

Information, Calcul et Communication

Module 3 : Systèmes

Leçon III.2 : Stockage et transmission de l'information

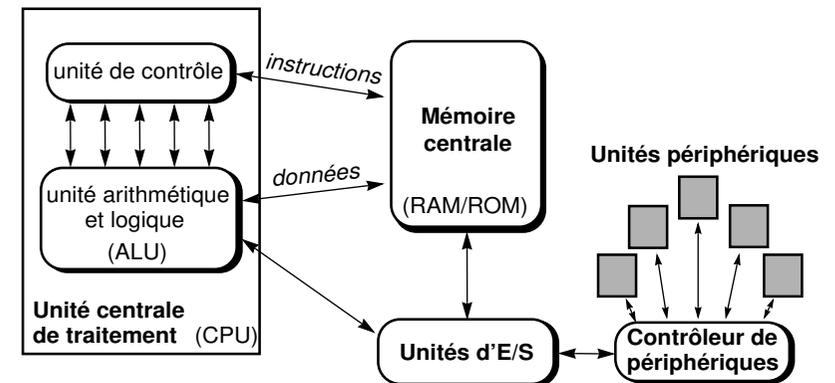
préparée par Prs. Ph. Janson, W. Zwaenepoel
& A. Ailamaki

Objectifs du cours d'aujourd'hui

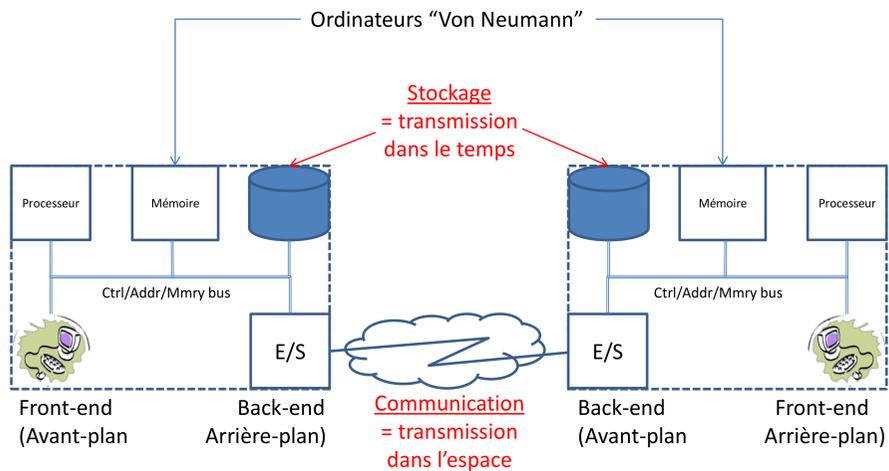
Après quelques notions sur l'architecture des ordinateurs « de Von Neumann », cette deuxième leçon porte sur :

- ▶ la structure des mémoires de masse (mémoires de stockage, mémoires rémanentes)
- ▶ la structure des communications réseau

Architecture de Von Neumann (1955)



Stockage et communication des données



But de cette leçon

Répondre à une seule et même question dans deux cas :

- ▶ Dans le cas du stockage :
Où et comment **stocker** des données de façon à pouvoir les **retrouver plus tard** ?
- ▶ Dans le cas des réseaux :
Quand et comment **envoyer** des données de façon à pouvoir les **recevoir à distance** ?
- ▶ stockage : communication dans le temps au moyen de l'espace
- ▶ télécommunication (réseaux) : communication dans l'espace en utilisant du temps

Plan

- ▶ Le **besoin de structure** dans les données
- ▶ Stockage :
 - ▶ **Types de structures** de stockage : séquentielles, hiérarchiques, relationnelles, ...
 - ▶ **Identification, localisation, et accès** à des données stockées
- ▶ Réseaux :
 - ▶ **Types de structures** de transmission : couches et encapsulation de protocoles
 - ▶ **Identification, localisation, et accès** à des données en réseau

Le stockage : pourquoi ?

Au départ du problème : contraintes technologiques :

- ▶ La mémoire centrale est
 - ▶ Trop petite (quelque Go)
 - ▶ Volatile (tout est perdu quand on éteint)
 - ▶ Trop chère
 - ▶ Le stockage est donc nécessaire
 - ▶ pour retenir *toutes* les données
 - ▶ à long terme
 - ▶ et à coût honorable
- 👉 **Où** et **comment** stocker des données de façon à les **retrouver** ?

Contraintes technologiques

| | Latence | Débit | Coût (\$/Go) | Capacité | Rétention | Accès |
|--------------------|--------------------|-----------|---------------------|---------------------|-----------|----------------------|
| RAM | 1 - 100 ns | Go/s | 10 | Mo - Go | NON | Aléatoire |
| Flash | µs | Go/s | 0.5 | Go - To | Oui | Aléatoire |
| Disques | ms | 100s Mo/s | 0.05 | > To | Oui | Aléatoire avec délai |
| Bandes magnétiques | Encore plus lent ! | 100s Mo/s | Encore moins cher ! | Encore plus grand ! | Oui | Séquentiel |

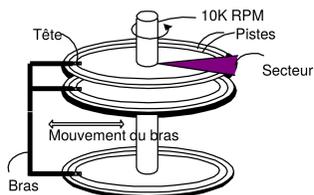
Flash

Accès aléatoire comme RAM mais par pages comme HDD



HDD

Latence de rotation + positionnement du bras



Bandes

Accès strictement séquentiel => latence de déroulement



Le besoin de structure dans le stockage de données

Imaginons un disque (ou autre support) sans aucune structure :



Comment y retrouver une information qu'on cherche ?

- Même la «recherche non structurée» (e.g. Google) a besoin de structure (à elle, qu'elle crée) pour retrouver ce qu'on lui demande !

Le besoin de structure dans la transmission de données

- Imaginons un signal arrivant d'un réseau, sans aucune structure



- Comment savoir si c'est un signal ou juste du bruit ?
- Comment savoir où il commence et où il finit ?
- Comment savoir d'où il vient ?
- Comment interpréter ce qu'il contient ?

Le besoin de structure dans le stockage de données (1/2)

Données non-structurées
(= désordonnées, sans aucun ordre particulier)

- Facile à gérer, stocker, transporter
- Plus difficile à exploiter, explorer, interpréter

Exemple : dans l'océan de données non-structurées que Google tente d'indexer sur la toile, retrouver des informations sur le *professeur de Statistiques Michael Jordan* est un défi.



Le besoin de structure dans le stockage de données (2/2)

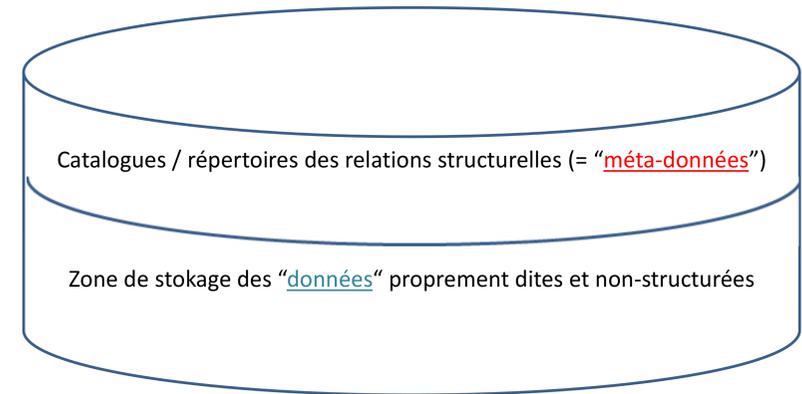
Données structurées
(= ordonnées en listes, piles, hiérarchies, tables, etc.)

- ▶ Facile à exploiter, explorer, interpréter
- ▶ Plus difficile à gérer, stocker, transporter

Exemple : retrouver des informations sur le *professeur Michael Jordan* dans les bases de données de son université est trivial.



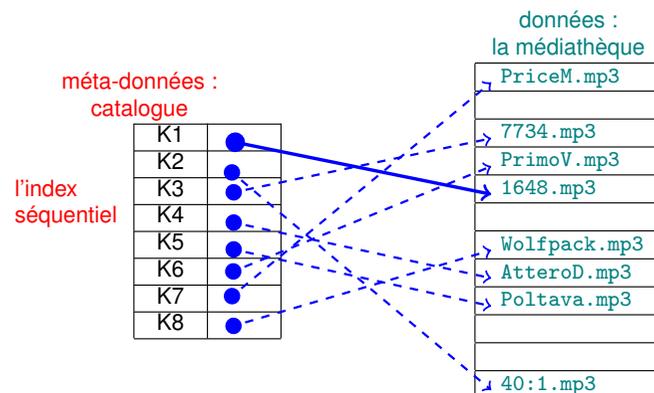
Principe de base de la structuration des données stockées



Le catalogue d'une médiathèque

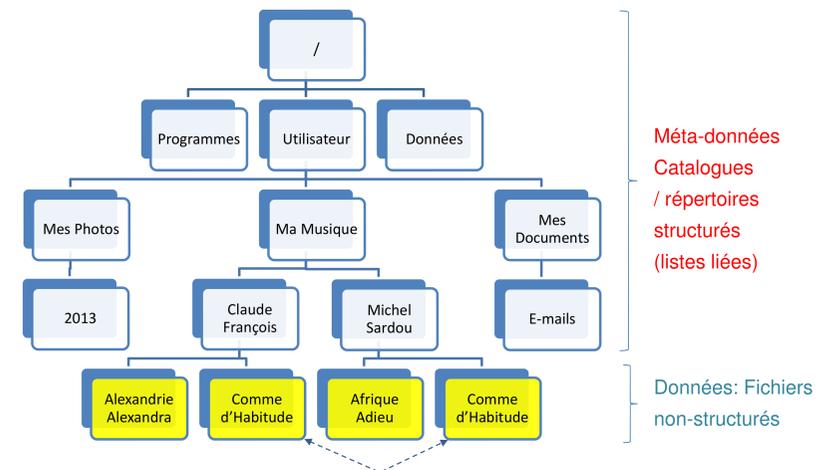
Catalogue :
indexé (= structuré) par ordre al-
phabétique (par exemple des titres)

Médiathèque :
rangée par ordre d'acquisition
rangée selon la place disponible



⚠ Mais il est difficile de retrouver dans cette médiathèque toutes les œuvres d'un interprète

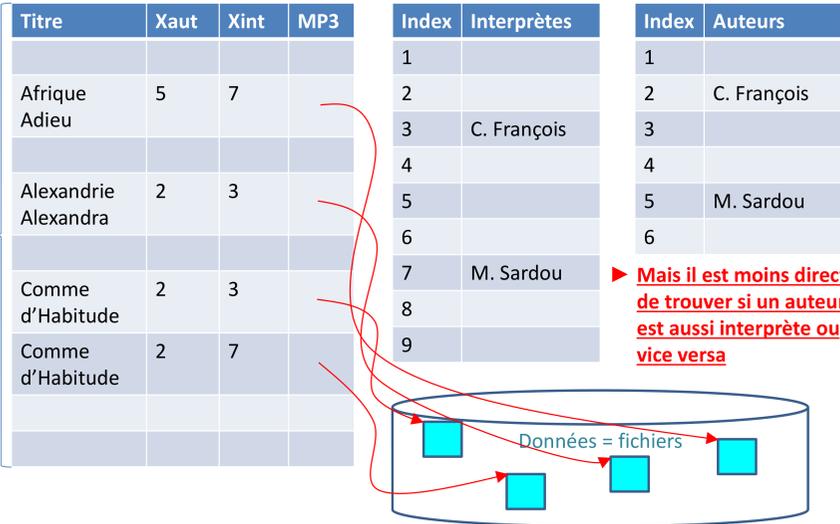
Une médiathèque hiérarchique



⚠ Mais il est difficile de retrouver dans cette médiathèque tous les interprètes d'une œuvre donnée

Une médiathèque relationnelle

Méta-données = Catalogues / répertoires tabulaires



Accès à l'information

Le but de toutes ces métadonnées est de permettre l'accès à l'information :

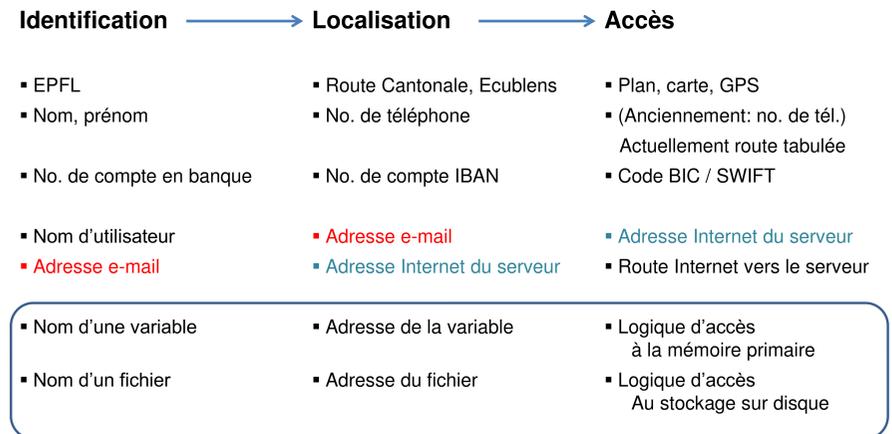
- ▶ **l'identifier** :
 - ▶ comprendre les besoins, les requêtes possibles, ...
 - ▶ l'information elle-même (quelle est elle ? ☞ besoins spécifiques)
- ▶ **retrouver** l'information identifiée :
 - ▶ la **localiser**
 - ▶ y **accéder**

Identification, localisation, et accès à des données stockées

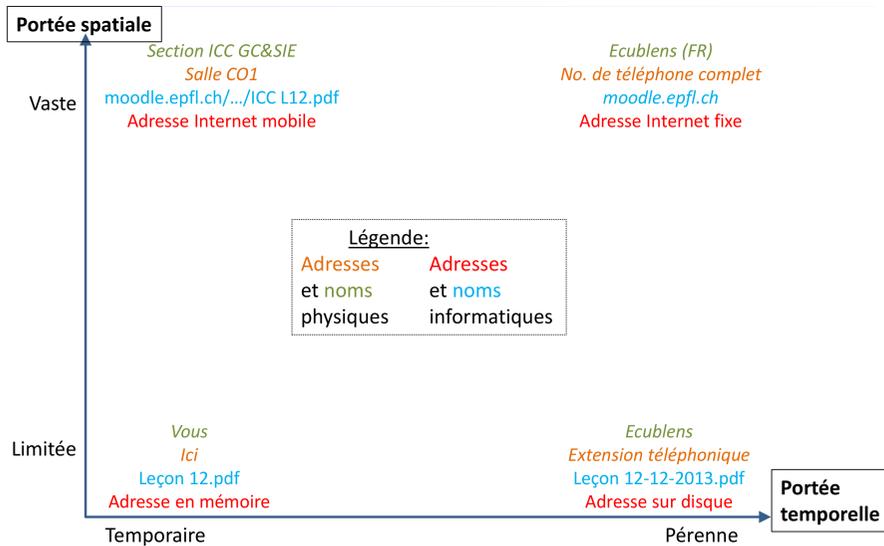
- ▶ Structurée ou non, une information n'a de sens que dans son contexte spatio-temporel

Paris ne signifie pas la même chose à Troie (France) en 2013 qu'à Troie (Turquie) en 1250 avant J.C.
- ▶ Pour **identifier**/distinguer une information parmi d'autres, il faut lui donner un nom/**identificateur**
Un répertoire/catalogue est alors nécessaire pour **localiser** cette information
- ▶ La localisation de cette information est dénotée par son **adresse** (une place qui lui a été attribuée)
Un mécanisme d'acheminement est alors nécessaire pour **accéder** à cette adresse
- ▶ Pour accéder à cette adresse il faut déterminer une **route** jusque là

Exemples



Portée spatio-temporelle des noms et adresses



Légende:
 Adresses et noms physiques (green)
 Adresses et noms informatiques (red)

Résolution des noms et adresses

Vu le nombre de façons de faire référence à des informations :

- ▶ Noms ou adresses
- ▶ Universels ou locaux
- ▶ Permanents ou temporaires

les ordinateurs doivent régulièrement traduire une forme en une autre.

- ☞ C'est un des rôles essentiels des *logiciels systèmes* :
- ▶ Les compilateurs traduisent ainsi les noms de variables en adresses mémoire
- ▶ Les systèmes de fichiers et les bases de données gèrent le mouvement des données entre mémoire primaire et secondaire y compris la traduction de noms et adresses « secondaires » (universels et permanents) en équivalents « primaires » (locaux et temporaires)

Exemple : compilation de noms de variables en adresses

Identification par nom

- ▶ E.g. nom, prénom
- ▶ Programme source
 $\text{delta} \leftarrow b^2 - 4 \cdot a \cdot c$

| | |
|-------|--------|
| a | @a |
| b | @b |
| c | @c |
| (x) | @x |
| delta | @delta |

Table des symboles du compilateur

Localisation par adresse

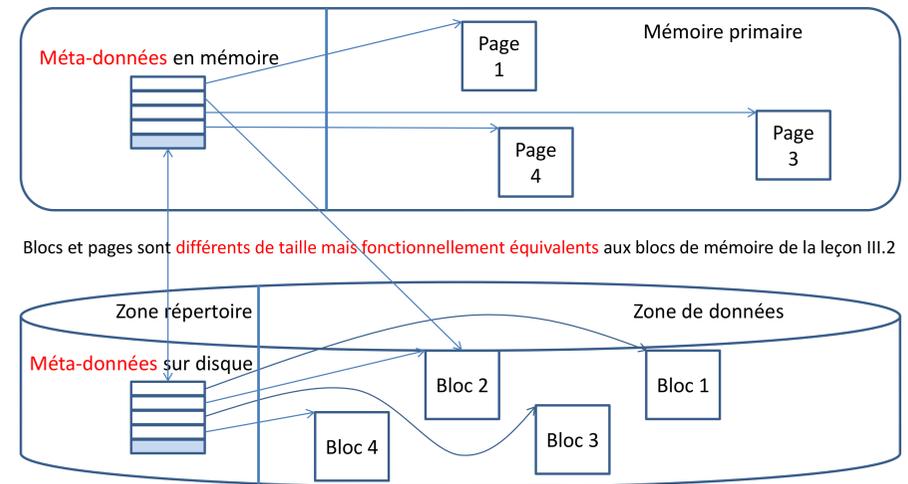
- ▶ No. de téléphone
 - ▶ Programme objet
- charge R,@c
 mult R,@a
 décale R,2
 écris R,@x
 charge R,@b
 mult R,@b
 diff R,@x
 écris R,@delta

Routage de l'appel
 Mémoire primaire

| |
|-------|
| a |
| b |
| c |
| x |
| delta |

Logique d'adressage de la mémoire

Exemple : gestion de noms et adresses primaires et secondaires



Blocs et pages sont différents de taille mais fonctionnellement équivalents aux blocs de mémoire de la leçon III.2

Notion de protocole de communication

Protocole : jeu de règles qui gouverne une communication

Toute communication est gouvernée par un protocole y compris, la communication entre êtres humains

- ▶ en cours
- ▶ au téléphone
- ▶ ...

Un protocole entre êtres humains peut être vaguement défini (quoi que...)

Un protocole entre ordinateurs doit être fixé dans tous les détails

Notion de protocole de communication : exemples

- ▶ Où commence et où finit une communication ?
délimitée par deux silences
- ▶ D'où vient-elle et à qui s'adresse-t-elle ?
de l'orateur au public
- ▶ Dans quel langue est-elle exprimée ?
un langage commun aux interlocuteurs
- ▶ Que faire si la communication est perturbée ?
demande de répétition : « Pardon ? »
- ▶ A qui le tour de communiquer ?
à chacun son tour

Structuration par couches (abstraction des protocoles)

- ☞ Chaque couche gère et abstrait les phénomènes de son niveau pour affranchir les autres couches de ces détails

Exemple d'une conversation téléphonique :

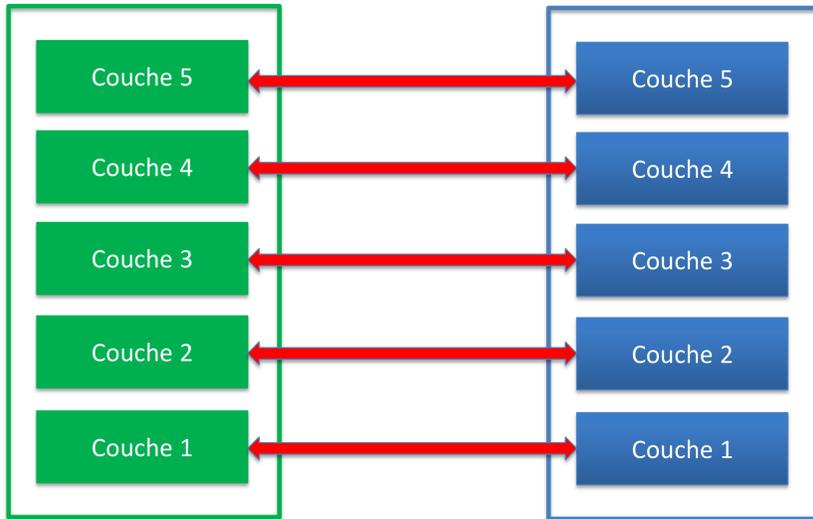


Les couches de l'Internet

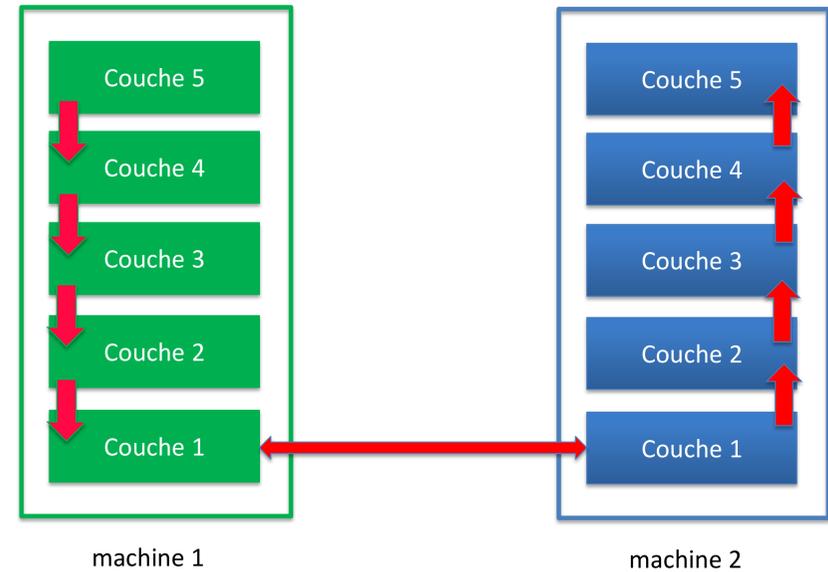
| | | | | | | |
|----------------|-----------------------------|-------------------------------|----------------------------|----------------------------|-----------------------|-----------|
| 5. Application | Terminal interactif ex. SSH | Transfert de fichiers ex. FTP | Courrier électronique SMTP | Naviguer sur la toile HTTP | Le botin Internet DNS | Etc. |
| 4. Transport | TCP | | SSL / TLS | | UDP | |
| 3. Réseau | IP (adressage et routage) | | | | | |
| 2. Lien | CSMA / CD | | | PPP | Trunk lines | |
| 1. Physique | Wi-Fi | Ethernet | CATV | ADSL | Trunk lines | |

DNS – le bottin de l'Internet, protocole de résolution de noms en adresses
 SSH – protocoles de terminaux à distance
 FTP, NFS – protocoles de transfert de fichiers
 SMTP, POP, IMAP – protocoles de courrier électronique
 HTTP – LE protocole de la toile (web, Facebook, Twitter, Google, etc.)

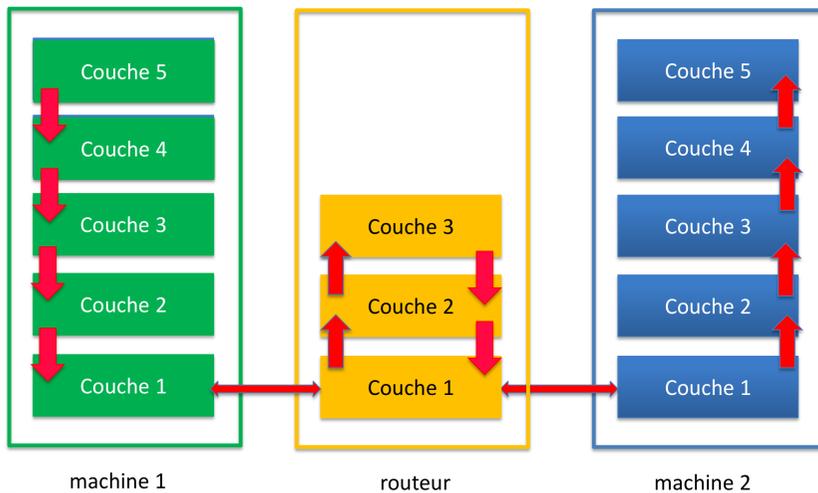
Couches : communication aux niveaux « logiques »



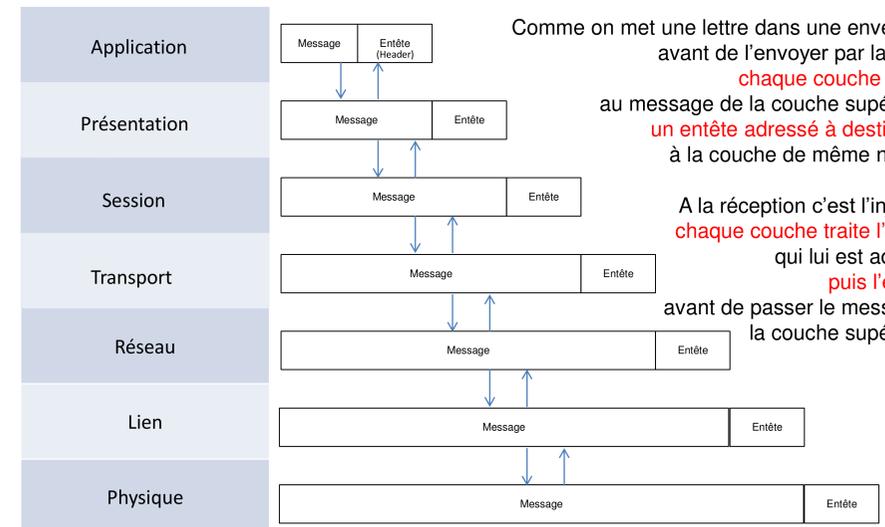
Couches : communication physique simple



Couches : communication physique avec relais



La notion d'encapsulation



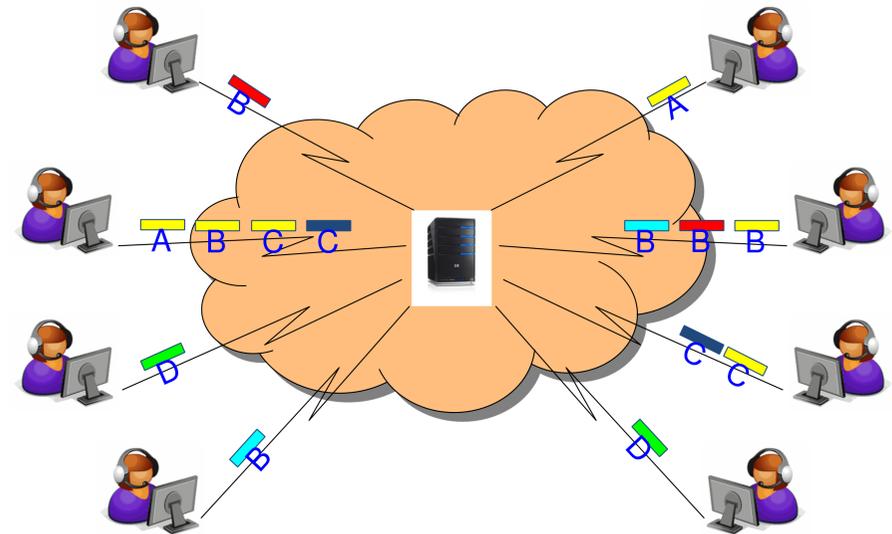
Comme on met une lettre dans une enveloppe avant de l'envoyer par la poste **chaque couche ajoute un entête adressé à destination** à la couche de même niveau.

A la réception c'est l'inverse: **chaque couche traite l'entête** qui lui est adressé **puis l'enlève** avant de passer le message à la couche supérieure

Opérations d'un protocole

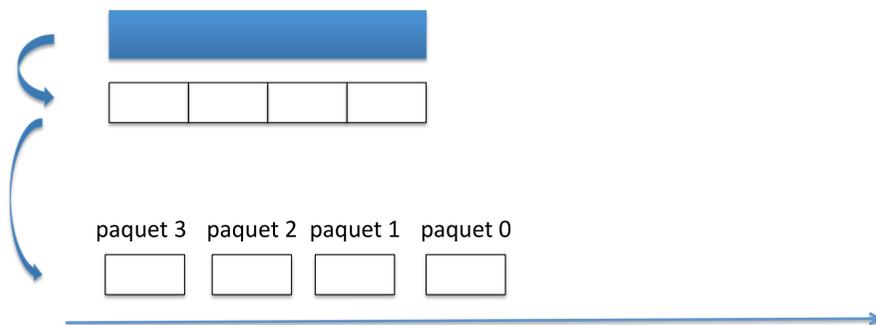
- ▶ A la *transmission* d'un paquet : le protocole
 - ▶ reçoit des données
 - ▶ ajoute un entête
 - ▶ envoie le paquet
- ▶ A la *réception* d'un paquet : le protocole
 - ▶ reçoit le paquet
 - ▶ supprime l'entête (en le vérifiant)
 - ▶ passe les données

Couche 3 – Routage Internet par commutation de paquets



Envoi d'un fichier par paquets

fichier (+ entêtes couches supérieures)



Commutation de paquets

Note :

- ▶ des paquets peuvent être perdus en route
- ▶ des paquets peuvent arriver en désordre

Fonctionnalités :

- ▶ envoyer des paquets
- ▶ savoir s'ils sont arrivés :
le destinataire envoie un paquet d'acquittement
- ▶ pouvoir les remettre dans l'ordre
(et ne pas traiter 2 fois le même paquet)



Commutation de paquets

| type | seq | données |
|------|-----|---------|
|------|-----|---------|

- ▶ type : données ou « paquet d'acquittement »
- ▶ seq : 0,1,2,... identifiant de paquet

Si l'expéditeur ne reçoit pas son paquet d'acquittement au bout d'un certain temps

- ▶ il retransmet
- ▶ un certain nombre de fois puis abandonne

« seq. » permet au receveur de remettre les paquets dans l'ordre et de savoir si ce sont des retransmissions ou non

« type », « seq. » sont un exemple de **meta-données**

On appelle ces méta-données (y compris d'autres : taille, source, destinataire, code correcteur d'erreurs, ...) l'« *entête* » du paquet

Couche 3 : adressage IP (Internet Protocol)

IPv4 : 32 bits ne peuvent adresser que 2^{32} ($\approx 4 \cdot 10^9$) systèmes écrits a . b . c . d (4 fois 8 bits, en décimal) ; Exemple : 192 . 168 . 1 . 1 interprétés comme R . A
= un abonné A (de 24 à 8 bits) sur un réseau R (de 8 à 24 bits)

IPv6 : 128 bits peuvent adresser $\approx 2 \cdot 10^{38}$ systèmes écrits XXXX : XXXX : XXXX : XXXX : XXXX : XXXX : XXXX (8 groupes de 16 bits, en hexadécimal)

Exemple : 2001:db8:0:85a3:0:0:ac1f:8001

Couche 3 : routage IP (1/2)

Contrairement aux réseaux téléphoniques : **décentralisation totale**, aucune autorité

Les nœuds de commutation n'ont *aucune notion* ni connaissance de connexions

Le routage se fait par « ouï-dire » mais résulte en un calcul distribué de « plus courts chemins » :

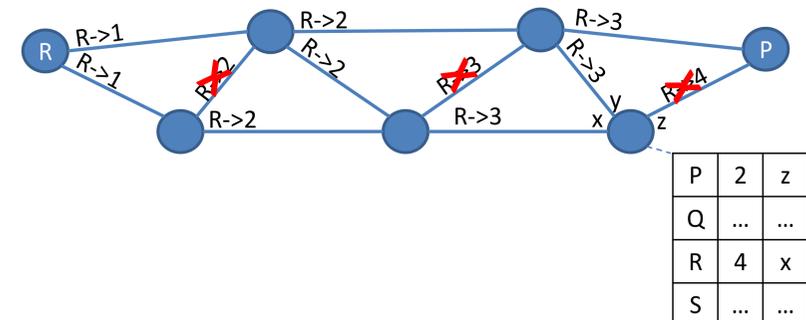
- ▶ chaque nœud annonce à ses voisins la « longueur » des chemins vers les adresses qu'il connaît
- ▶ chaque nœud retient et propage le chemin le plus court parmi ceux annoncés par ses voisins

(cf transp. suivant)

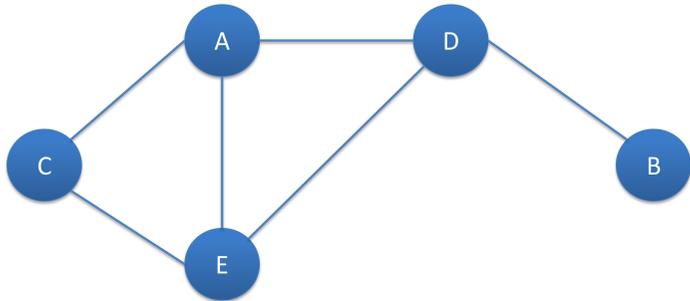
Couche 3 : routage IP (2/2)

Le routage se fait par un calcul distribué de « plus courts chemins » :

- ▶ chaque nœud annonce à ses voisins la « longueur » des chemins vers les adresses qu'il connaît
- ▶ chaque nœud retient et propage le chemin le plus court parmi ceux annoncés par ses voisins

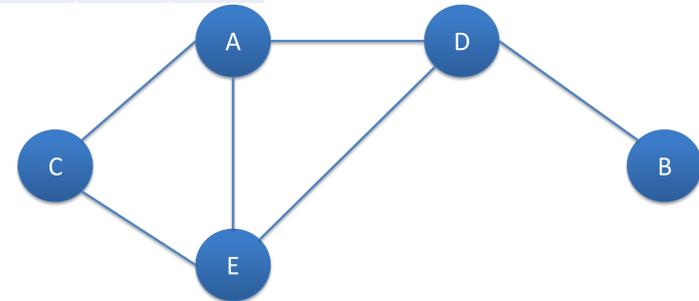


routing IP : exemple



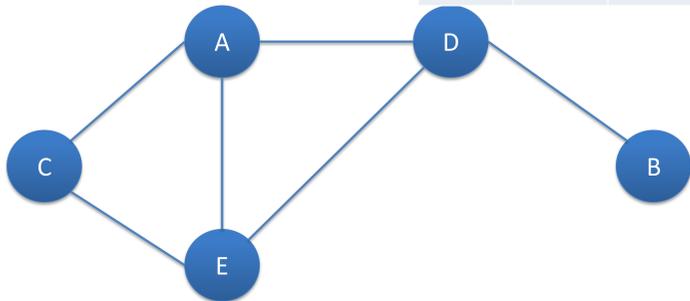
Exemple de table de routage IP (A)

| dest | chemin | distance |
|------|--------|----------|
| B | D | 2 |
| C | C | 1 |
| D | D | 1 |
| E | E | 1 |



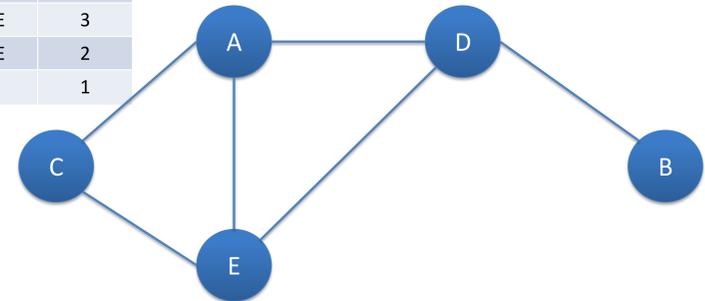
Exemple de table de routage IP (D)

| dest | chemin | distance |
|------|--------|----------|
| A | A | 1 |
| B | B | 1 |
| C | A/E | 2 |
| E | E | 1 |



Exemple de table de routage IP (C)

| dest | chemin | distance |
|------|--------|----------|
| A | A | 1 |
| B | A/E | 3 |
| D | A/E | 2 |
| E | E | 1 |



Tables de routage en pratique

NOTE : en pratique, il est clair que chaque nœud ne stocke pas explicitement toutes les routes vers tous les autres nœuds du réseau. C'est ce que nous ferons dans ce cours pour les petits réseaux utilisés en exemple/exercices, mais en pratique cela serait bien trop grand/coûteux.

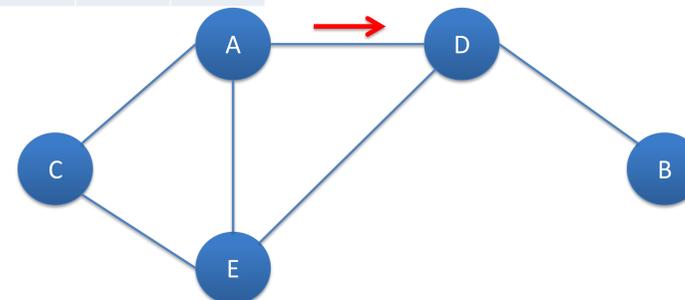
En pratique, les « noms » (adresses IP) des nœuds suivent une hiérarchie qui fait que l'on peut :

1. déléguer la gestion des nœuds plus lointains dans la hiérarchie à un nœud particulier de plus haut niveau : la « gateway »
2. décrire de façon plus compacte la route vers des nœuds proches dans la hiérarchie.

Mais cela sort du cadre de ce cours d'introduction.

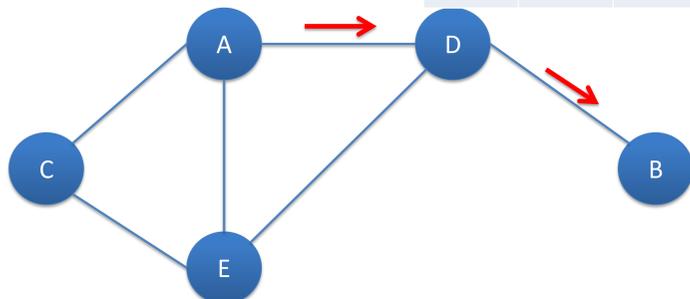
Routage IP de A à B (1/2)

| dest | chemin | distance |
|------|--------|----------|
| B | D | 2 |
| C | C | 1 |
| D | D | 1 |
| E | E | 1 |

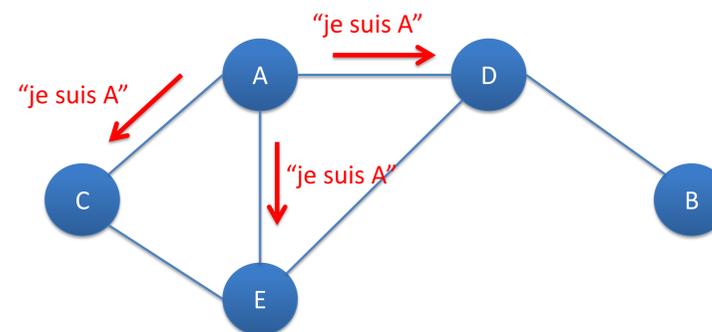


Routage IP de A à B (2/2)

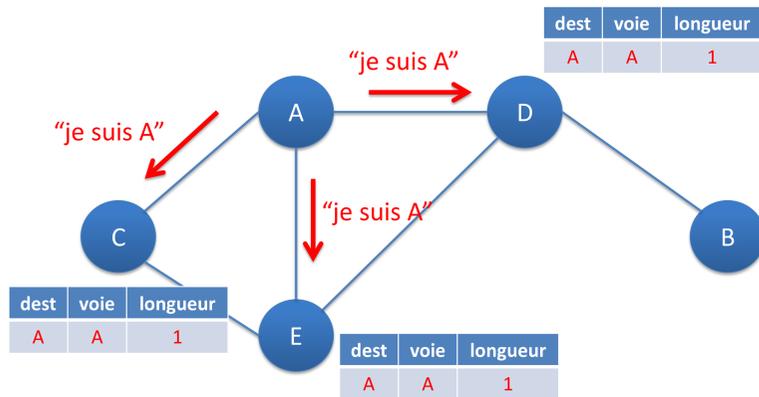
| dest | voie | longeur |
|------|------|---------|
| A | A | 1 |
| B | B | 1 |
| C | A/E | 2 |
| E | E | 1 |



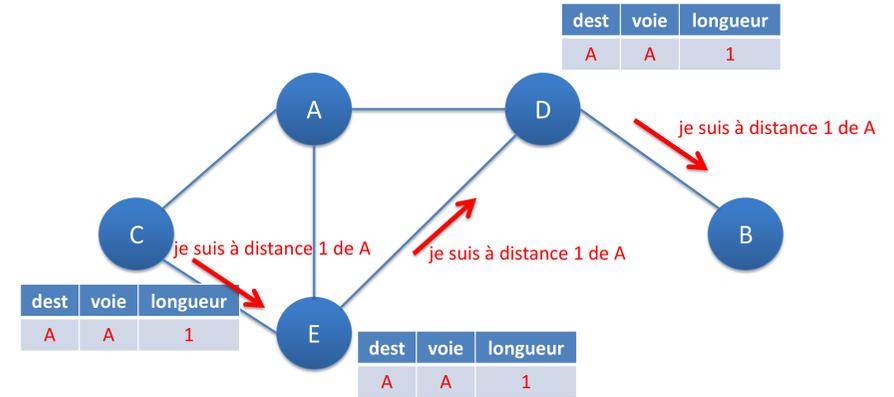
Exemple de calcul de table de routage (1/4)



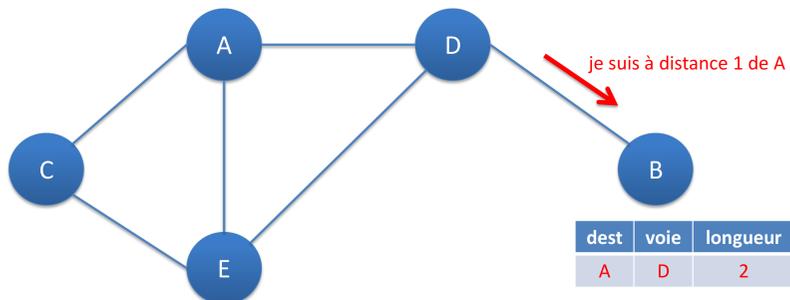
Exemple de calcul de table de routage (2/4)



Exemple de calcul de table de routage (3/4)



Exemple de calcul de table de routage (4/4)



Couche 4 : transport – TCP (Transport Control Protocol)

TCP :

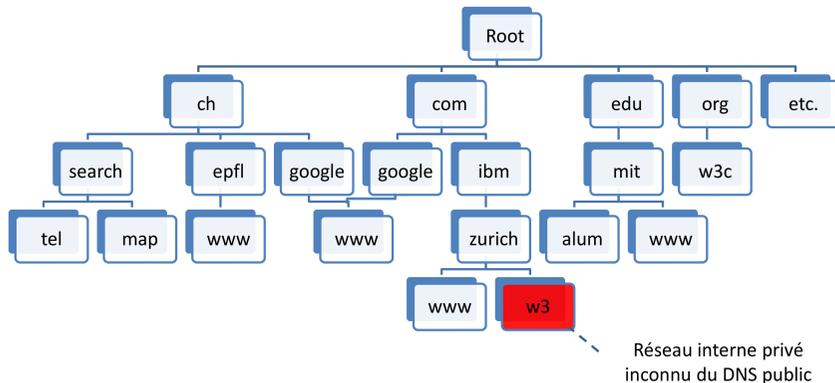
- ▶ protocole de programme à programme (« processus »)
- ▶ flux d'octets fiable :
 - ▶ Adresses des partenaires
 - ▶ Nombre et volume des messages envoyés et reçus
 - ▶ Réordonnancement et réassemblage des messages
 - ▶ Détection d'erreurs et retransmissions

SSL (Secure Session Layer) / TLS (Transport Layer Security) :
une version de TCP *sécurisée cryptographiquement* (prochaine leçon)

Couche 5 : p.ex. DNS (Domain Name System)

DNS traduit les noms en adresses, p.ex. `www.epfl.ch` en `128.178.50.12` (adresse IPv4) ou `2001:620:618:1a6:1:80b2:a60a:2` (adresse IPv6)

L'assignation de noms est **hiérarchique** mais également totalement **décentralisée**



Couche 5 : p.ex. le Web (HTTP)

HTTP (Hyper-Text Transfer Protocol)
Permet de manipuler des « ressources » au moyen de 8 messages différents dont 4 principaux :

- ▶ POST : crée - ou ajoute une information à - une ressource
- ▶ PUT : crée ou met à jour une ressource
- ▶ GET : lit et renvoie le contenu d'une ressource
- ▶ DELETE : élimine une ressource

Ces ressources sont désignées par des URI (Universal Resource Identifier)

`http:// hôte [: port] / [chemin-arborescent [? requête]]`

Le contenu des messages est exprimé en format HTML (Hyper-Text Markup Language)

HTTPS indique l'usage de HTTP sur SSL (Secure Session Layer) / TLS (Transport Layer Security) versions sécurisées cryptographiquement (v. prochaine leçon)

Résumé

- ▶ Architecture « de Von Neumann »
est une abstraction : la réalité est bien plus complexe
↳ contraintes technologiques « opposées » (capacité/vitesse)
- ▶ Stokage / Réseaux : 2 facettes (espace / temps) d'un même problème
- ▶ Besoin de structuration de l'information, des données (stockées/transmises)
 - ▶ **stockage** : les données peuvent être organisées logiquement de différentes façons : séquentielle, hiérarchique, relationnelle, ...
Le choix dépend des besoins (quel accès à quelle information ?)
 - ▶ **réseaux** : modèles en couches
- ▶ 3 besoins : identifier, localiser et accéder à (l'information)
et 3 moyens : noms, adresses, routes
- ▶ Identifiants : locaux / globaux, temporaires / permanents
- ▶ Un exemple concret : le routage IP (plus court chemin, distribué)