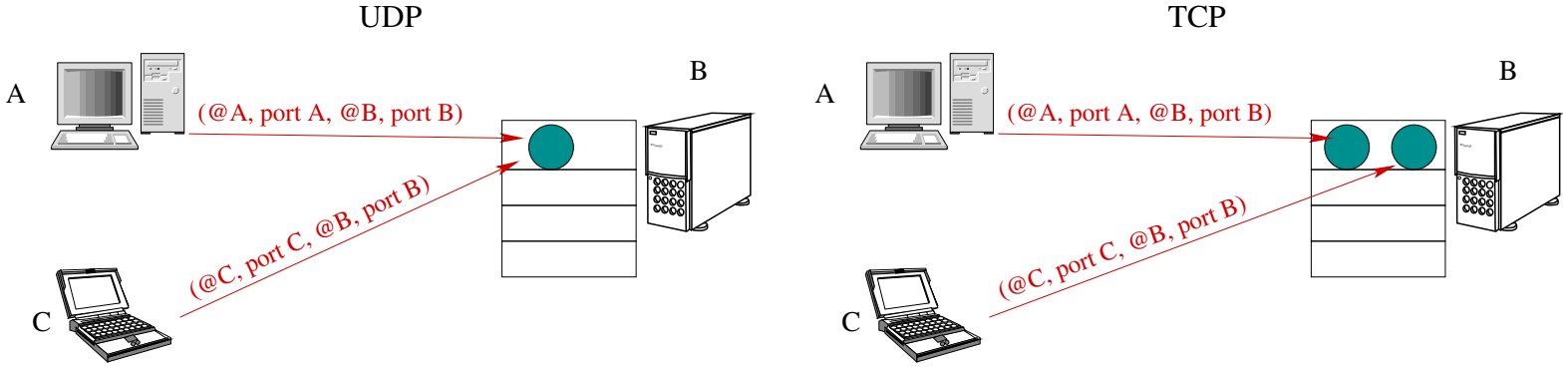
- les protocoles de couche transport proposent une communication logique entre processus sur des hôtes différents
- paquets de transport = segments (parfois appelés datagrammes pour UDP)
- implantés sur les hôtes terminaux (pas sur les routeurs)
- il en existe plusieurs (TCP, UDP, SCTP, DCCP, TFRC, ...)

Multiplexage/démultiplexage

- couche réseau : service d'hôte à hôte
- couche transport : service de process à process
- segments contiennent des champs numéro de port source et destination qui servent à indiquer de quelle application provient le paquet et à quelle application il est destiné
- Trois types de ports :
  1. les ports "système" (0-1023) sont affectés à des applications standard et assignés par IETF Ex : HTTP=80, SSH=22, ... voir `/etc/services` sous Linux et OS X ;
  2. les ports "utilisateur" (1024-49151) assignés par IANA
  3. les ports "dynamiques" et/ou privés (49152-65535)

Identifiants de transport

- UDP : (adr dst, port dst)
- TCP : (adr src, port src, adr dst, port dst)



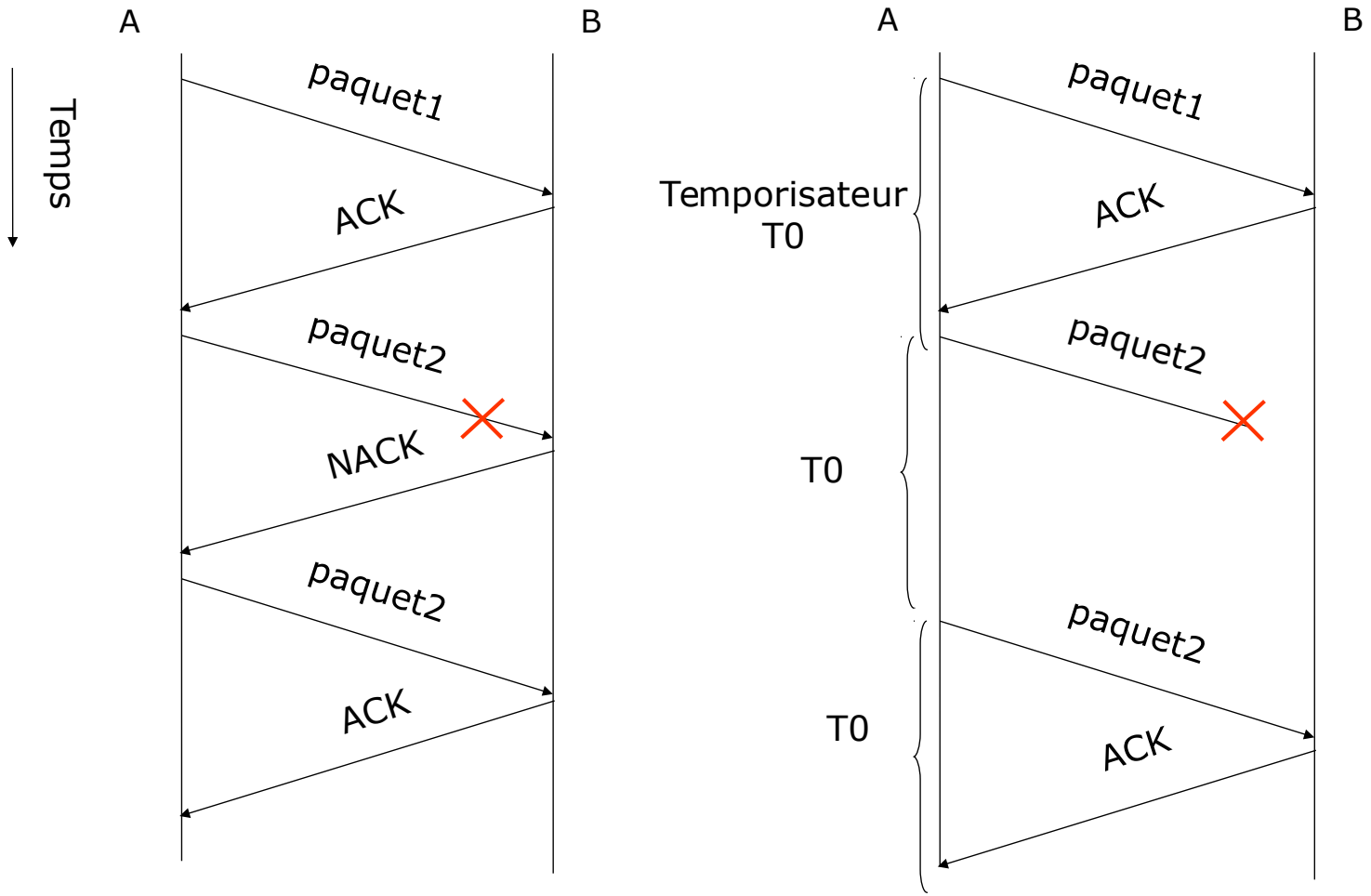
Mécanisme de fiabilisation

- acquittements positifs et négatifs,
- numérotation des paquets,
- numérotation des acquittements.

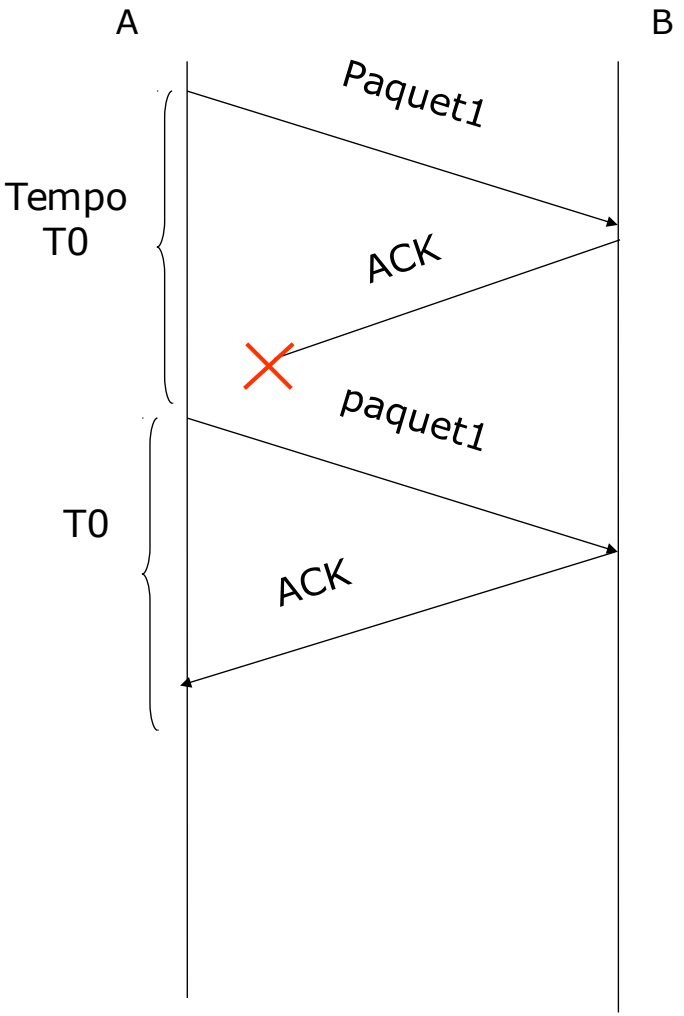
Optimisations :

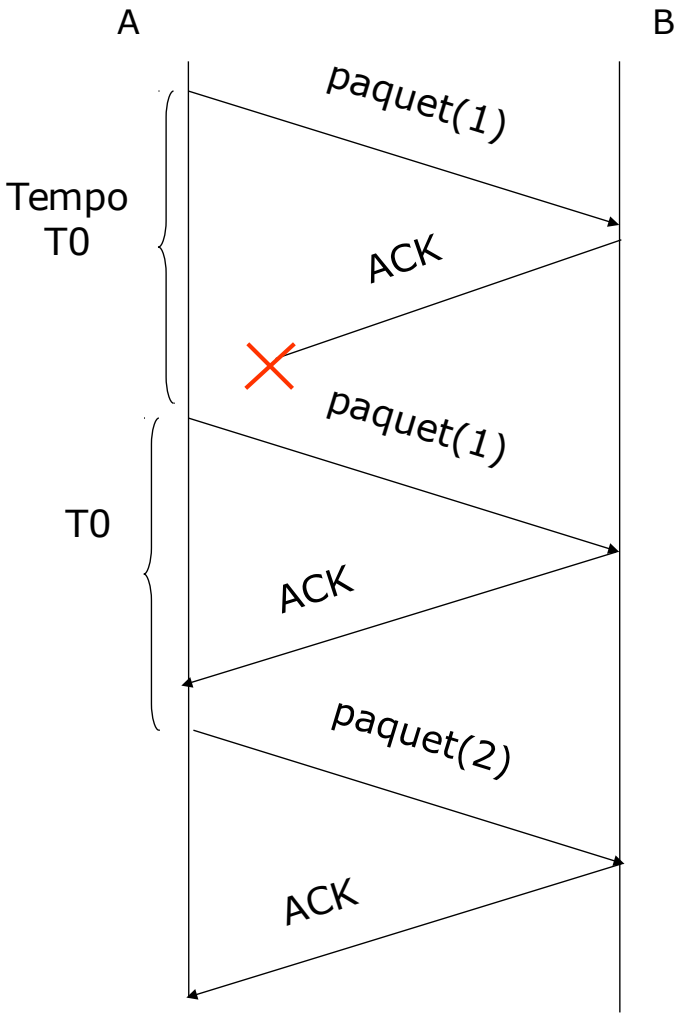
- piggypacking : ajout d'acquittements dans paquets de données envoyés dans l'autre sens
- fenêtre d'anticipation (pipelinage) : permet l'envoi d'un certain nombre de paquets sans attendre d'acquittements $\Rightarrow$ besoin de tampons pour paquets non encore acquittés

Acquittements positifs et négatifs

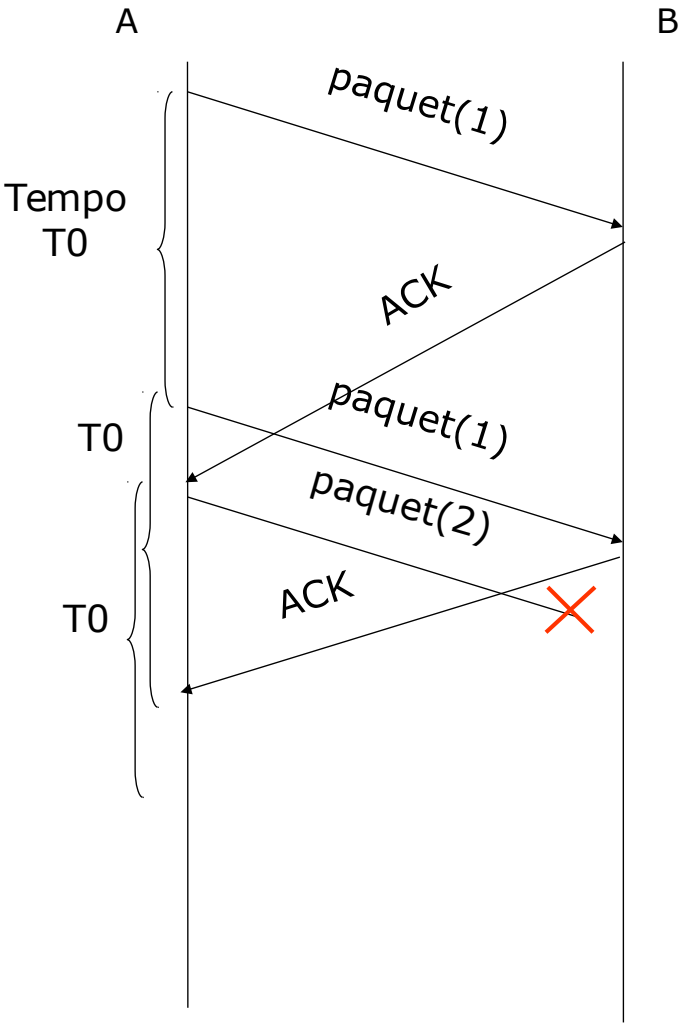


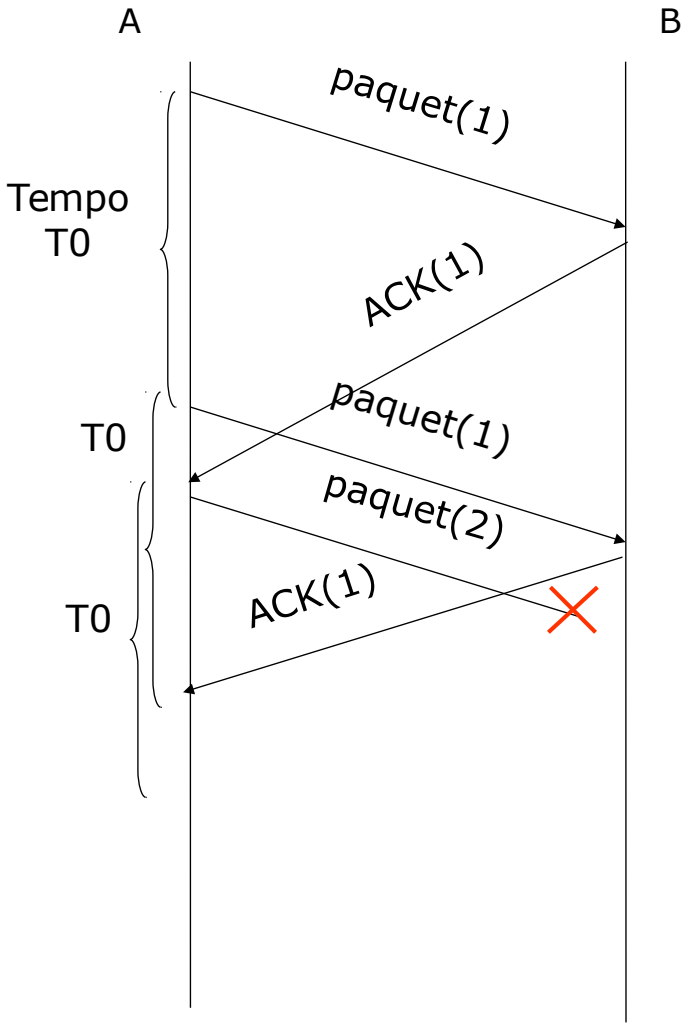
Numérotation des paquets





Numérotation des acquittements





Contrôle d'erreur

- champs Checksum (16 bits) = somme des mots de 16 bits de l'entête et complément à 1
- même si contrôle d'erreurs effectué dans couches inférieures (ex. Ethernet) car contrôle de bout en bout

Ex. pour 001110110101011110000110 sur 8 bits :
$00111011 + 01010111 = 10010010$
$10000110 + 10010010 = 00011011$
Checksum : 11100100

UDP (User Datagramm Protocol, RFC 768)

Service de transport non fiable en mode non connecté, peu de valeur ajoutée par rapport à IP

- multiplexage/démultiplexage
- contrôle d'erreur sur entête mais pas de reprise sur erreur (segment erroné jeté ou passé au process avec warning)

Utile quand :

- données à envoyer très rapidement,
- petites quantités de données,
- perte de paquets pas très importante,
- données multicast

Utilisé par les applis DNS, TFTP, streaming, VoixSurIP

Format d'un paquet UDP

```
 0      7 8     15 16    23 24    31
+--------+--------+--------+--------+
|     Source      |   Destination   |
|      Port       |      Port       |
+--------+--------+--------+--------+
|                 |                 |
|     Length      |    Checksum     |
+--------+--------+--------+--------+
|
|          data octets ...
+---------------- ...
```

length : longueur des données en octets, y compris en-tête
Checksum : somme de contrôle sur pseudo en-tête incluant les adresses IP

TCP (Transmission Control Protocol, RFC 793)

Service de transport fiable en mode connecté (sur IP non fiable, sans connexion) =>

- établissement, maintien et fermeture d'une connexion virtuelle
- acquittements, séquencement et réassemblage des données
- contrôle de flux par fenêtre glissante

Utilisé par la plupart des applis : HTTP, SSH, FTP, SMTP, ...
95% du trafic Internet est transporté par TCP

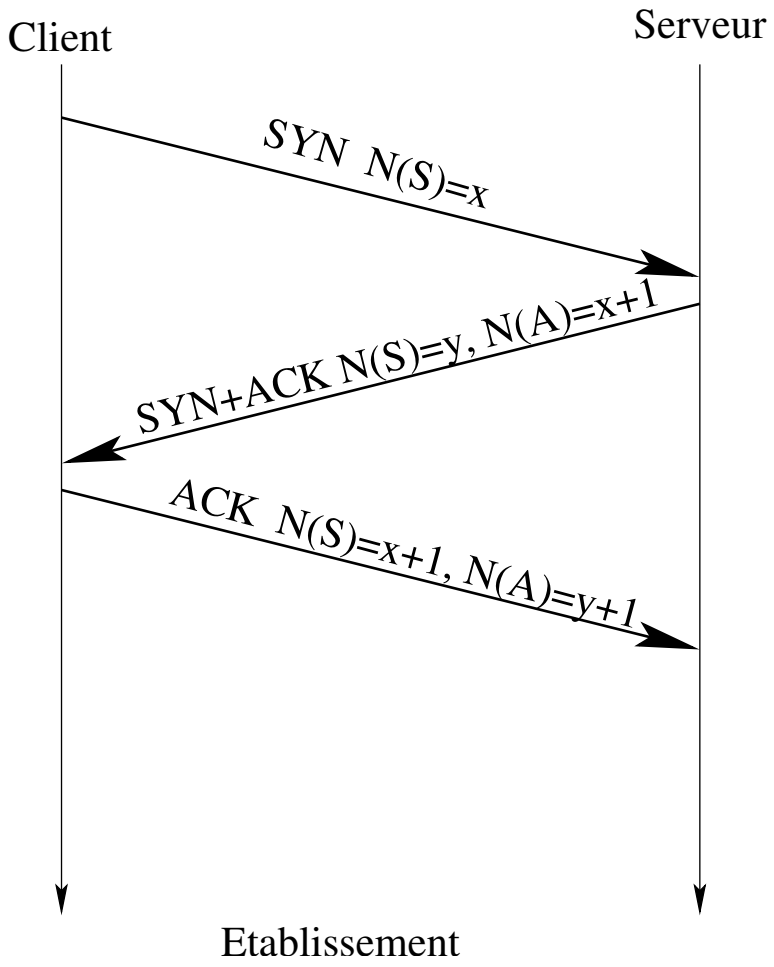
Format des segments TCP

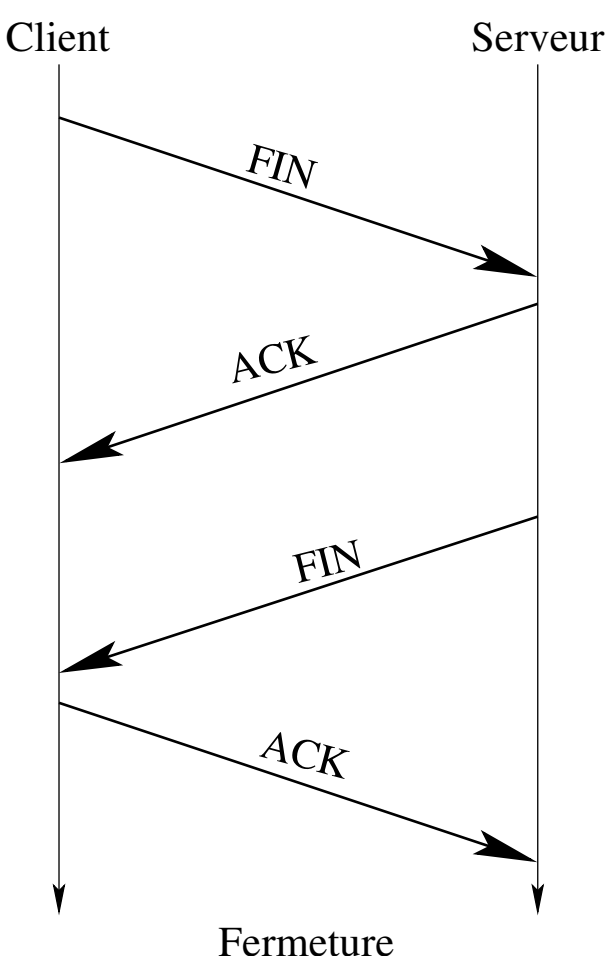
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Data offset = nombre de mots de 32 bits de l'entête

Urgent pointer : position du dernier bit de donné urgente dans le segment

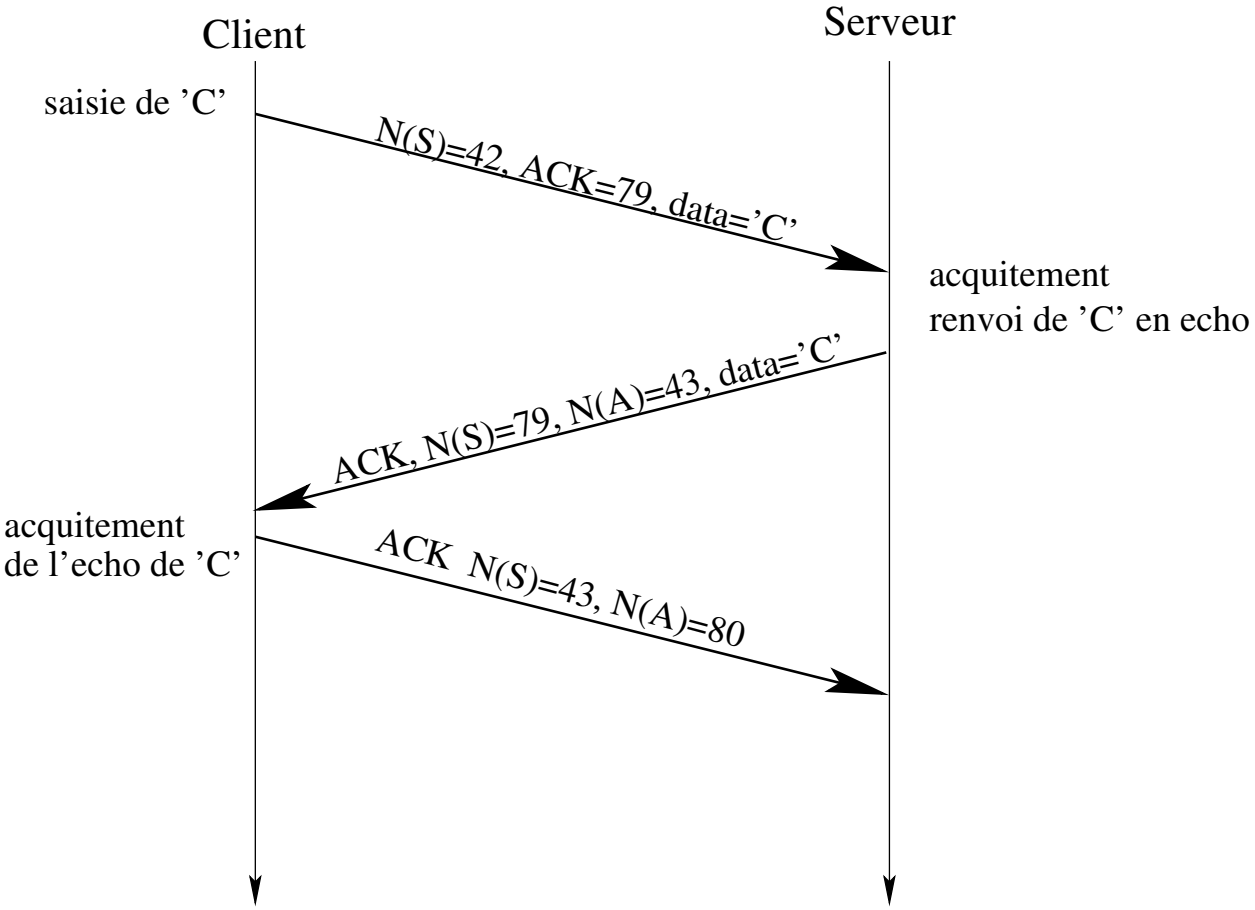
Gestion de la connexion TCP





séquencement TCP

Exemple : session telNet



Temporisation

Combien de temps attendre si pas de ACK reçu suite à l'envoi d'un segment TCP ?

Ce délai (Timeout) doit être supérieur au RTT. Il est calculé à partir d'une estimation du RTT moyen et en y ajoutant une marge de sécurité

$$\text{RTTmoyen} = (1-\alpha)\text{RTTmoyen} + \alpha\text{RTTechantillon}$$
$$\text{VarianceRTT} = (1-\beta)\text{VarianceRTT} + \beta|\text{RTTechantillon} - \text{RTTmoyen}|$$
$$\text{Delai} = \text{RTTmoyen} + 4\text{VarianceRTT}$$

valeurs typiques de α, β : $\alpha = 0.125$ et $\beta = 0.25$

Si retransmission, alors Timeout doublé

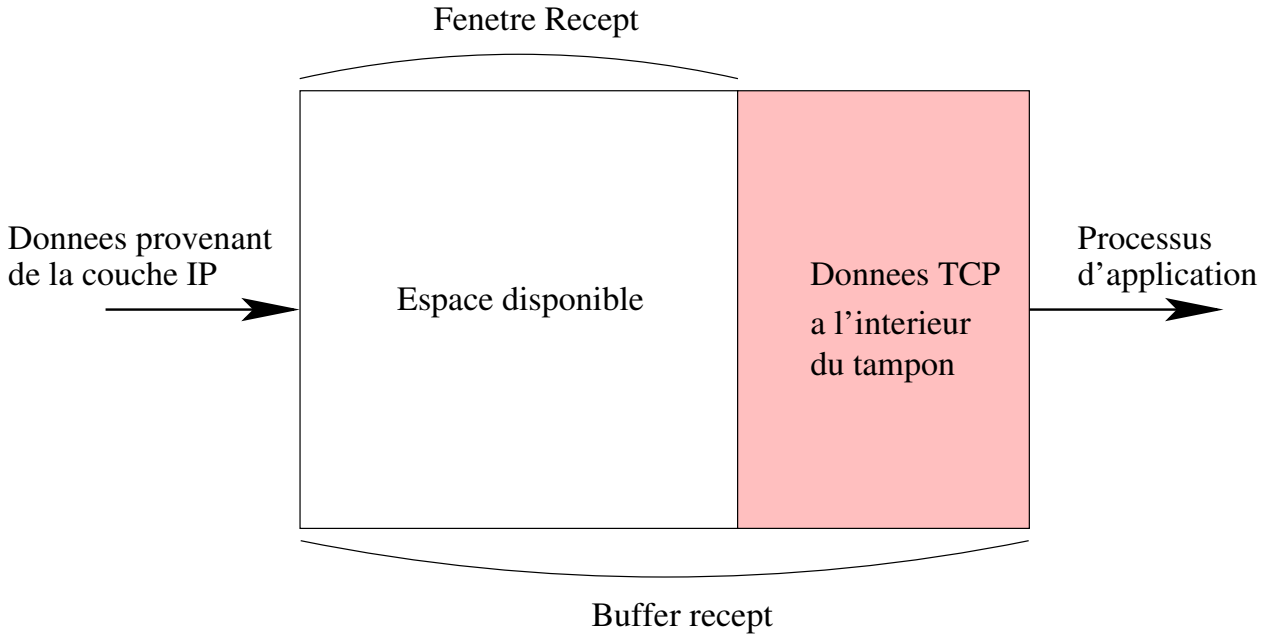
Envoi des acquittements

Timeout peut être long mais possibilité de s'apercevoir d'une perte avant : ACK dupliqués (ACK pour un segment qui a déjà été acquitté)

Evénement	Action
arrivée segment avec NS attendu et data jusqu'à NS déjà acquittées	attendre 500 ms l'arrivée d'un autre segment avant envoi ACK
arrivée segment avec NS attendu et segment reçu en attente ACK	envoyer un ACK immédiatement pour les deux
arrivée segment avec NS plus grand qu'attendu trou)	envoyer ACK dupliqué immédiatement
arrivée segment qui comble un trou partiellement ou totalement	envoyer ACK immédiatement si segment comble le début du trou

Contrôle de flux TCP

Le champs "Fenêtre" (window) du segment TCP indique le nombre d'octets que l'entité est capable de recevoir (taille de la fenêtre de réception)



Contrôle de congestion

Variable *congwin* maintenue par chaque entité TCP : émetteur vérifie que
dernier_octet_envoyé - dernier_octet_acqté $\leq$ min(*congwin*, *fenetre*)
perte = Timeout ou 3 ACK dupliqués $\rightarrow$ indication de congestion pour l'émetteur
Congestion $\rightarrow$ Router overflow $\rightarrow$ perte paquet IP $\rightarrow$ perte segment TCP
Evolution de *congwin* : ACK reçu $\rightarrow$ *congwin* augmentée ; perte $\rightarrow$ *congwin* diminuée
Débit environ *congwin*/*RTT* octets/s

Algorithme de contrôle de congestion

Défini dans RFC 5681 (TCP congestion control, 2009)
Trois modes :

1. Slow start
2. Congestion avoidance
3. Fast recovery (optionnel)

Slow start et Congestion avoidance

- Slow start
  Débit initial : 1 MSS (Maximum Segment Size = MTU -40 octets en-tête IP + TCP)
  ACK reçu $\Rightarrow$ $Congwin+ = 1MSS$ $\Rightarrow$ 2 segments envoyés
  2 ACK reçu $\Rightarrow$ $Congwin+ = 2MSS$
  $\Rightarrow$ doublement du débit à chaque RTT
  Perte $\Rightarrow$ $congwin = 1MSS$
- Congestion avoidance : quand *congwin* arrive à la moité de sa valeur lors de la dernière perte (la congestion est proche), alors $congwin+ = 1MSS$ à chaque RTT

Fast recovery (Reno)

- $congwin+ = 1MSS$ pour chaque ACK dupliqué reçu pour le segment manquant qui a fait passer en mode Fast recovery
- Timeout $\Rightarrow$ passage en mode Slow Start
- De nombreuses autres variantes de l'algorithme de contrôle de congestion de TCP existent (Vegas, CUBIC, ...)
- En conclusion : débit en "dents de scie", de type AIMD (additive increase, multiplicative decrease)

Alternatives à TCP

- DCCP (Datagram Congestion Control Protocol), RFC 4340 = UDP + controle de congestion
- TFRC (TCP Friendly Rate Control), RFC 5348, utilisable avec DCCP, par ex. pour le multimédia
- SCTP (Stream Control Transmission Protocol), RFC 4960
- QUIC : A UDP-Based Multiplexed and Secure Transport, RFC 9000 : service de transport fiable et sécurisé qui utilise UDP pour réduire la latence