R3.06 Architectures des réseaux

Jean-François Remm

IUT Nantes département Informatique

2022–2023

Objectif

Compétences ciblées :

- Installer, configurer, mettre à disposition et maintenir en conditions opérationnelles des infrastructures, des services et des réseaux et optimiser le système informatique d'une organisation

Savoir de références étudiés :

- Technologie des réseaux (piles, couches transport, TCP/IP/UDP, DHCP, DNS, ...)
- Interconnexions de réseaux (par ex : routage, NAT, filtrage, proxy)

Organisation du module

Volume horaire : 14H

- par semaine : 1H20 CM + 2H40 TD

Modalité d'évaluation :

- 1 test coeff. 1
- 1 note de TD coeff. 1

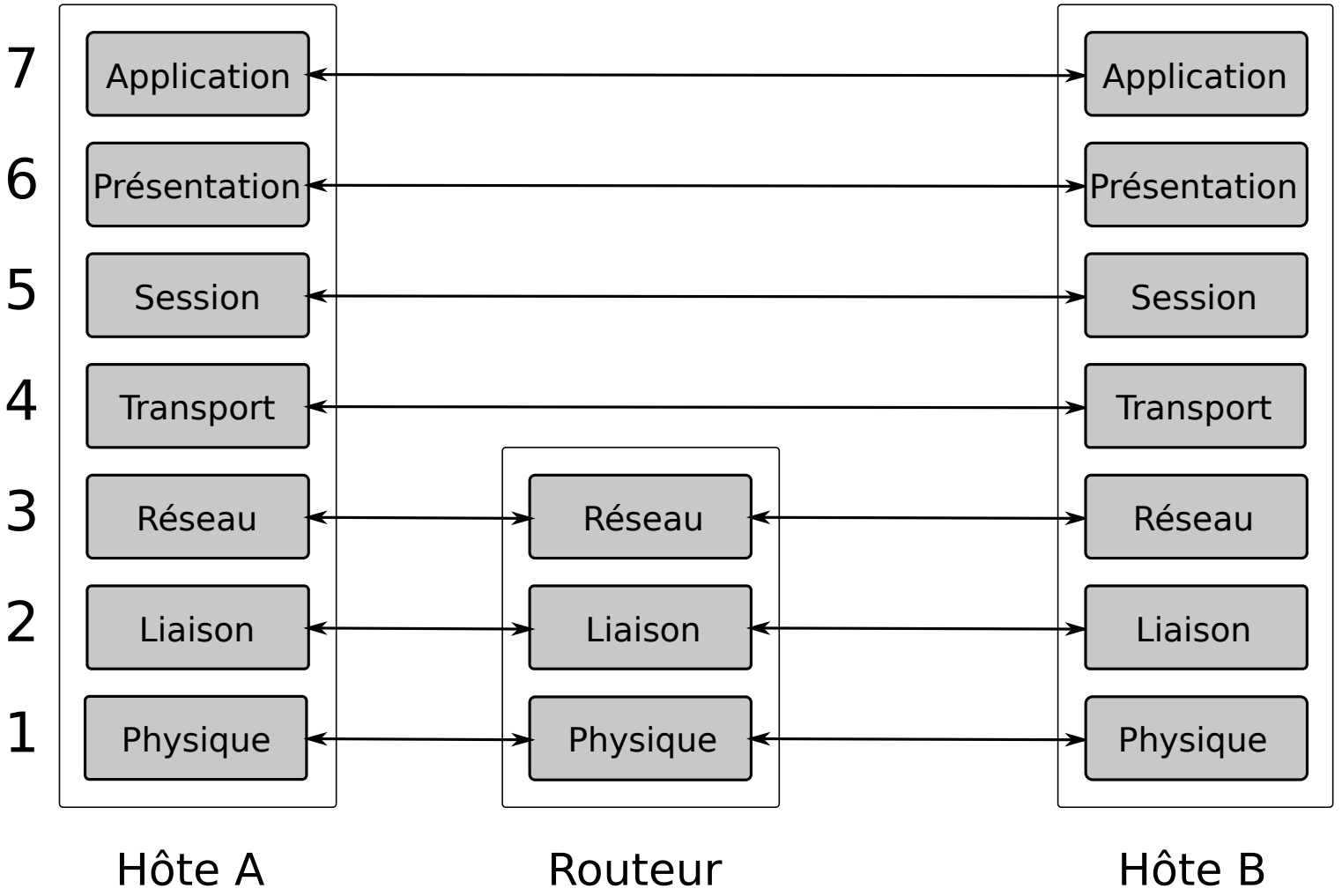
Équipe pédagogique

- CM : Jean-François Remm
- TD : Erwann Helleu, Nicolas Hernandez et Jean-François Remm

Remerciements – Contributeurs – Sources

- cours ASR (Administration Système–Réseau) de P. Levasseur
- travail collaboratif réalisé avec J.-F. Berdjugin et P.-A. Jacquot à de l'IUT1 Grenoble
- cours d'administration réseau de H. Pinvidic (M2 Alma)

Modèle OSI


7
6
5
4
3
2
1
Application
Présentation
Session
Transport
Réseau
Liaison
Physique
Application
Présentation
Session
Transport
Réseau
Liaison
Physique
Réseau
Liaison
Physique
Hôte A
Routeur
Hôte B

Couche 1 : physique

- fournit les moyens nécessaires aux transferts des élément binaires
  - câblage (coaxial, paire torsadées, fibres optique)
  - interfaçage de connexion (prise RJ45, connecteur BNC, ...)
  - codage des bits (niveau électrique)
  - équipement de transmission (modems, commutateurs, ...)
  - topologie du réseau
- spécifie **canal** et **signal** sans aucune sémantique de l'information

$\Longrightarrow$ niveau bit

Canal

- le canal est constitué de tout médium capable d'assurer le transfert d'une information binaire
- caractérisé par une **bande passante** (bande des fréquences utilisés) :

$$W = F_{max} - F_{Min}$$

- Exemples :

type	bande passante
paire torsadée	$>$ 100 kHz
cable coaxial	$>$ 100 MHz
fibre optique	$>$ 1 GHz
espace entre 2 antennes	variable

- les données binaires sont transportés par un **signal**

Débit

- les éléments transmis sur un réseaux : 0 ou 1
- le **débit** mesure la rapidité d'une communication numérique
- débit = nombre d'éléments binaires transmis par seconde
- unité : bits par seconde (bps)
- Exemple : D = 10 Mbps
- formule de Shannon :

$$D = W log_2(1 + \frac{S}{N})$$

Signal

- **Signal** : variation d'une grandeur physique, porteuse d'informations
- Exemple :
  - signal électrique : tension
  - signal optique : onde lumineuse
  - signal hertzien : onde électromagnétique

Transmission

- Deux types de transmissions :
  1. transmission en bande de base
  2. transmission d'un signal modulé

Transmission en bande de base

- suite de bits représentant les données numériques,
- changement d'états discret du signal physique,
- pas de transposition en fréquence,
- durée de chaque bit est constante

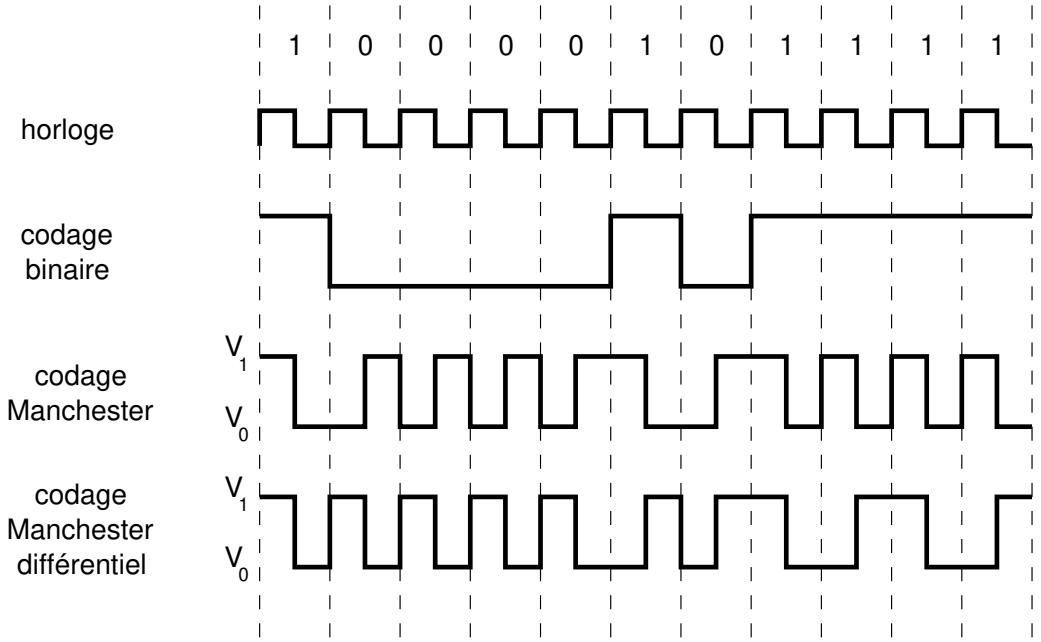


Figure – Transmission en bande de base

Transmission d'un signal modulé

- utilisation d'une onde porteuse
- modification pour augmenter le débit, diminuer le taux d'erreur

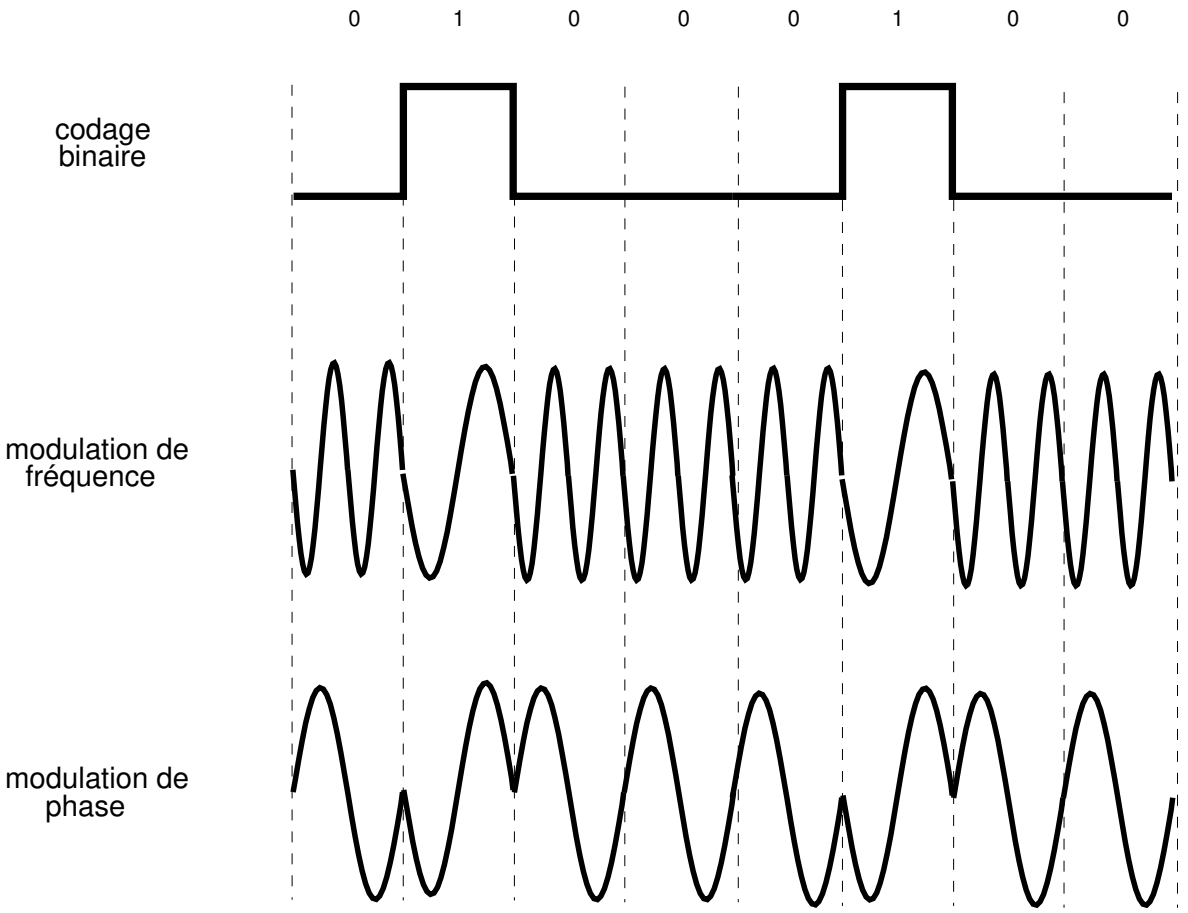


Figure – Transmission d'un signal modulé

Couche 2 : liaison de données

- transforme un flot binaire brut en **trames**
- gère l'établissement, le maintien et la libération de la liaison entre **terminaux**
  - transmission
  - contrôle de flux
  - contrôle d'erreur
  - adressage des terminaux
  - accusé de réception

$\Longrightarrow$ niveau trame

Couche 1bis : MAC *Medium Acces Control*

- sous-couche de contrôle d'accès au canal pour un réseau à diffusion
- mécanisme d'adressage des hôtes : adresses MAC
- protocole de gestion d'accès :
  1. CSMA/CD
  2. CSMA/CA

Couche 3 : réseau

- permet la connexion de réseaux entre systèmes ouverts :
  - fonction d'**adressage** des réseaux
  - fonction de **routage/relayage** pour l'acheminement d'un datagramme
- doit permettre l'interconnexion de réseaux hétérogènes

$\Longrightarrow$ niveau paquet

Couche 4 : transport

- reçoit des messages des couches supérieures et découpe ces messages en **segments** ou **datagramme** avant transmission aux couches inférieures
- optimise les ressources réseau :
  - contrôle de flux : ordonnancement, gestion des pertes
  - correction des erreurs

$\Longrightarrow$ niveau segment ou datagramme

Couche 5 : session

- fournit à la couche 6 des moyens de synchroniser le dialogue
- définit les séquences de l'échange :
  - échange bi- ou unidirectionnel
  - point de reprise, retour arrière

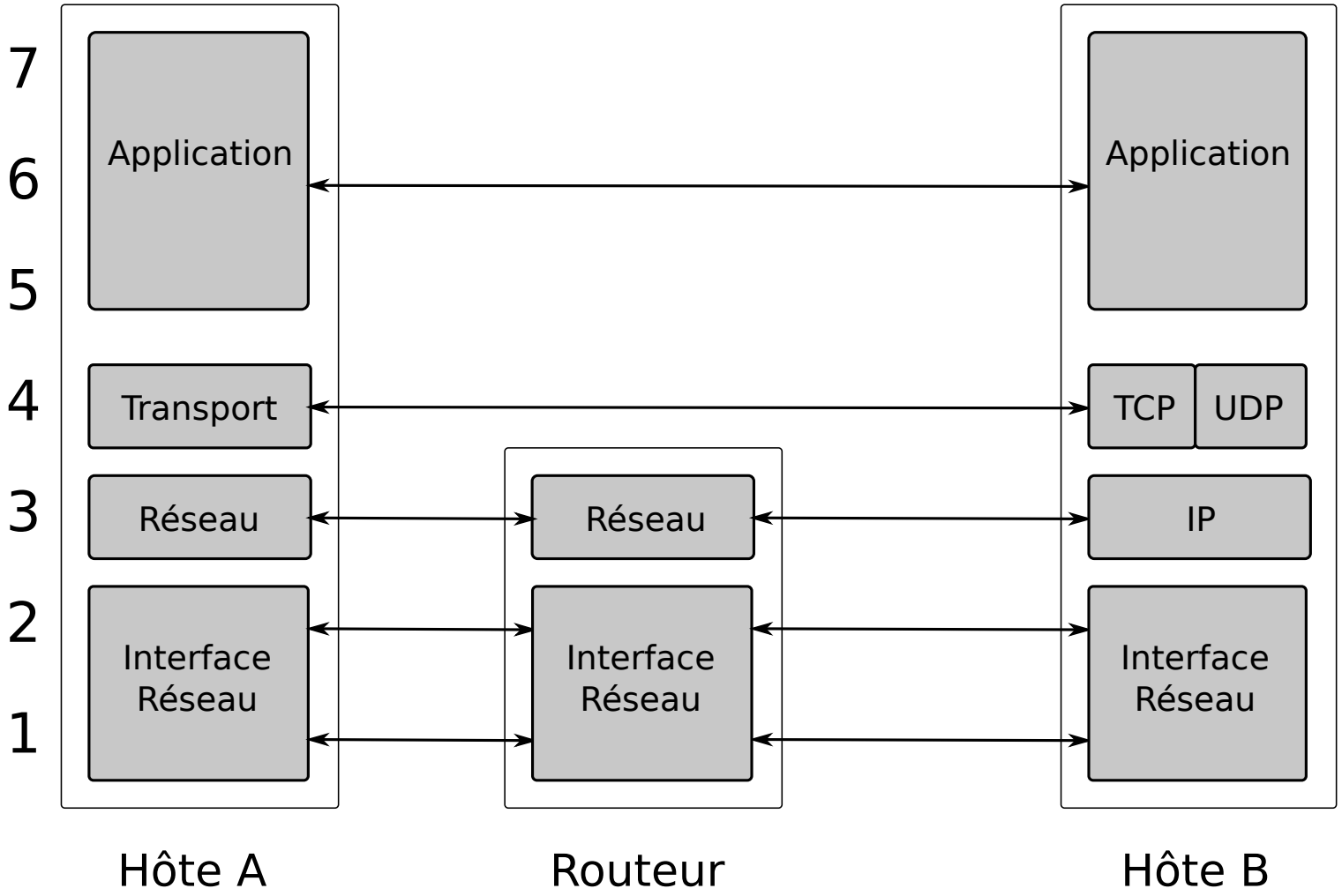
Couche 6 : présentation

- Définit syntaxe et sémantique des informations :
  - traduction (ex : ASN.1)
  - compression et décompression de données
  - chiffrement, déchiffrement

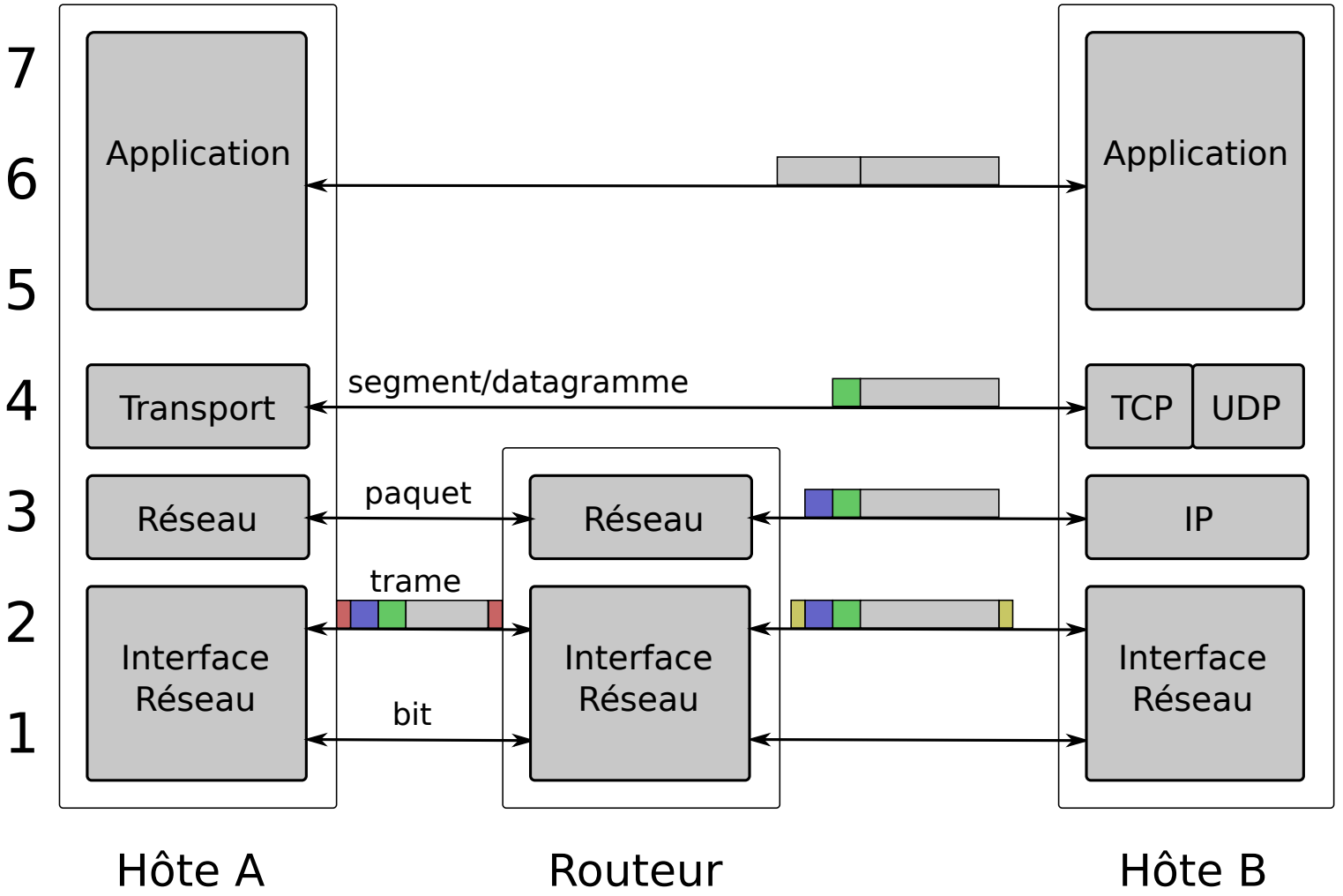
Couche 7 : application

- chargé de l'exécution de l'application
- applications "classiques"
  - mail
  - web
  - transfert de fichiers
  - groupes de discutions

Modèle TCP/IP



Encapsulation



Rappel

Éléments d'interconnexion :

- couche 1 : câbles, cartes réseaux, répéteur (*repeater*), concentrateur (*hub*)
- couche 2 : pont (*bridge*), commutateur (*switch*)
- couche 3 : routeur (*router*)
- couches supérieures : passerelles applicatives (*gateway*).

ARP

Association entre adresse réseau (IP) et adresse MAC (Éthernet)

- arp

```
#arp -s 192.168.1.1 ca:fe:00:ca:fe:00
#arp
Address          HWtype  HWaddress           Flags Mask  Iface
172.21.60.1      ether   00:e0:b1:a9:75:c0   C           eth0
192.168.1.1      ether   00:ca:fe:00:ca:fe   CM          eth0
```

- iproute2

```
#ip neighbor add 192.168.1.1 lladdr 00:ca:fe:00:ca:fe dev eth0

#ip neighbor show
172.21.60.1 dev eth0 lladdr 00:e0:b1:a9:75:c0 STALE
192.168.1.1 dev eth0 lladdr 00:ca:fe:00:ca:fe PERMANENT
```

Configuration MAC

consulter ou changer son adresse MAC.

- utiliser `ifconfig`

```
#ifconfig eth0 hw ether 00:ca:fe:00:ca:fe

#ifconfig
eth0      Link encap:Ethernet  HWaddr 00:ca:fe:00:ca:fe
          ...
```

- utiliser les fonctionnalités d'iproute2

```
#ip link set eth0 addr 00:ca:fe:00:ca:fe

#ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
    UNKNOWN mode DEFAULT group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state
    UP mode DEFAULT group default qlen 1000
    link/ether 00:ca:fe:00:ca:fe brd ff:ff:ff:ff:ff:ff
```

Adresse IP

- Chaque interface réseau (i. e. carte réseau) d'un hôte du réseau possède (au moins) une adresse IP unique dans le réseau.
- Une interface ethernet est désignée par ethX ou X est le numéro de l'interface.
- Une adresse IP est un nombre de 32 bits souvent noté en décimal pointé ; quatre entiers (compris entre 0 et 255) séparés par des points. Exemple : 192.168.2.200.
- L'adresse IP est structurée en deux parties :
  1. partie réseau : permet de désigner le réseau (*netID*)
  2. partie hôte : permet de désigner l'hôte dans le réseau (*hostID*)
- Un masque de sous-réseau permet de séparer partie réseau et partie hôte.

CIDR

- CIDR *Classless Inter-Domain Routing* : suppression de la notion de classe réseau
- notation /X où X désigne le nombre de bits à 1 dans le masque de sous-réseau
- permet d'agréger plusieurs réseaux en un seul
- permet de découper un réseau en plusieurs
- permet de simplifier les tables de routages
- exemple :

```
125.0.0.0/8
135.18.0.0/16
195.220.84.0/24
193.48.96.0/20
192.168.128.0/27
```

Utilisation du masque

- Ce masque est une succession de 1 suivi d'une succession de 0 qui donne l'étendue de la partie réseau.
- mettre tous les bits de la partie réseaux à 0 nous donne la partie hôte.
- mettre tous les bits de la partie hôte à 0 nous donne la partie réseau.
- exemple : 192.168.168.38 avec masque de 255.255.255.224

11000000	10101000	10101000	00100110	192.168.168.38
11111111	11111111	11111111	11100000	255.255.255.224
11000000	10101000	10101000	00100000	$\Longrightarrow$ 192.168.168.32

- mettre tous les bits de la partie hôte à 1 nous donne l'adresse de diffusion dans le réseaux..

Configuration IP

Trois solutions :

- configurer le fichier `/etc/network/interface` en remplaçant

  ```
  iface eth0 inet dhcp
  ```

  par :

  ```
  iface eth0 inet static
  address 192.168.1.1
  netmask 255.255.255.0
  ```

- utiliser `ifconfig`

  ```
  #ifconfig eth0 192.168.1.1/24
  ```

- utiliser les fonctionnalités d'iproute2 (consultation `ip addr show`)

  ```
  #ip addr add 192.168.1.1/24 dev eth0
  ```

Configuration IP

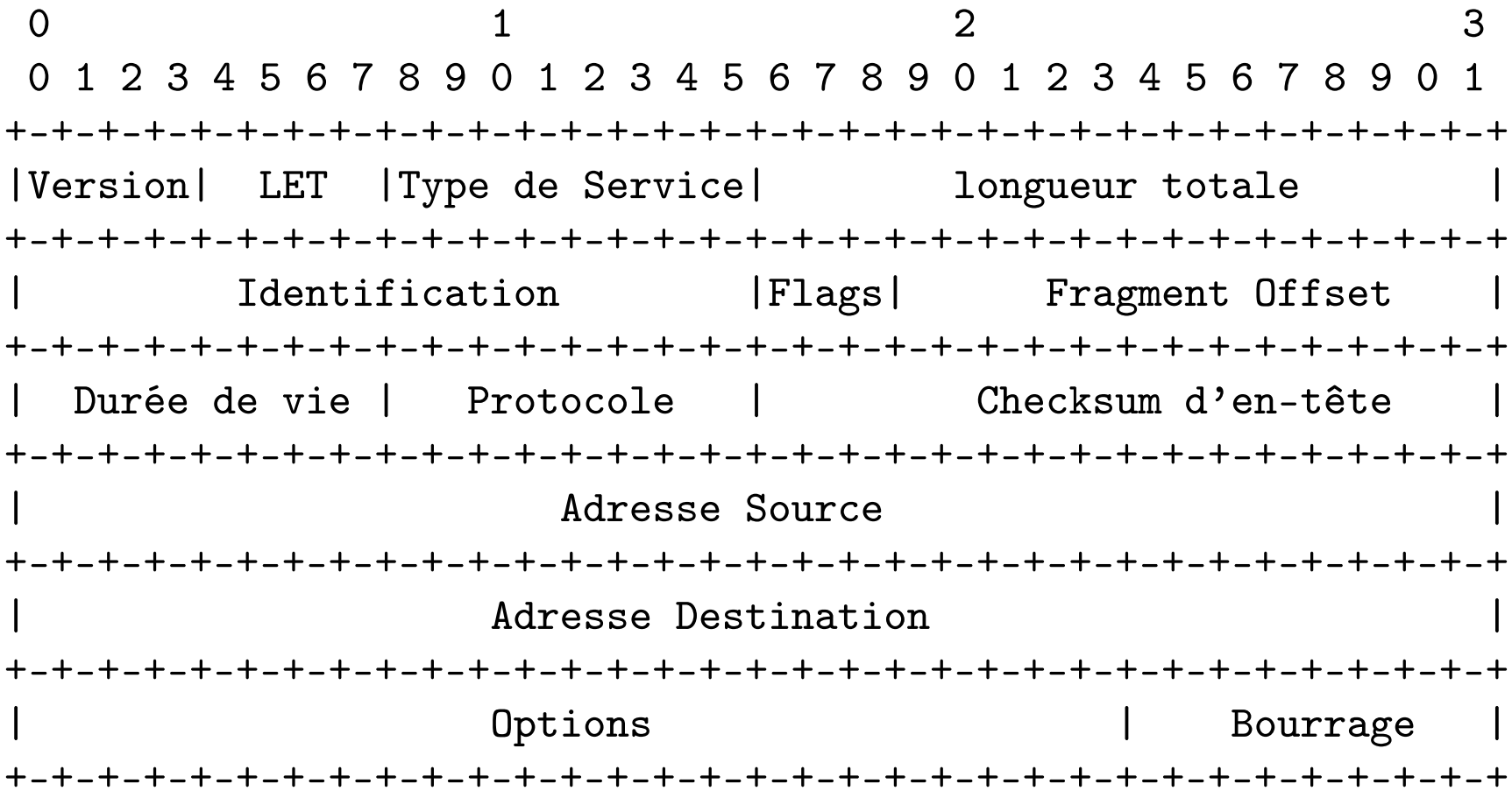
```
#ifconfig
eth0    Lien encap:Ethernet  HWaddr 00:00:C0:9A:01:F2
        inet adr:192.168.0.7 Bcast:192.168.0.255 Masque:255.255.255.0
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:197 dropped:0 overruns:0 carrier:197
        collisions:0 lg file transmission:100
        RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
        Interruption:10 Adresse de base:0xc400

lo      Lien encap:Boucle locale
        inet adr:127.0.0.1  Masque:255.0.0.0
        UP LOOPBACK RUNNING  MTU:16436  Metric:1
        RX packets:188 errors:0 dropped:0 overruns:0 frame:0
        TX packets:188 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 lg file transmission:0
        RX bytes:14264 (13.9 Kb)  TX bytes:14264 (13.9 Kb)
```

Configuration IP

```
#ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast st
    link/ether 00:00:c0:9a:01:f2 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.1/24 brd 192.168.1.255 scope global eth0
    inet6 fe80::200:c0ff:fe9a:01f2/64 scope link
       valid_lft forever preferred_lft forever
```

Paquet IP


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
Version
LET
Type de Service
longueur totale
Identification
Flags
Fragment Offset
Durée de vie
Protocole
Checksum d'en-tête
Adresse Source
Adresse Destination
Options
Bourrage

Routage

- le routage permet l'interconnexion de réseaux
- il est inutile sur un LAN isolé.
- l'information transite d'un réseau à l'autre par des hôtes spécialisés appelés routeur
- un routeur possède une connexion sur chacun des réseaux qu'il interconnecte
- le routeur maintient une table de routage

Routage : principe

- Un hôte voulant faire une transmission constitue un paquet IP
- Ce paquet contient l'adresse du destinataire et l'adresse de l'expéditeur.
- Au niveau de la couche réseau, le routage utilise une **table de routage** qui contient une ou plusieurs lignes contenant chacune essentiellement trois informations :
  1. une adresse de réseau : la destination
  2. un masque de réseau
  3. comment atteindre le réseau :
     - soit **directement** par une interface connectée sur ce réseau (on parle de routage direct),
     - soit **en passant par un routeur** (on parle de routage indirect) qui est identifié par son IP et l'interface à utiliser pour l'atteindre.
- Un routeur peut être un équipement spécialisé ou simplement un hôte ordinaire relié à plusieurs réseaux.

Routage : mécanisme

- la décision de routage se fait par la recherche d'une correspondance dans la table de routage
- on applique pour chaque ligne, le masque de réseau à l'adresse de destination.
- Quatre cas peuvent alors se présenter :
  1. le réseau de la destination est directement connecté. Il y une remise directe en utilisant le réseau local sous-jacent.
  2. le réseau de la destination est accessible via un routeur. Le paquet est transmis au routeur sans changer les adresses IP de l'émetteur et du destinataire.
  3. le réseau de la destination est absent de la table de routage, mais il existe une route par défaut. Le paquet est transmis au routeur désigné.
  4. le réseau de la destination est absent de la table de routage, et il n'existe pas de route par défaut. Envoi d'un message ICMP à l'émetteur : `Network is unreachable`
- Chaque routeur recevant un paquet IP applique le même algorithme.

Configuration du routage

- routage non adaptatif ou routage statique : la table de routage est gérée manuellement
- routage adaptatif ou routage dynamique : la table de routage est gérée automatiquement. C'est à dire qu'un routeur du réseau applique un algorithme qui, en fonction d'information sur l'état du réseau, lui permet de calculer sa table de routage.

Routage statique

Trois solutions :

- configurer le fichier `/etc/network/interface` en remplaçant

  ```
  iface eth0 inet dhcp
  ```

  par :

  ```
  iface eth0 inet static
  address 192.168.1.1
  netmask 255.255.255.0
  gateway 192.168.1.254
  # route statique supplementaire
  up route add -net 172.20.11.0/16 gw 192.168.1.253 dev eth0
  ```

- utiliser la commande `route`

  ```
  #route add -net 172.20.11.0/16 gw 192.168.1.253
  ```

- utiliser la commande `ip`

  ```
  #ip route add 172.20.11.0/16 via 192.168.1.253
  ```

Routes

- `route`

```
#route
Table de routage IP du noyau
Destination     Passerelle      Genmask          Indic Metric Ref
172.21.60.0     *               255.255.252.0    U     0      0
10.0.0.0        172.21.63.5     255.0.0.0        UG    0      0
default         172.21.60.1     0.0.0.0          UG    100    0
```

- `ip route show`

```
#ip route
172.21.60.0/22 dev eth0  proto kernel  scope link  src 172.21.62.8
10.0.0.0/8 via 172.21.63.5 dev eth0
default via 172.21.60.1 dev eth0  metric 100
```

Routage dynamique

Principe :

- les routeurs échangent des informations avec leurs voisins
- ils se servent de ces informations pour établir leur table de routage
- les modifications de le topologie des réseaux (coupures, nouvelles liaisons) sont prises en comptes automatiquement.

Routage dynamique

Les deux familles de protocoles les plus répandues sont :

1. protocoles à vecteurs de distance
2. protocoles à états de lien

Dans les deux cas,

Protocoles à vecteur de distance

- échange local d'informations globales
- table d'un routeur transmise à ses voisins
- table de routage $\simeq$ unions tables de routage
- la distance est le nombre de sauts à faire (*hops*)
- algorithme de Bellman-Ford.
- exemple : RIP
- inconvénients : convergence lente, 15 sauts maximum, problème du "comptage à l'infini", ...

Protocoles à état de lien

- échange global d'informations locales
- état des liens transmis à tous les routeurs
- principe :
  - découverte des voisins
  - mesure de la distance à chaque voisin
  - construction d'un paquet contenant ces informations
  - transmissions aux autres routeurs
  - chaque routeur calcule le chemins le plus courts vers les autres (algorithme *Shortest Path First* de Dijkstra)
- différentes métriques peuvent être utilisées : qualité du lien, encombrement, le coût financier...
- exemple : OSPF
- inconvénients : charge de calcul

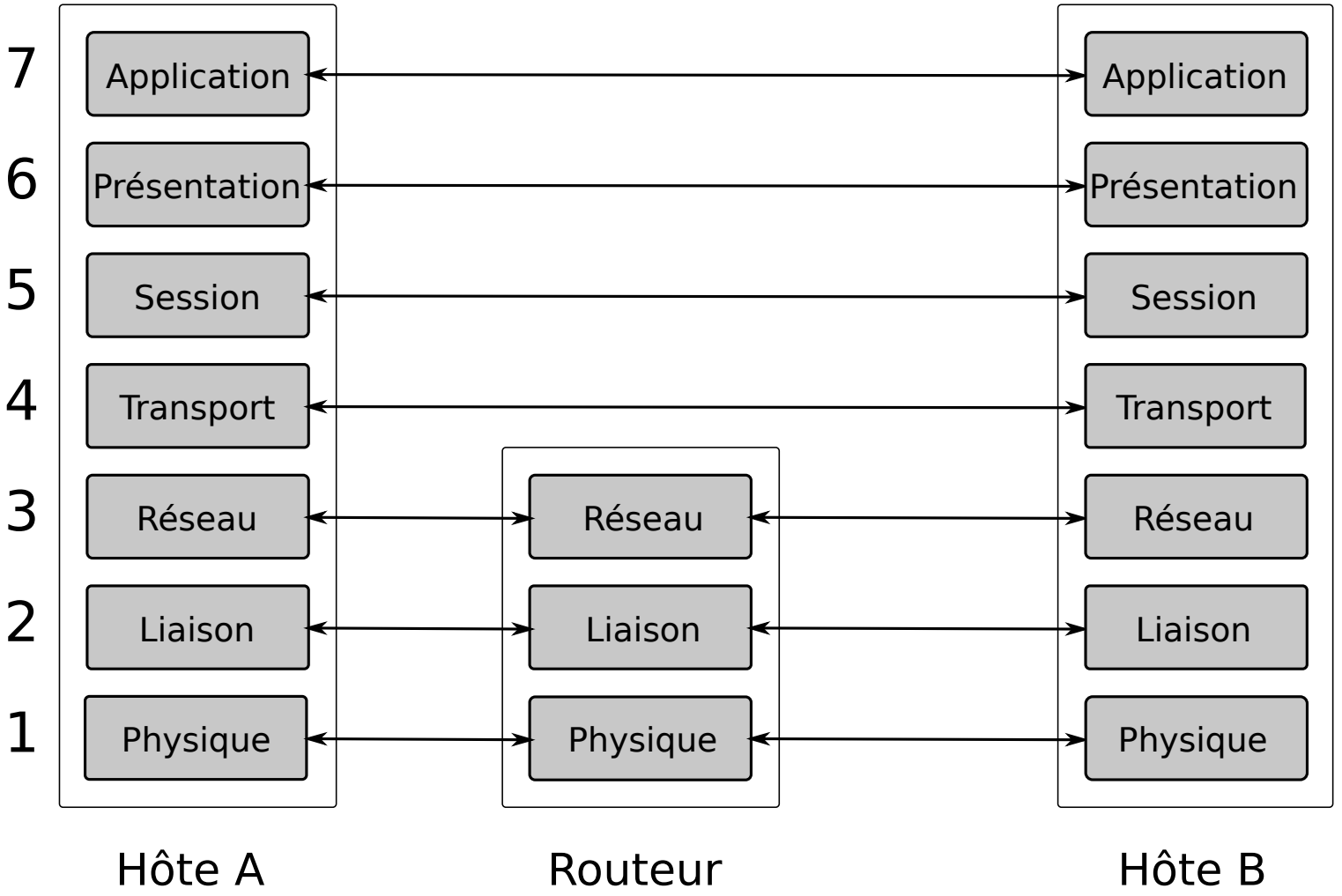
But

- La couche réseau offre une communication de machine à machine.
- La couche transport permet :
  - une communication d'application à application,
  - un transfert fiable :
    - sans corruption : checksum
    - sans pertes : FOO BAR $\nrightarrow$ BAR
    - dans l'ordre : FOO BAR $\nrightarrow$ BAR FOO
    - sans duplication : FOO BAR $\nrightarrow$ FOO FOO BAR
  - des services avec ou sans connexion,
  - d'offrir différentes qualités de service (QoS) pour le mode connecté,
  - une communication de bout en bout.

Connecté/Non Connecté

- Mode connecté : établissement d'une connexion avant transmission puis transmission et libération de la connexion (téléphone)
- Mode non connecté : transmission directe (courrier postal)

Transport

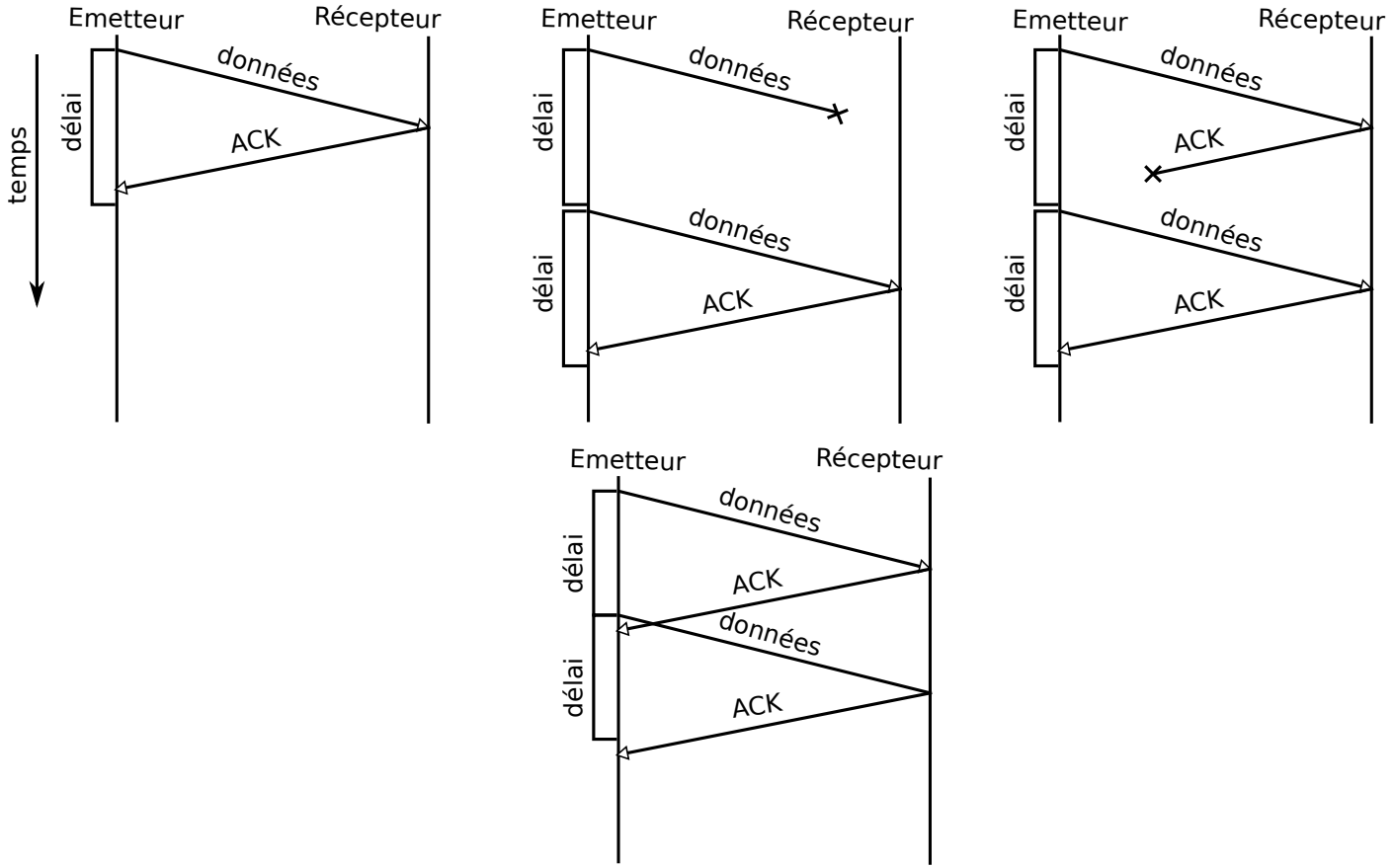

7
6
5
4
3
2
1
Application
Présentation
Session
Transport
Réseau
Liaison
Physique
Application
Présentation
Session
Transport
Réseau
Liaison
Physique
Réseau
Liaison
Physique
Hôte A
Routeur
Hôte B

Problèmes à résoudre

- Transport fiable
- Adressage
- Établissement/Libération d'une connexion
- Contrôle de flux ($\rightarrow$ contrôle de congestion)
- Multiplexage / segmentation

Transport fiable

- Principe général : *Automatic Repeat reQuest* (ARQ) : acquittements + timeouts
- Basic : émettre et attendre (*Stop-and-wait ARQ*)



- nécessité du *numéro de séquence* → identification des données

Transport fiable

- Problème : délai entre deux paquets égal au double du temps de transmission
- Solution : envoyer plusieurs paquets successivement sans attendre d'acquittement
- Plusieurs déclinaisons :
  - *Go-Back-N ARQ* : retransmission complète à partir de la détection de perte
  - *Selective Repeat ARQ* : possibilité d'acquitter/de redemander des données sélectivement

Flux vs Congestion

- contrôle de congestion :
  - problème global du réseau pour éviter trop de trafic dans un sous-réseau
  - cas typique : réseau lent, nombreux terminaux transmettant de gros fichiers
- contrôle de flux :
  - éviter qu'un émetteur rapide ne sature un récepteur lent
  - cas typique : réseau rapide, un terminal rapide transmet à un terminal lent

Services réseaux

- Web
- Firewall
- NAT
- DNS

Web

- *World Wide Web* (toile d'araignée mondiale) ou web :
  - un des service d'internet
  - crée par Tim Berners-Lee au CERN en 1989
  - concept d'information distribuée et d'hypermédia
  - document $\longrightarrow$ référence aux autres documents
  - 1993 première interface utilisateur conviviale : Mosaic

Modèle client-serveur

Deux acteurs :

- le client (ex : firefox) : effectue des requêtes vers le serveur
- le serveur (ex : Apache) : exécute les requêtes et renvoie le résultat au client .

Rôle d'un programme client :

- traduire les ordres de l'utilisateur en messages conformes à un protocole d'échange avec un serveur
- contacter le serveur adéquat et lui passer la requête
- attendre la réponse du serveur
- mettre en forme la réponse et la présenter à l'utilisateur

Trois mécanismes

- schéma d'adressage uniforme $\rightarrow$ localiser : URLs
- protocoles $\rightarrow$ communication : HTTP
- documents hypermédias $\rightarrow$ navigation : HTML

URL : *Uniform Ressource Locator*

Les documents hypermédias sont répartis à travers le monde $\Longrightarrow$ identification et localisation de manière unique d'un document.

Format des URLs

1. nom du protocole
2. nom de la machine (i. e. serveur)
3. nom ressource

```
protocole://serveur[:port][/chemin][/fichier][#position]
```

protocole	le protocole d'échange entre le client et le serveur. Le plus souvent on utilise `http` ou `ftp`
serveur	l'adresse internet du serveur qui diffuse les documents.
port	Numéro du port
chemin	le chemin (suite de répertoires séparés par des /)
fichier	le nom du document qui nous intéresse
position	une position précise à l'intérieur du document

Protocole HTTP

- défini les échanges entre un serveur web et un client
- client $\rightarrow$ serveur :
  - `GET` demande d'une page
  - `POST` ajout à une ressource nommée
  - `DELETE` suppression d'une page
  - `PUT` rangement d'une page web
  - `HEAD` demande de l'en-tête d'une page
  - `PUT` ajout ou remplacement d'une ressource
  - ...
- serveur $\rightarrow$ client :
  - codes 500 : erreur serveur
  - codes 400 : erreur client
  - codes 300 : redirection
  - codes 200 : succès
  - codes 100 : information

Protocole HTTP

- HTTP/0.9 : ouverture, GET, réponse, fermeture
- HTTP/1.0 : utilisation d'en-têtes (`Host`, `Referer`, `Content-Type`, `Content-Length`, ...), autres méthodes, ...
- HTTP/1.1 : `Host` obligatoire, connexions persistantes
- HTTP/2.0 : compression des entêtes, push, chiffrement (non obligatoire), multiplexage
- HTTP/3.0 : transport QUIC

Dialogue HTTP

```
GET /~remm/200_ok_get_post.html HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; U; Linux i686; fr; rv:1.8.0.8) Gecko/20061115 Ubuntu/dapper-s
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,ima
Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://127.0.0.1/~remm/

HTTP/1.1 200 OK
Date: Wed, 08 Dec 2008 16:27:52 GMT
Server: Apache/2.0.55 (Ubuntu)
Last-Modified: Mon, 29 Sep 2003 20:40:13 GMT
ETag: "57a4d-3f3-e7a5c540"
Accept-Ranges: bytes
Content-Length: 1011
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<html>...
```

Dialogue HTTP : Rechargement de page

```
GET /~remm/200_ok_get_post.html HTTP/1.1
...
If-Modified-Since: Mon, 29 Sep 2003 20:40:13 GMT
If-None-Match: "57a4d-3f3-e7a5c540"
Cache-Control: max-age=0

HTTP/1.1 304 Not Modified
Date: Wed, 08 Dec 2008 16:29:33 GMT
Server: Apache/2.0.55 (Ubuntu)
Connection: Keep-Alive
Keep-Alive: timeout=15, max=100
ETag: "57a4d-3f3-e7a5c540"
```

Dialogue HTTP : Redirection

```
GET /~remm/302_redirect.cgi HTTP/1.1
...

HTTP/1.1 302 Moved
Date: Wed, 08 Dec 2008 16:31:22 GMT
Server: Apache/2.0.55 (Ubuntu)
Location: http://www.google.fr
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/plain; charset=UTF-8
```

Dialogue HTTP : Erreurs Client

```
GET /~remm/403_forbidden HTTP/1.1
...

HTTP/1.1 403 Forbidden
Date: Wed, 08 Dec 2008 16:33:05 GMT
Server: Apache/2.0.55 (Ubuntu)
Content-Length: 296
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
page erreur
...
```

Dialogue HTTP : Erreurs Client

```
GET / HTTP/1.1

HTTP/1.1 400 Bad Request
Date: Wed, 08 Dec 2008 16:33:53 GMT
Server: Apache/2.0.55 (Ubuntu)
Content-Length: 301
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.2.10 (Ubuntu) Server at 127.0.0.1 Port 80</address>
</body></html>
```

Proxy HTTP

proxy ou serveur mandataire :

- relaie des requêtes entre un client et un serveur
- fait office de cache
- aide à la sécurisation du réseau local
- journalisation des requêtes (log)

Serveurs HTTP

nombreuses implémentations du protocole HTTP

- Apache
- Nginx
- IIS (Microsoft)
- Lighttpd,
- ...

Apache

- dérivé de NCSA httpd
- modulaire → chargement de fonctionnalités supplémentaires :
  - CGI, SSI,
  - réécriture d'URL,
  - négociation de contenu,
  - protocoles de communication additionnels,
  - pages personnelles,
  - ...

Configuration serveur

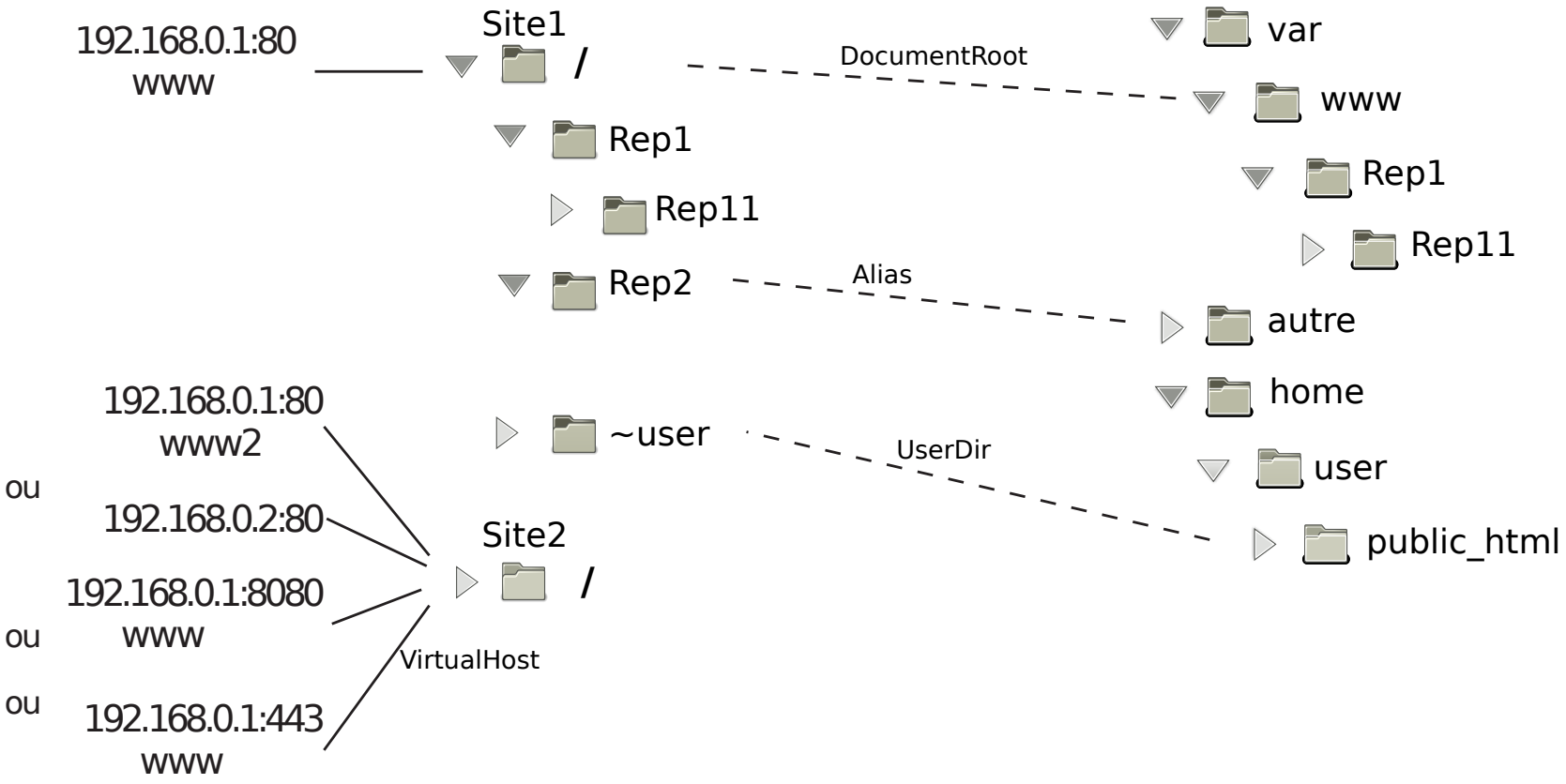


Figure – Serveur HTTP

Configuration

- fichiers de configuration :

```
/etc/apache2/
├── apache2.conf
│       └──  ports.conf
├── mods-enabled
│       ├── *.load
│       └── *.conf
├── conf-enabled
│       └── *.conf
└── sites-enabled
        └── *.conf
```

- configuration par des directives :

```
Directive valeur1 valeur2
```

ou

```
<portée>
    Directive valeur1 valeur2
</portée>
```

- portée de la configuration :
  - globale au serveur,
  - pour un serveur virtuel (`<VirtualHost>`),
  - un répertoire ou un fichier (`<Directory>`, `<Location>` et `<Files>`)
  - ou définie par un utilisateur pour un répertoire → `.htaccess`

Configuration serveur (base)

```
# sockets d'écoutes
Listen 192.168.0.1:80
# nom du serveur par défaut
ServerName www.fai.com
#
DocumentRoot "/Site1/"
# utilisateur non-privilégié
User apache2
Group apache2

<Directory "/Site1/">
# options parmi :
#   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI
    Options Indexes FollowSymLinks
# voir plus loin -> .htaccess
    AllowOverride None
# droits d'accès
    Require all granted
</Directory>
```

Configuration serveur (hôtes virtuels)

```
# différentiation sur nom/ip/port -> sites différents
<VirtualHost 192.168.0.1:80>
    DocumentRoot /Site1/
    ServerName www.fai.com
</VirtualHost>

<VirtualHost 192.168.0.1:80>
    DocumentRoot /Site2/
    ServerName www2.fai.com
</VirtualHost>

<VirtualHost 192.168.0.2:*>
    DocumentRoot /Site2/
</VirtualHost>

<VirtualHost 192.168.0.1:8080>
    DocumentRoot /Site2/
    ServerName www.fai.com
</VirtualHost>
```

Configuration serveur (listing)

```
# fichiers du répertoire à charger par défaut (dans l'ordre)
DirectoryIndex index.html index.html.var default.html

# voir plus loin -> .htaccess
AccessFileName .htaccess

<Files ~ "^\.ht">
    Require all denied
</Files>

IndexOptions FancyIndexing VersionSort
ReadmeName README.html
HeaderName HEADER.html
IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t
```

Configuration serveur (logs)

```
ErrorLog logs/error_log

# debug, info, notice, warn, error, crit, alert, emerg.
LogLevel warn

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

CustomLog logs/access_log common
```

Configuration serveur (alias)

```
Alias /Rep2 "/autre/"

<Directory "/autre/">
    Options Indexes MultiViews
    AllowOverride None
    Require all granted
</Directory>

ScriptAlias /cgi-bin/ "/opt/apache2/cgi-bin/"
<Directory "/opt/apache2/cgi-bin">
    AllowOverride None
    Options None
    Require all granted
</Directory>

Redirect permanent /essai http://www.fai.com
```

Configuration serveur (modules)

- chargement d'un module

```
LoadModule negotiation_module /usr/lib/apache2/modules/mod_negotiation.so
```

- évaluation conditionnelle des directive

```
<IfModule mod_negotiation.c>
    LanguagePriority en ca cs da de el eo es et fr
</IfModule>
```

Configuration serveur (public_html)

```
<IfModule mod_userdir.c>
UserDir public_html

  <Directory /home/*/public_html>
    AllowOverride FileInfo AuthConfig Limit Indexes
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    <Limit GET POST OPTIONS PROPFIND>
      Require all granted
    </Limit>
    <LimitExcept GET POST OPTIONS>
      Require all denied
    </LimitExcept>
  </Directory>
</IfModule>
```

Configuration serveur (erreurs)

```
Alias /error/ "/opt/apache2/error/"
<Directory "/opt/apache2/error">
    AllowOverride None
    Options IncludesNoExec
    AddOutputFilter Includes html
    AddHandler type-map var
    Order allow,deny
    Allow from all
    LanguagePriority en fr de es it nl sv
    ForceLanguagePriority Prefer Fallback
</Directory>
ErrorDocument 400 /error/HTTP_BAD_REQUEST.html.var
ErrorDocument 401 /error/HTTP_UNAUTHORIZED.html.var
ErrorDocument 403 /error/HTTP_FORBIDDEN.html.var
ErrorDocument 404 /error/HTTP_NOT_FOUND.html.var
ErrorDocument 405 /error/HTTP_METHOD_NOT_ALLOWED.html.var
ErrorDocument 408 /error/HTTP_REQUEST_TIME_OUT.html.var
ErrorDocument 410 /error/HTTP_GONE.html.var
ErrorDocument 411 /error/HTTP_LENGTH_REQUIRED.html.var
ErrorDocument 412 /error/HTTP_PRECONDITION_FAILED.html.var
ErrorDocument 413 /error/HTTP_REQUEST_ENTITY_TOO_LARGE.html.var
ErrorDocument 414 /error/HTTP_REQUEST_URI_TOO_LARGE.html.var
ErrorDocument 415 /error/HTTP_SERVICE_UNAVAILABLE.html.var
ErrorDocument 500 /error/HTTP_INTERNAL_SERVER_ERROR.html.var
ErrorDocument 501 /error/HTTP_NOT_IMPLEMENTED.html.var
ErrorDocument 502 /error/HTTP_BAD_GATEWAY.html.var
ErrorDocument 503 /error/HTTP_SERVICE_UNAVAILABLE.html.var
ErrorDocument 506 /error/HTTP_VARIANT_ALSO_VARIES.html.var
```

.htaccess

- permet à un utilisateur de (re)définir des directives pour son (sous)site
- se comporte comme une directive `<Directory>`
- soumis à autorisation : `AllowOverride option`
- parmi : `AuthConfig`, `FileInfo Indexes`, `Limit`, ...
- le nom du fichier peut être redéfini : `AccessFileName`
- couteux en temps de calcul

MIME

- à l'origine un courrier ne comportait que du texte (ASCII)
- besoin d'envoyer des caractères non ASCII (accentués), des contenus non textuels, des contenus multiples
- définition d'entêtes supplémentaires :
  - `MIME-Version` : encore `1.0` pour "annoncer" la suite
  - `Content-Type` : 7 types principaux : `text`, `image`, `audio`, `video`, `application`, `multipart`, `message` et des sous-types (d'où `text/plain`, `image/jpeg`, `multipart/form-data`,...).
  - `Content-Transfert-Encoding` : `Quoted-Printable`, `Base64`, `7bit`