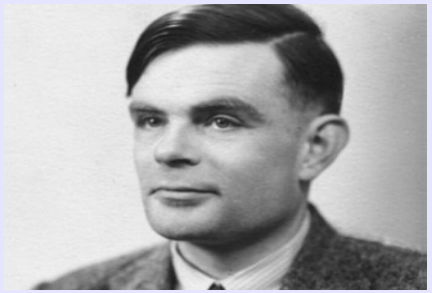
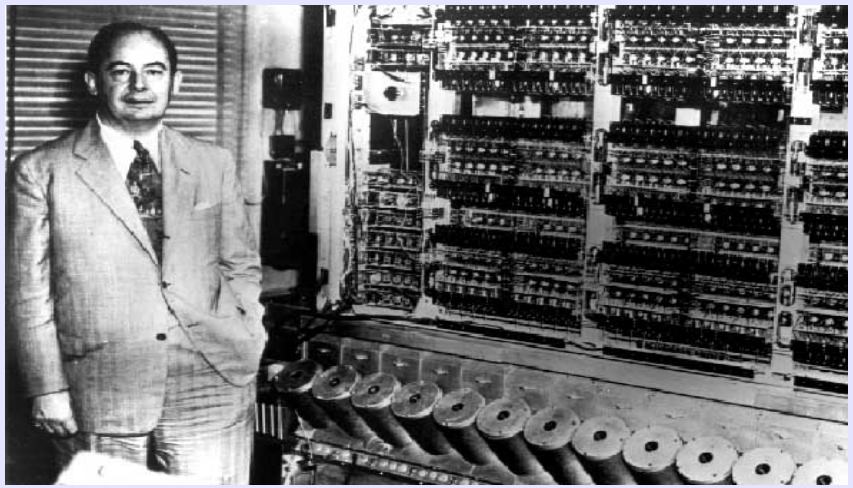




Information, Calcul et Communication

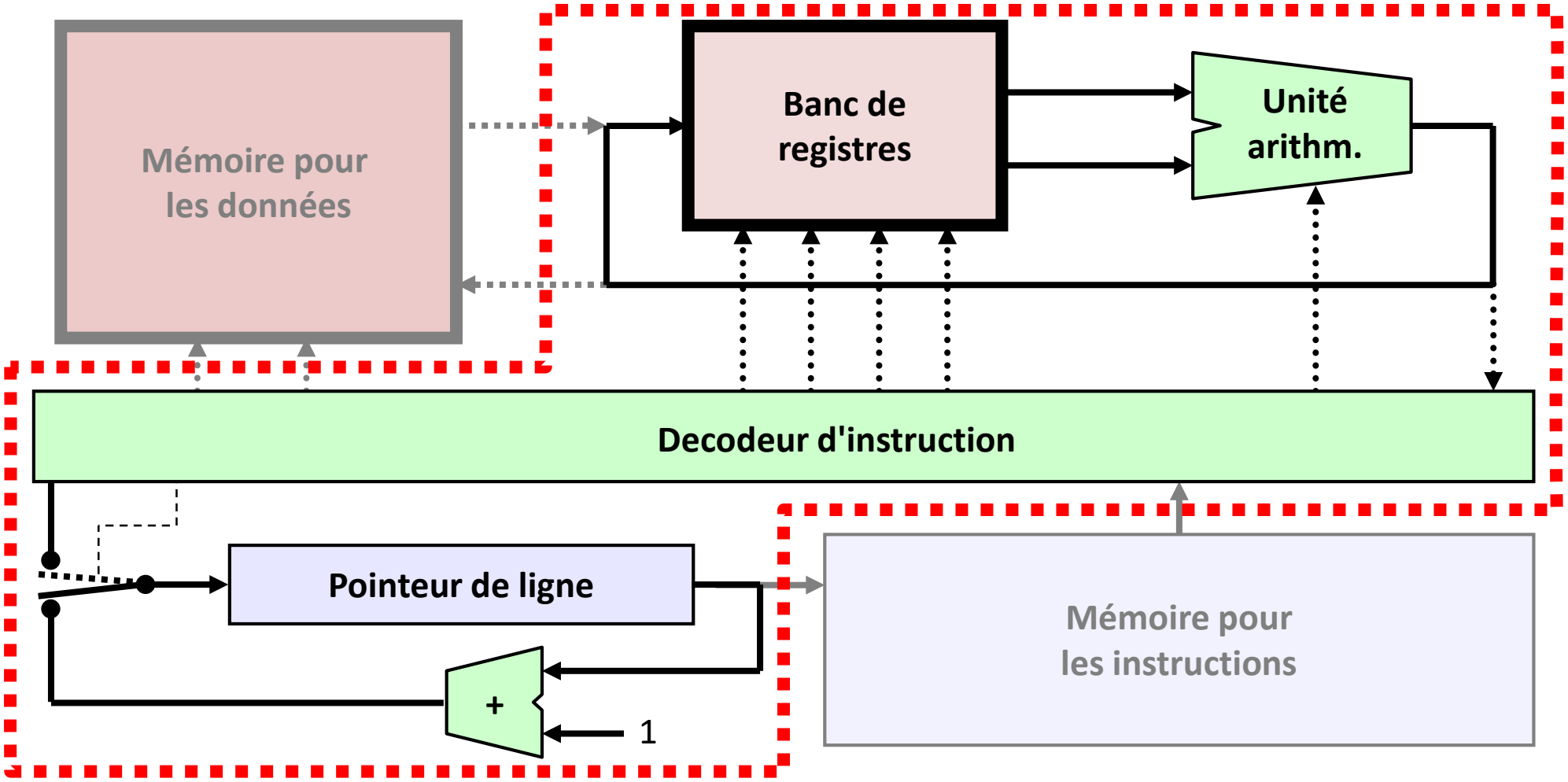
Module 3 : Systèmes

Hiérarchie de mémoire & stockage

Faculté Informatique et Communications

B. Falsafi, A. Schiper, W. Zwaenepoel

Rappel : architecture du processeur et mémoire des données (registres + …)



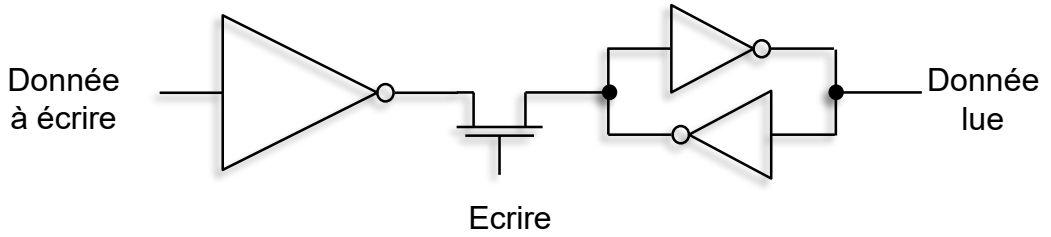
Plan

- **Introduction : mémorisation et stockage**
- Principe du cache: une mémoire grande et rapide
- Fonctionnement du cache avec le processeur et la mémoire centrale
- Le compromise entre localité spatiale et localité temporelle
- Impact de l'organisation des données sur les performances d'un algorithme
- Structuration du stockage des données

La mémoire idéale n'existe pas ...

La technologie des registres (mémoire on-chip) est coûteuse en espace et en énergie

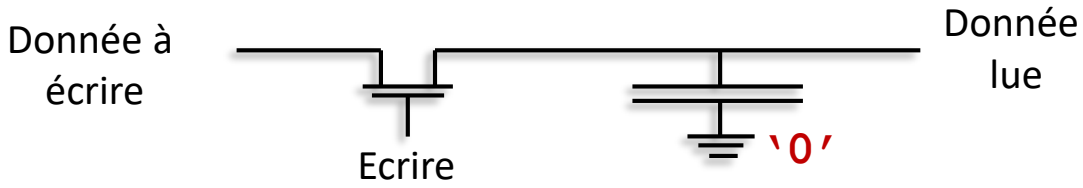
1 bit =



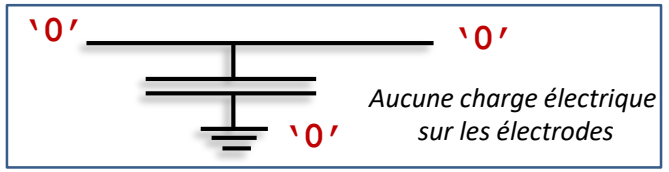
Chaque inverseur utilise deux transitors

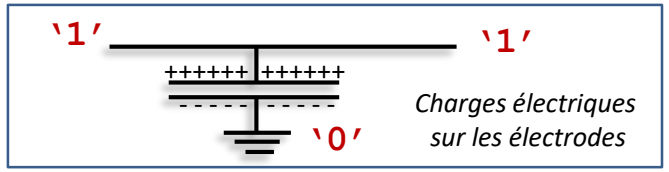
La technologie de la mémoire centrale (off-chip) demande beaucoup moins d'espace et d'énergie

1 bit =



Utilisation d'un élément appelé **condensateur** = deux électrodes séparées par un isolant.





Les électrodes peuvent accumuler des charges électriques et ainsi mémoriser les niveaux de tension 0 et 1 en sortie.
Demande beaucoup moins d'énergie que l'approche on-chip
MAIS demande 100ns pour lire/écrire

Caractéristiques de la mémoire: *capacité* et *temps d'accès*

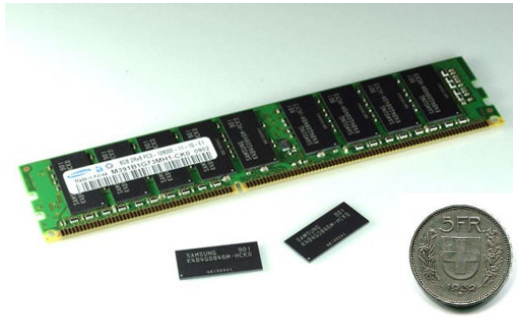
- **Compromis entre Capacité** (en octets / *Bytes, MB, GB...*) et **Temps d'accès** (en s, ns...)
- ***Stockage rapide, mais de petite taille, temporaire***
  - Mémoires temporaires
  - Données facilement disponibles
  - Données perdues lorsque éteint ➔volatile
  - Power hungry
  - Appelée "**mémoire**" (ex: les registres et la mémoire centrale)

Ou

- ***Lent, mais beaucoup plus grand, stockage permanent***
  - Pour archivage (non-volatile)
  - Données utiles pour accès ultérieur
  - Structuré et indexé
  - Accès plus lent
  - Peu gourmand en énergie
  - Appelée (mémoire de) "stockage"

Besoin de technologie pour stocker les données

Memory (GB)

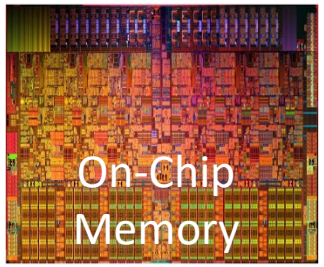


Hard Disk Array (TB)

Tape Robot (PB)

Hard Disk (TB)

Memory (MB)



Tablet FLASH (GB)

USB FLASH (GB)

1 MB = 10^6 bytes

1 GB = 10^9 bytes

1 TB = 10^{12} bytes // Tera

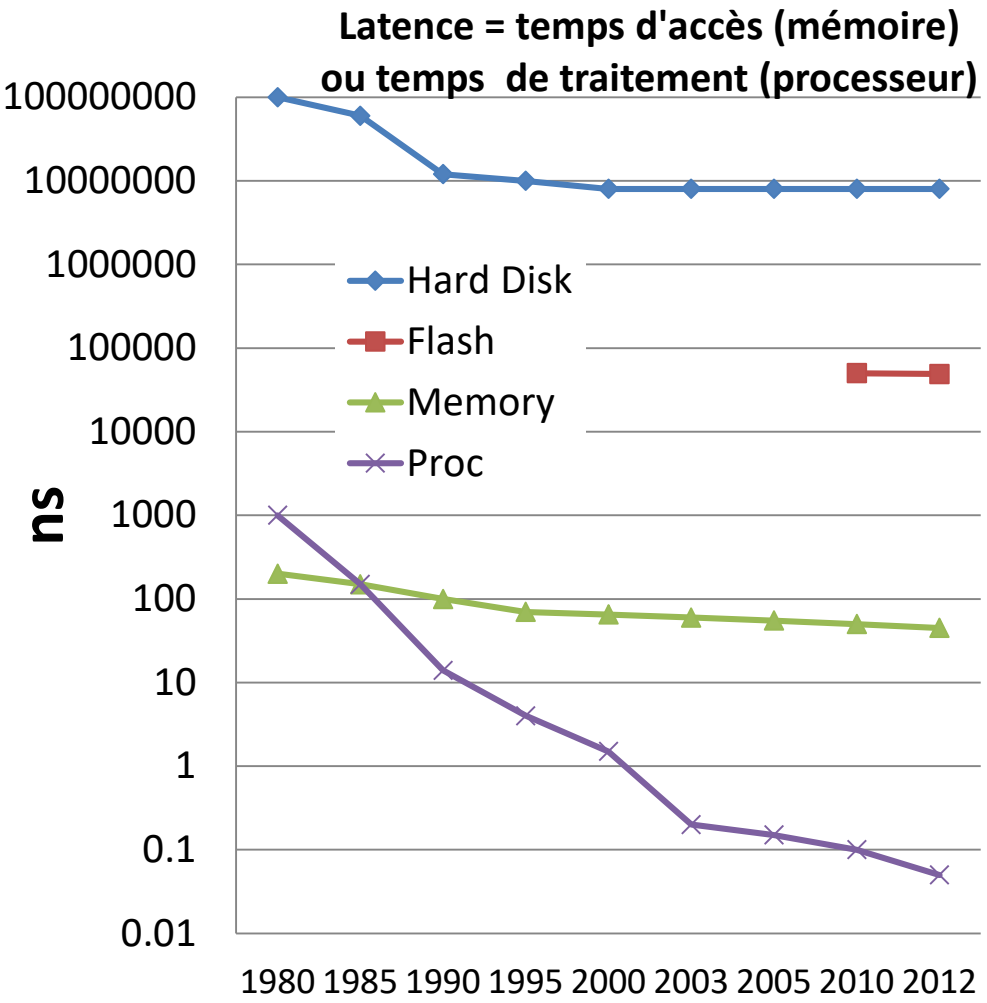
1 PB = 10^{15} bytes // Peta

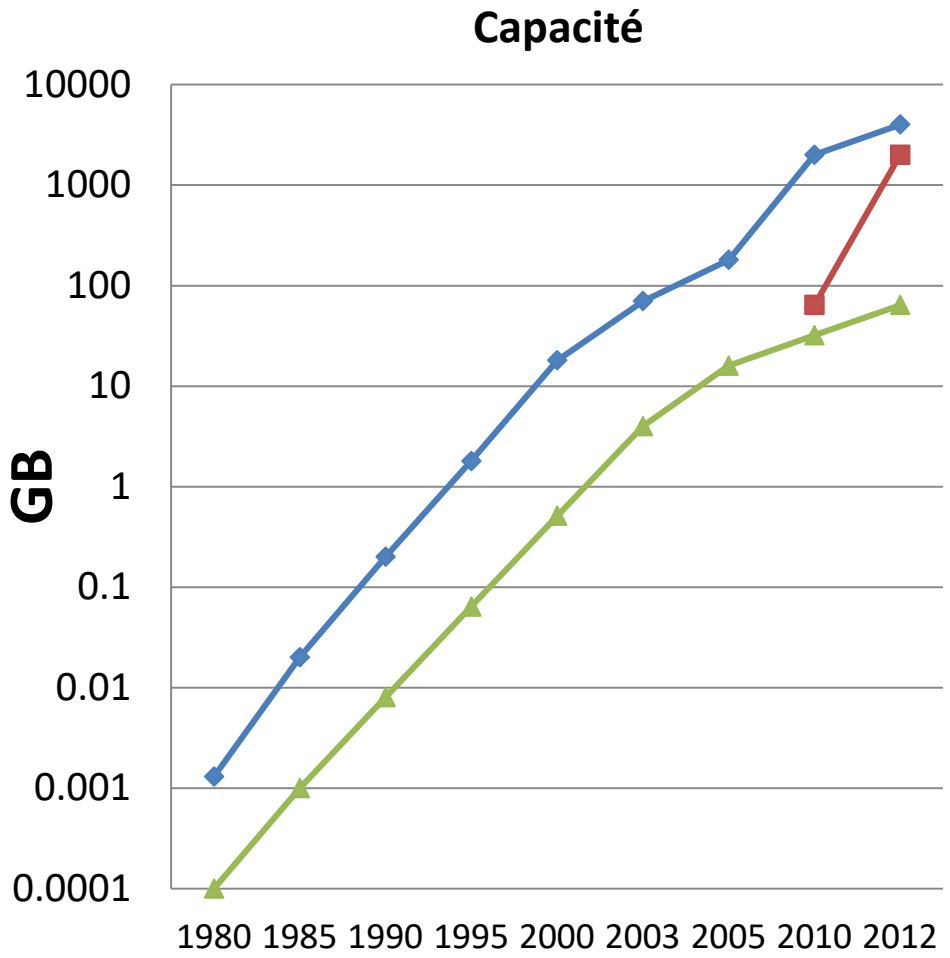
1 EB = 10^{18} bytes // Exa

1 ZB = 10^{21} bytes // Zetta

1 YB = 10^{24} bytes // Yota

Capacité vs. latence: tendances





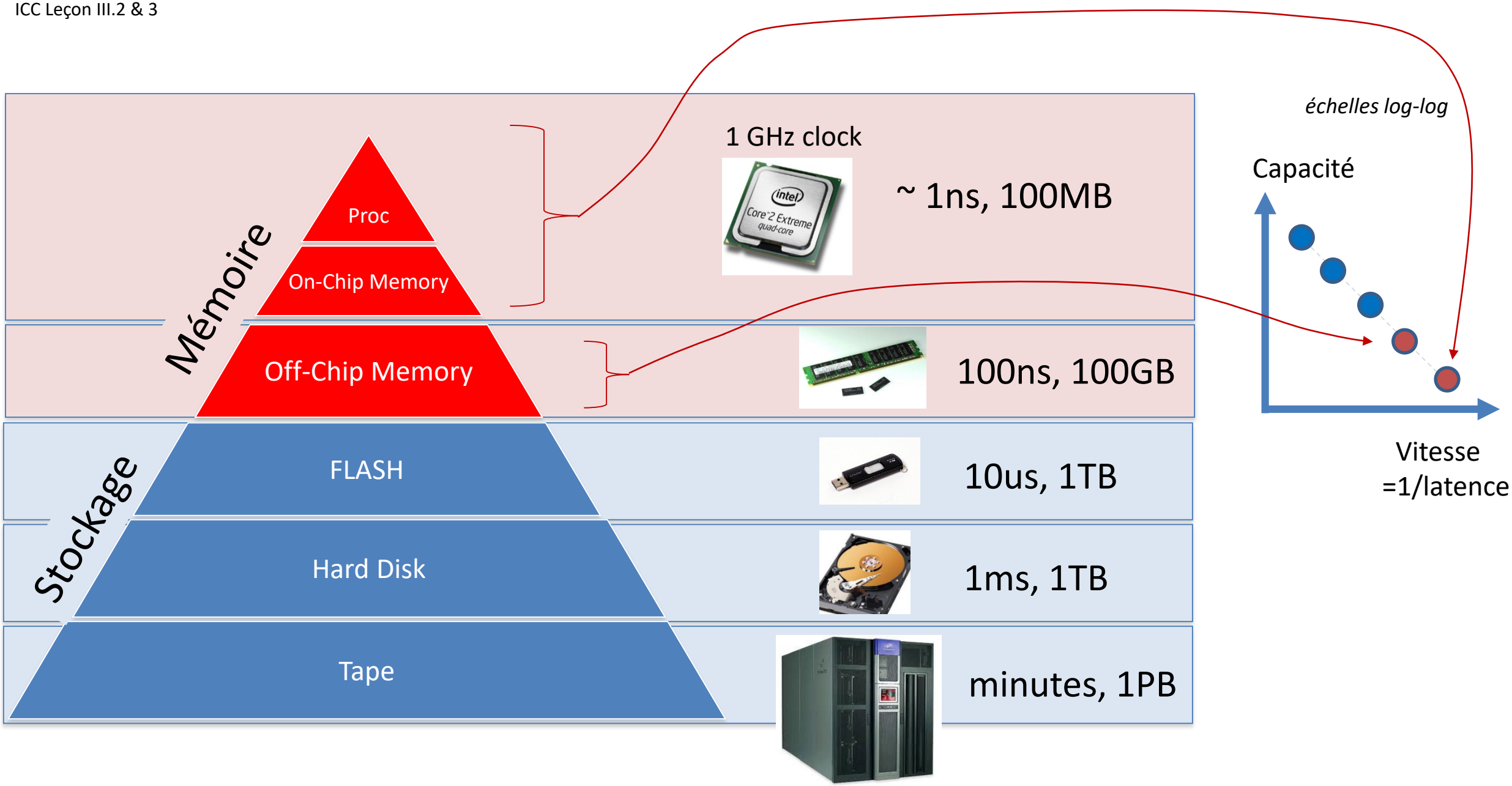
La Hiérarchie de Stockage et le *Memory Wall*

► Les performances des ***mémoires*** s'améliorent surtout en terme de ***capacité*** alors que celles des ***processeurs*** s'améliorent en terme de ***vitesse***.

- Or, la véritable vitesse d'un processeur est celle du traitement des données qu'il accède en mémoire. L'accès mémoire devient un goulot d'étranglement qui a conduit à l'expression ...

- **Memory Wall** : la latence de la mémoire détermine de plus en plus la vitesse des calculs à cause du temps d'accès aux données.

- La technologie Flash est une réponse récente (10 ans) à ce défi


Mémoire
Proc
On-Chip Memory
Off-Chip Memory
Stockage
FLASH
Hard Disk
Tape
1 GHz clock
~ 1ns, 100MB
100ns, 100GB
10us, 1TB
1ms, 1TB
minutes, 1PB
échelles log-log
Capacité
Vitesse
=1/latence

Pourquoi nous avons besoin du *Stockage*?

► La mémoire principale ne suffit pas?

► NON! Parce que:

- Elle coûte beaucoup trop cher (coût dans tableau suivant)
- Pas assez de capacité!
- La mémoire principale est ***volatile***. Nous voulons la sauvegarde des données, même en cas de panne de courant.

► D'où: ~ 60% du coût d'un système de production est dans les disques.

Technologie: latence, debit, coût et rétention

Mem centrale

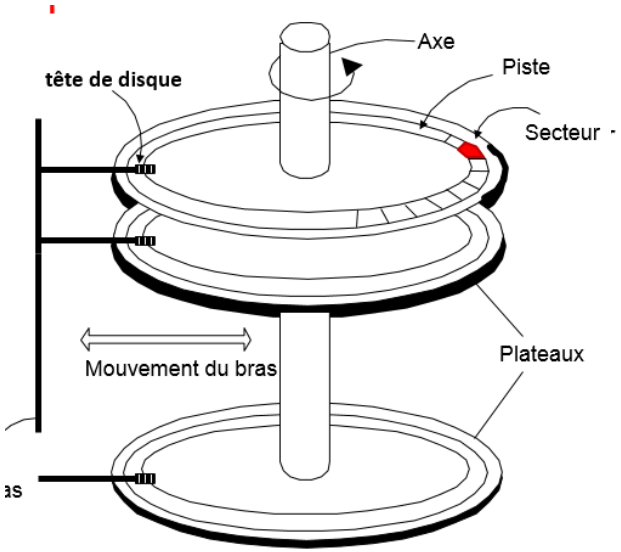
	Latence	Débit	Coût ($/GB)	Capacité	Rétention
RAM	100 ns	GB/s	10	MB - GB	NON
Flash	µs	GB/s	0.5	GB - TB	Oui
Disques HDD	ms	100s MB/s	0.05	> TB	Oui
Bandes magnétiques	Encore plus lent !	100s MB/s	Encore moins cher !	Encore plus grand !	Oui

FLASH

HARD DISK DRIVE

Bande Magnétique

électrons piégés dans semi-conducteur

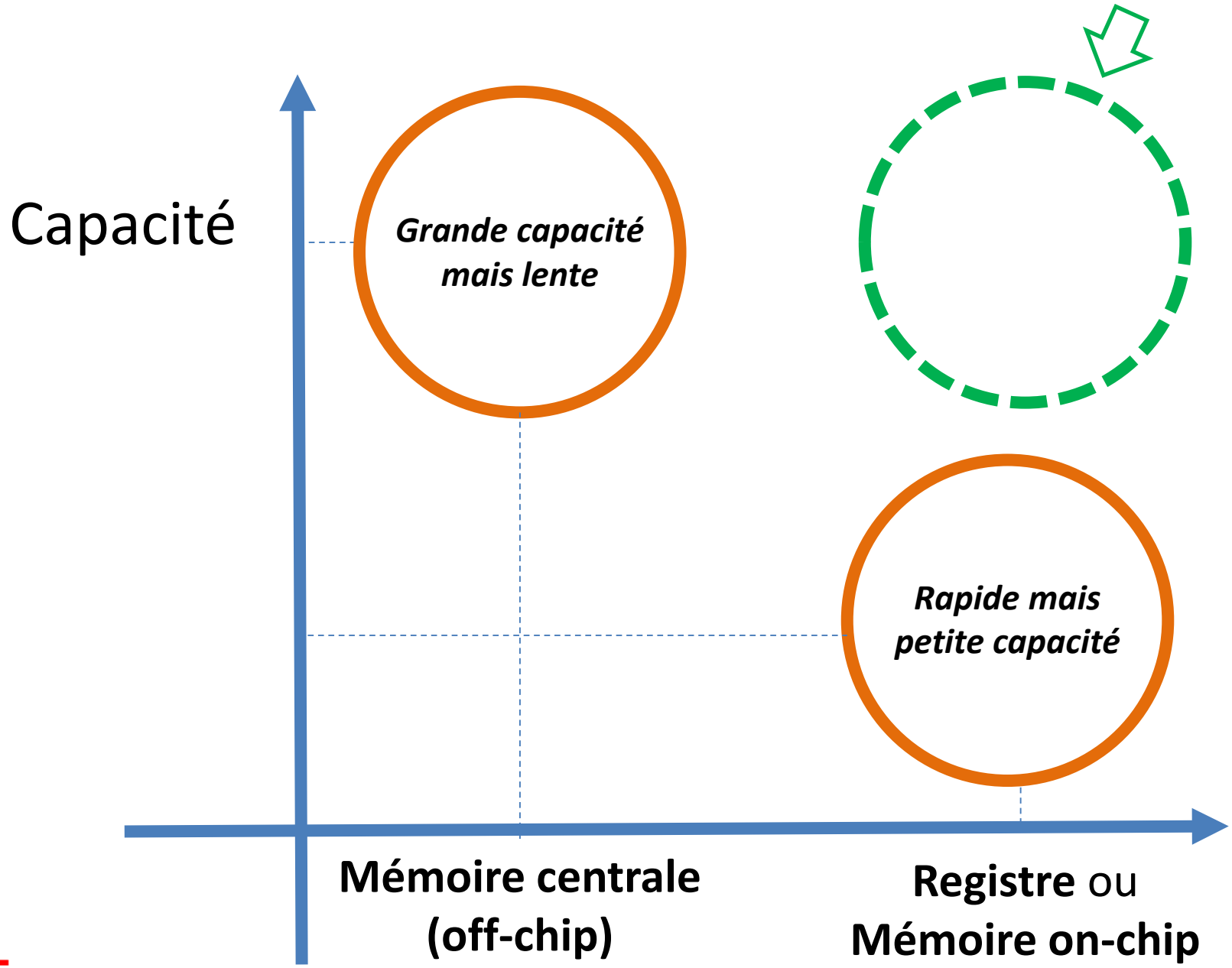


charge magnétique

Plan

- Introduction : mémorisation et stockage
- **Principe du cache: une mémoire grande et rapide**
- Fonctionnement du cache avec le processeur et la mémoire centrale
- Le compromise entre localité spatiale et localité temporelle
- Impact de l'organisation des données sur les performances d'un algorithme
- Structuration du stockage des données

QUE FAIRE ? Hiérarchiser la mémoire en utilisant un ***cache***



***Motivation**: donner l'illusion d'une mémoire **grande** et **rapide** au processeur*

Principe du cache:

► Idéalement, donner au processeur tout ce dont il a besoin en 1ns (= un cycle de l'horloge d'un processeur à 1 GHz)

• Mais toutes les données ne peuvent être gardées on-chip

Conserver on-chip seulement les données que ***l'on va utiliser prochainement***

Comment compenser une faible vitesse d'accès (latence) ?

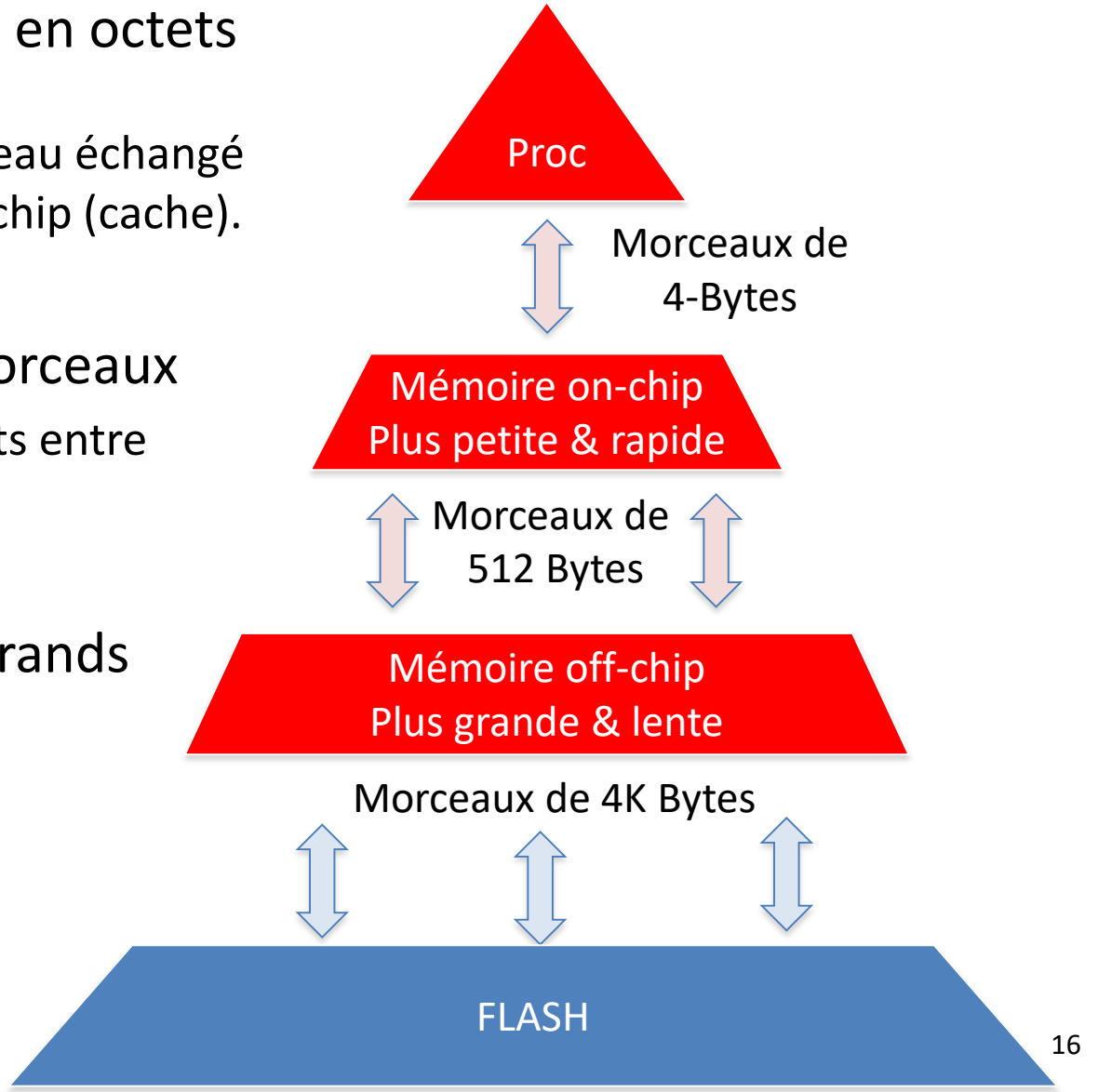
Réponse: en transférant des morceaux plus gros

- La taille des données est exprimées en octets
  - Plus petite unité de mémoire
  - Mot = 4 octets(bytes) = taille du morceau échangé entre le Processeur et la mémoire on-chip (cache).

- **mémoire *on-chip* (cache)*:*** petits morceaux
  - D'un mot à quelques centaines d'octets entre le cache et la mémoire off-chip

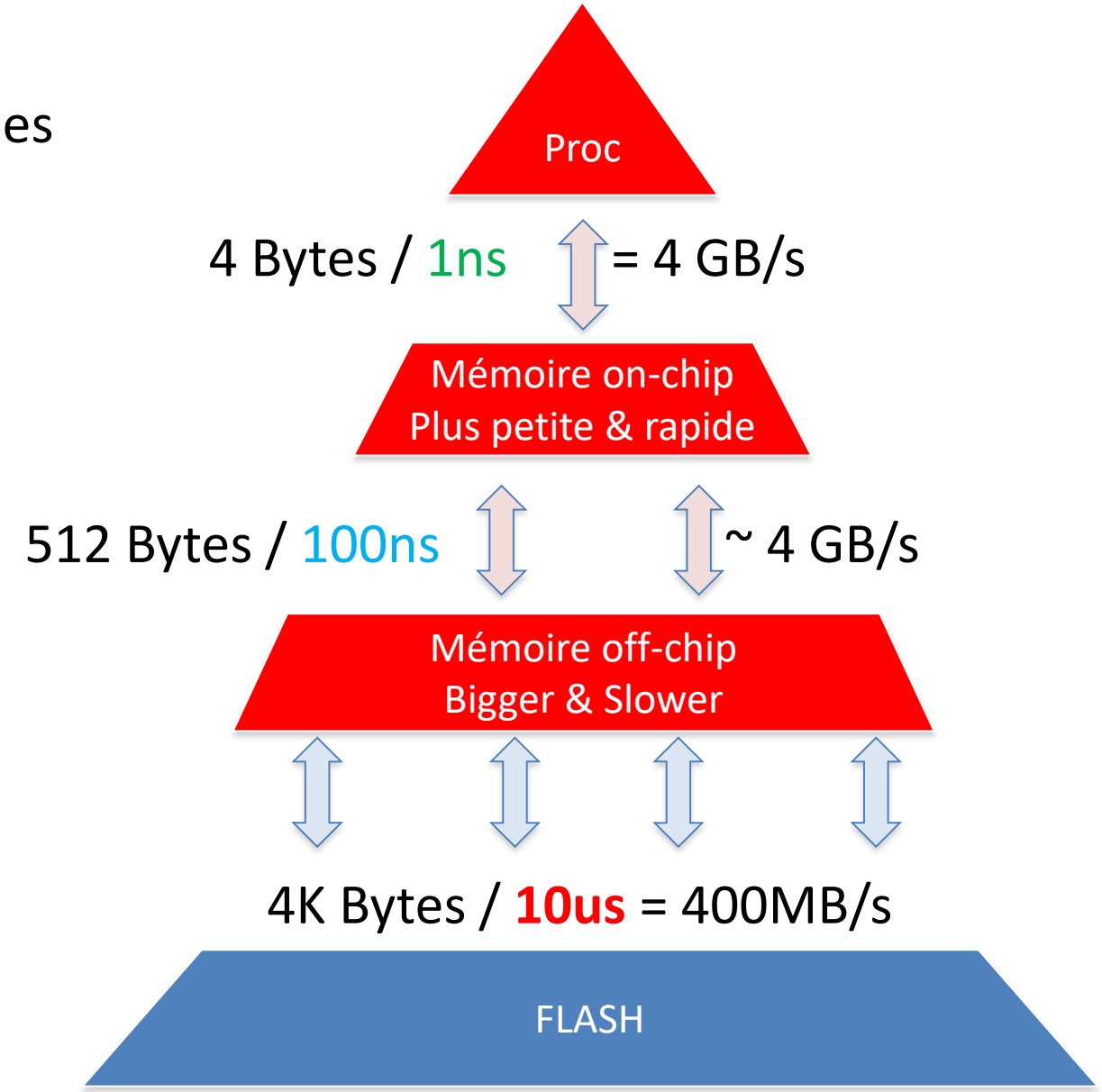
- **mémoire *off-chip***, morceaux plus grands
  - Quelques milliers d'octets

- Raisonnement niveaux inférieurs
  - Plus grands ➔ plus de capacité
  - Plus larges ➔ plus d'info déplacées par unite de temps



(Taille de morceau / Latence) = "Bande passante" (=débit)

- Différence de latence (temps d'accès) sur les 3 niveaux:
  - Temps d'accès d'un mot (4 Bytes)
  - **1ns** vs. **100 ns** vs. **10us** (flash)
- Mais les niveaux inf. sont plus *larges*
  - Peuvent transférer plus de données à la fois
  - Appelé "bande passante"
- Pour obtenir une bande passante élevée
  - Accès d'un morceau en parallèle
  - Avec un morceau de 4KB, la bande passante est 10 fois plus petite qu'au niveau du Proc.
  - Mais la latence est 10'000x plus élevée!



Plan

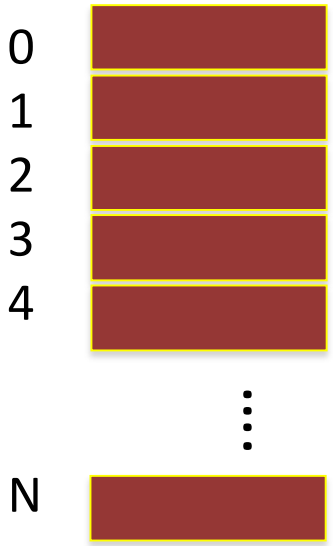
- Introduction : mémorisation et stockage
- Principe du cache: une mémoire grande et rapide
- **Fonctionnement du cache avec le processeur et la mémoire centrale**
- Le compromis entre localité spatiale et localité temporelle
- Impact de l'organisation des données sur les performances d'un algorithme
- Structuration du stockage des données

La Mémoire du point de vue du processeur (mémoire centrale)

- Organisée comme un tableau de mots
  - chaque mot:
    - contient 4 à 8 octets (selon machine)
    - est numéroté (~ adresse postale)
    - à une ***adresse*** entre 0 et N:
      1er mot @ 0,
      2ème mot @ 1, etc.

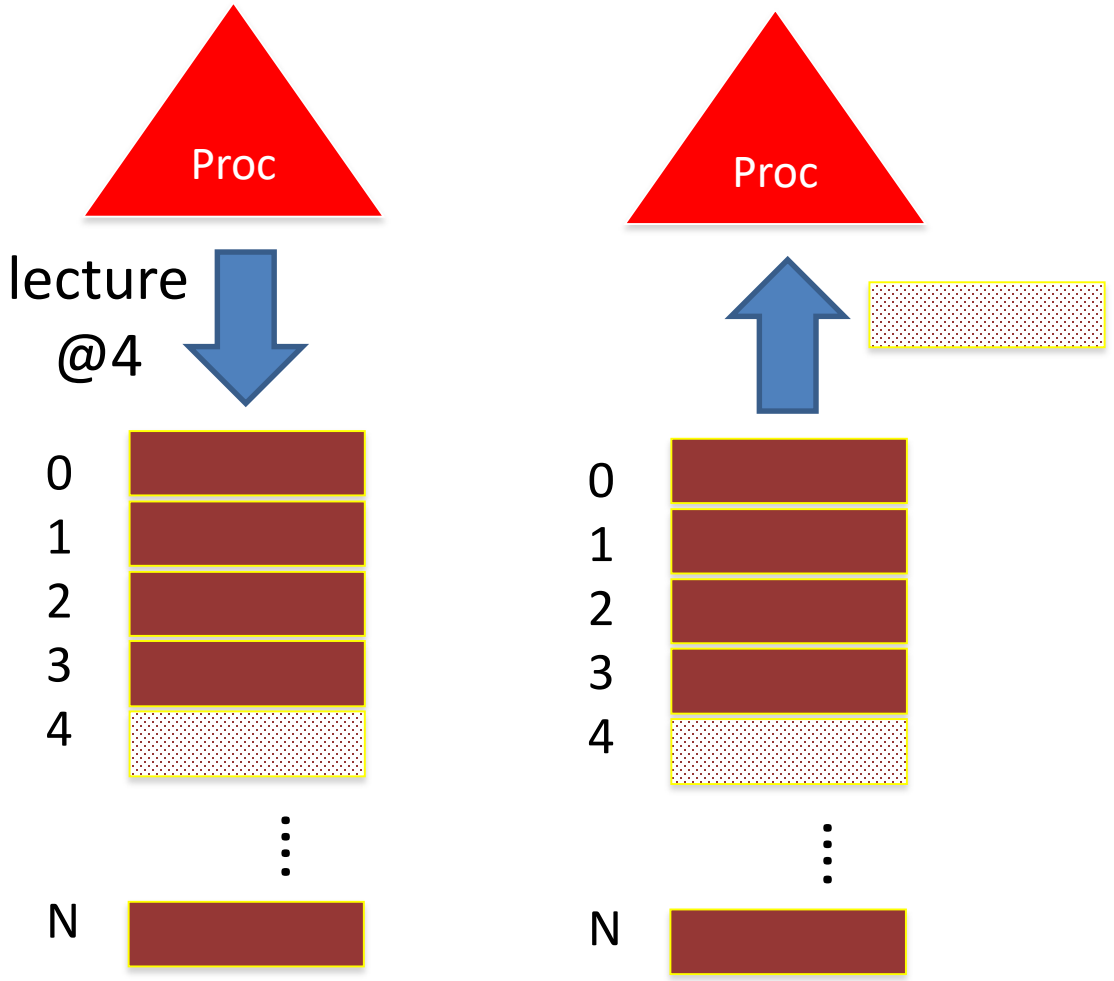
- Stocke données & instructions (III.1)
  - De multiples programmes peuvent coexister
  - Intervalle d'adresses assigné à chaque programme (segment)

Tableau de mots



Lecture d'un mot mémoire par le processeur

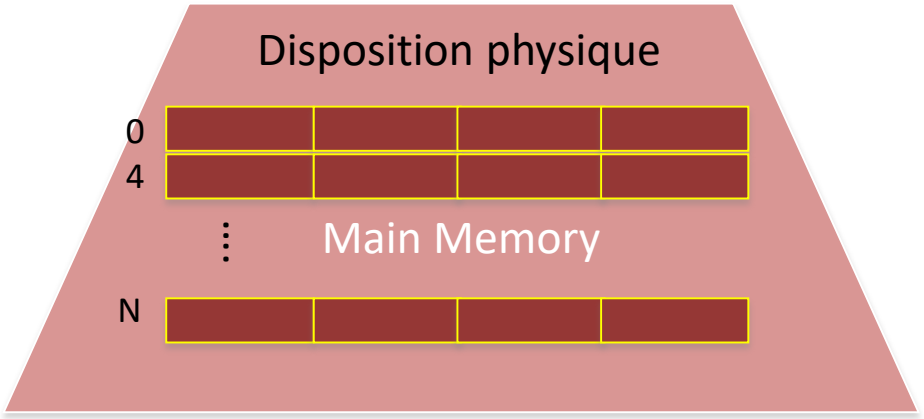
- le processeur demande à la mémoire le contenu (motif binaire) du mot d'adresse A
- La mémoire renvoie la valeur du mot au processeur
- le processeur stocke la valeur dans un registre



Organisation de la mémoire centrale

- Déplacer plus d'infos à la fois pour compenser la latence
  - la ***mémoire centrale*** est organisée en ***blocs (de 16 à 512 Bytes)***
  - le ***cache*** est aussi organisé en ***blocs***
  - le transfert entre les deux se fait par ***bloc***

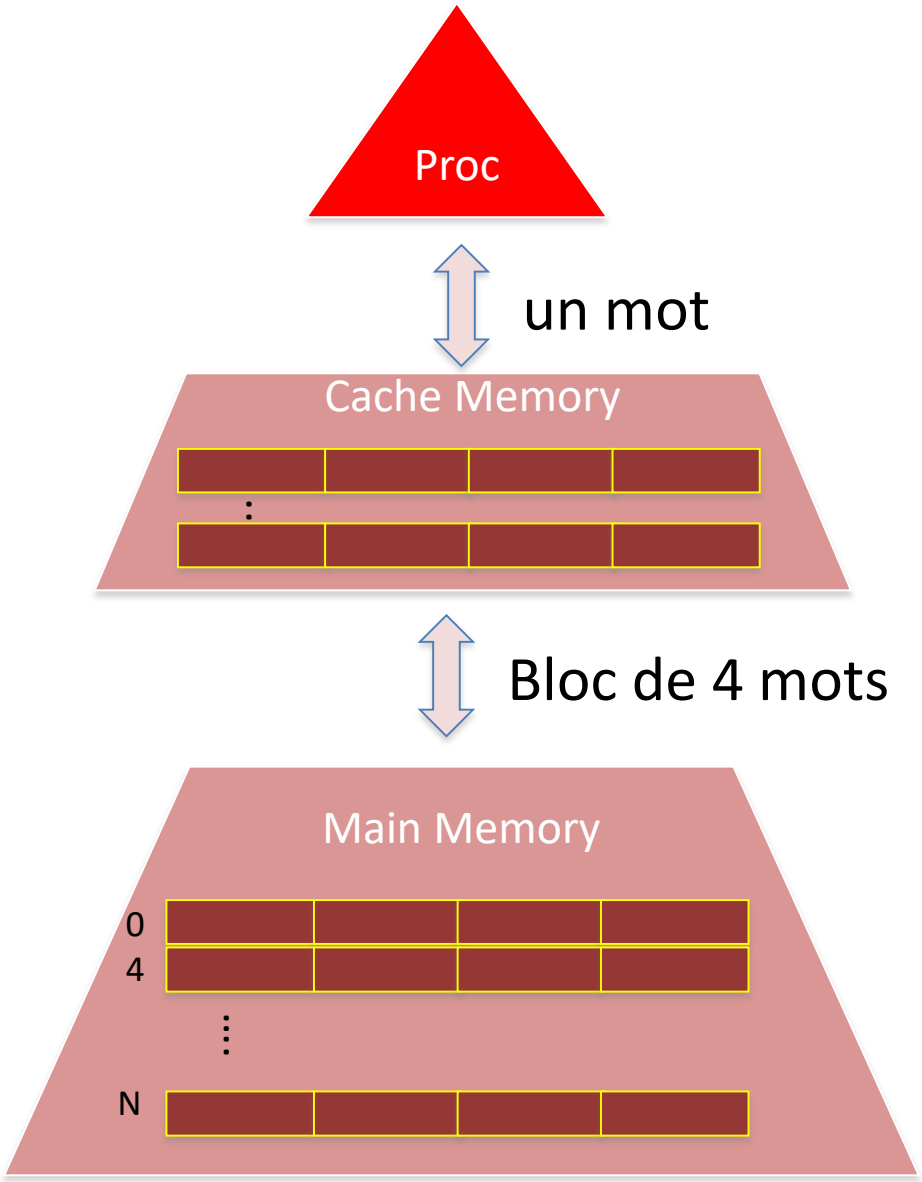
- representation de la mémoire centrale:



Bloc = 16 Bytes
= 4 mots

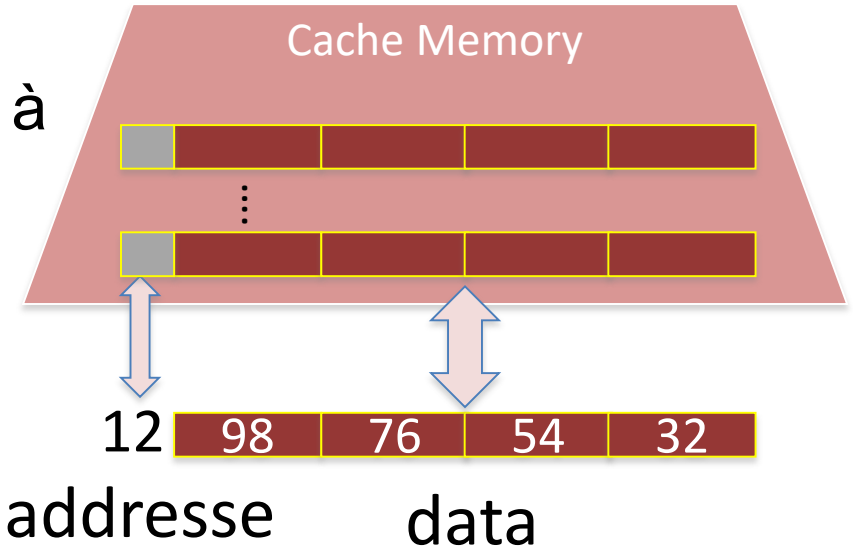
Mémoire on-chip (cache)

- Mémoire plus rapide
  - 1ns à 10ns
  - Plus proche de la vitesse du processeur
- Mais capacité limitée
  - 64KB à 64MB
  - Mémoire principale peut avoir 1TB
  - Ne peut contenir toutes les données/instructions
  - Infos regroupées par bloc
- Q: Comment rechercher un bloc?
- Q: Quels blocs en cache?



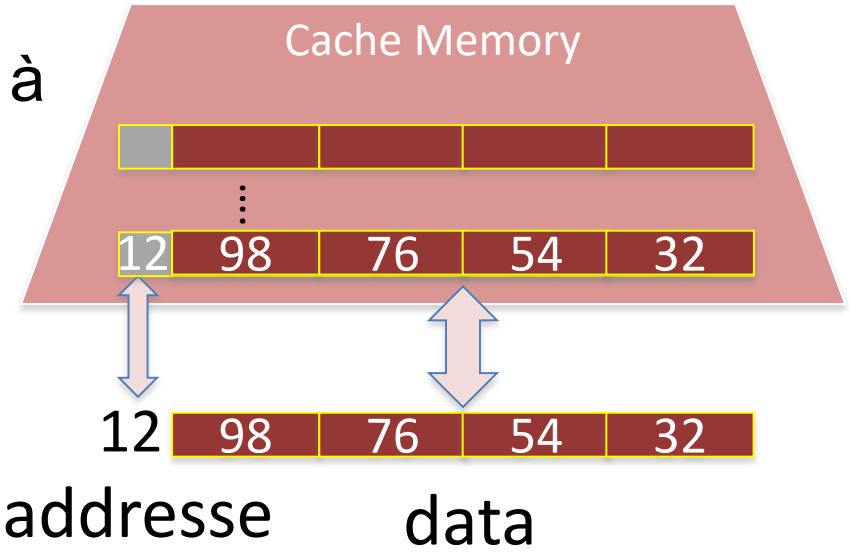
Transfert de la mémoire centrale vers le cache

- Cache:
  - Tableau composé de deux parties:
    - adresse d'un bloc de la mémoire centrale (ex: 12)
    - les données du bloc commençant à cette adresse (ex: 98765432)



Transfert de la mémoire centrale vers le cache

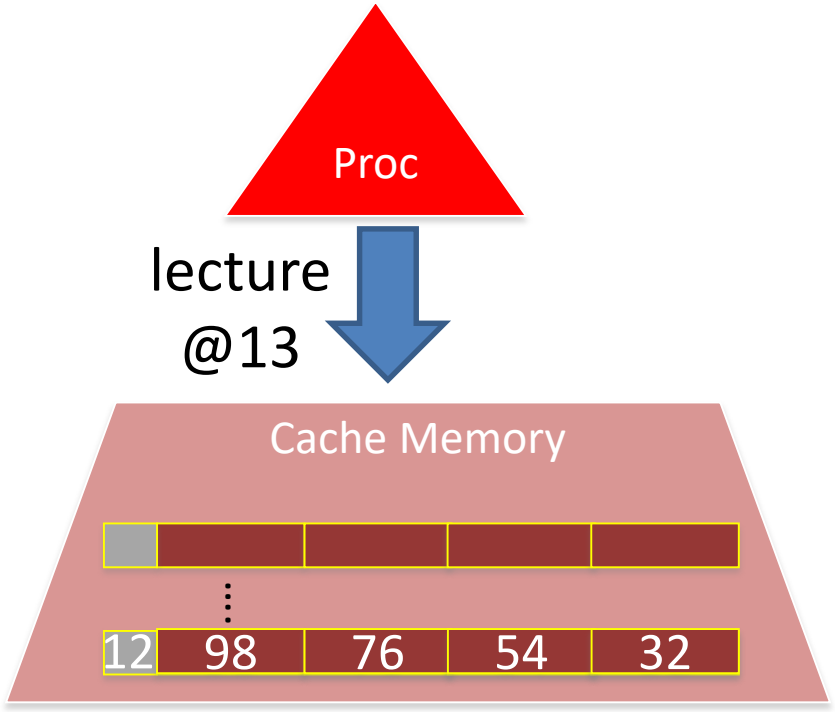
- Cache:
  - Tableau composé de deux parties:
    - adresse d'un bloc de la mémoire centrale (ex: 12)
    - les données du bloc commençant à cette adresse (ex: 98 76 54 32)



  - Plusieurs blocs de la mémoire centrale peuvent résider temporairement dans le cache

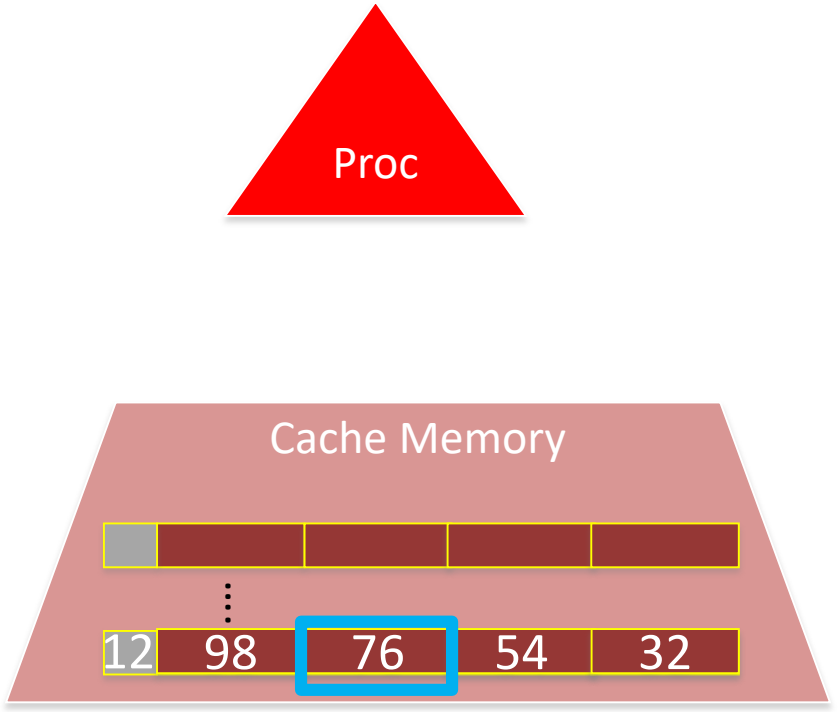
Recherche de données par le Processeur

- ***Le processeur envoie l'adresse du mot recherché***



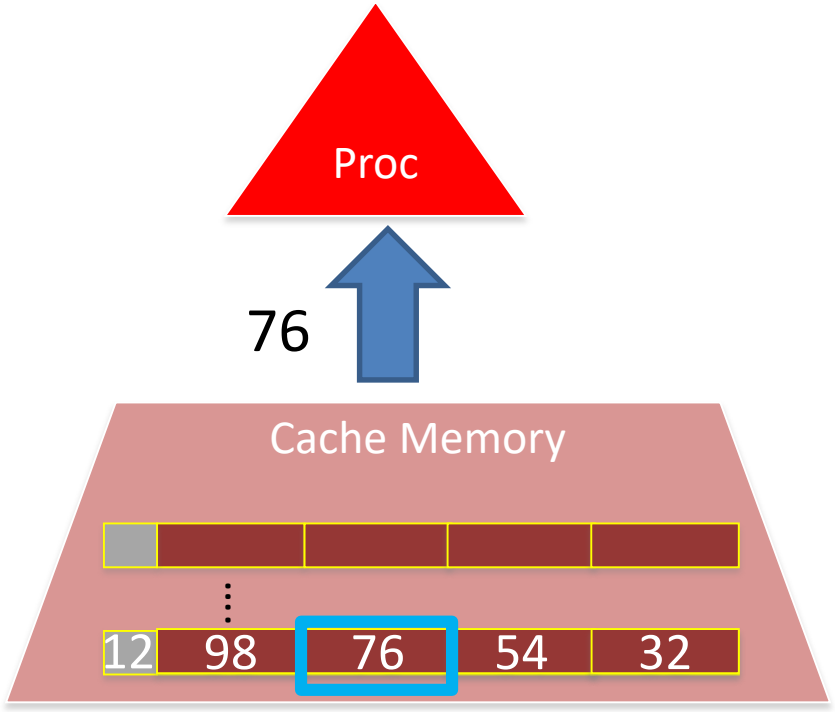
Recherche de données par le Processeur

- ***Le cache vérifie si la donnée est présente dans le cache***



Recherche de données par le Processeur

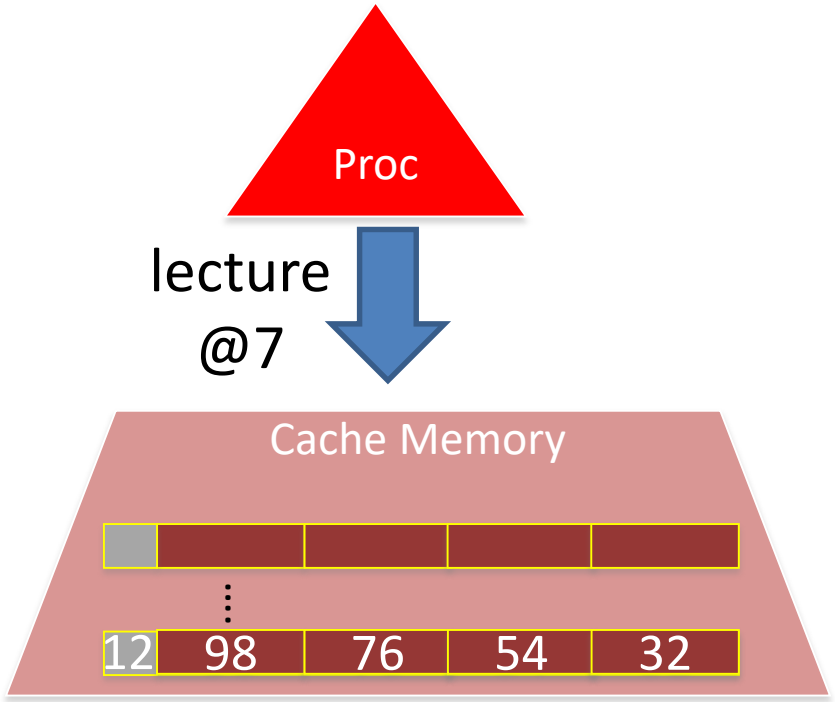
- ***Si oui, le cache envoie le mot (1ns)***



Recherche de données par le Processeur (cas échec = défaut de cache)

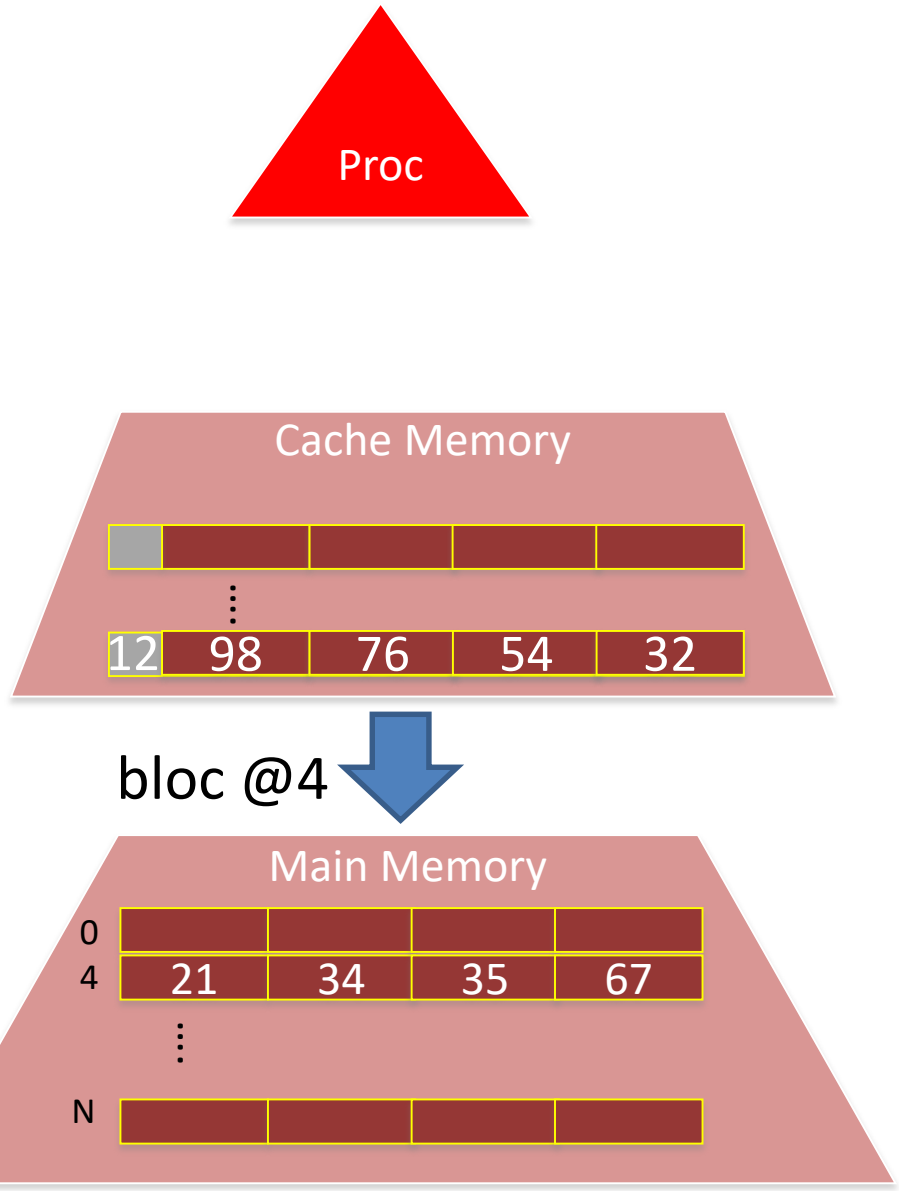
Toutes les données ne peuvent pas être dans le cache !

- ***Le processeur envoie l'adresse du mot recherché***



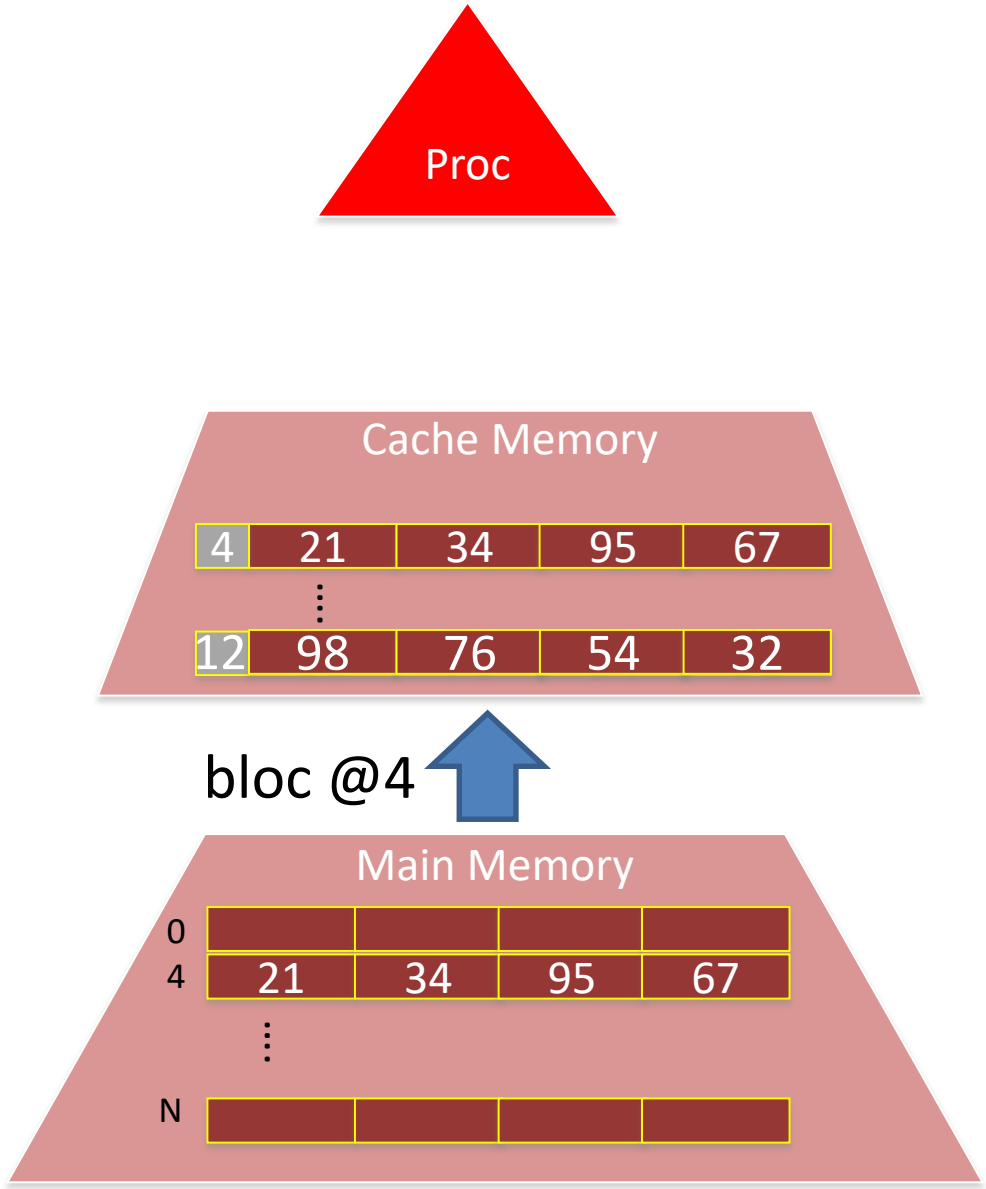
Recherche de données par le Processeur (cas échec = défaut de cache)

- ***Le cache vérifie si la donnée est présente dans le cache***
- ***Si non, il faut charger un bloc de la mémoire vers le cache (100ns)***



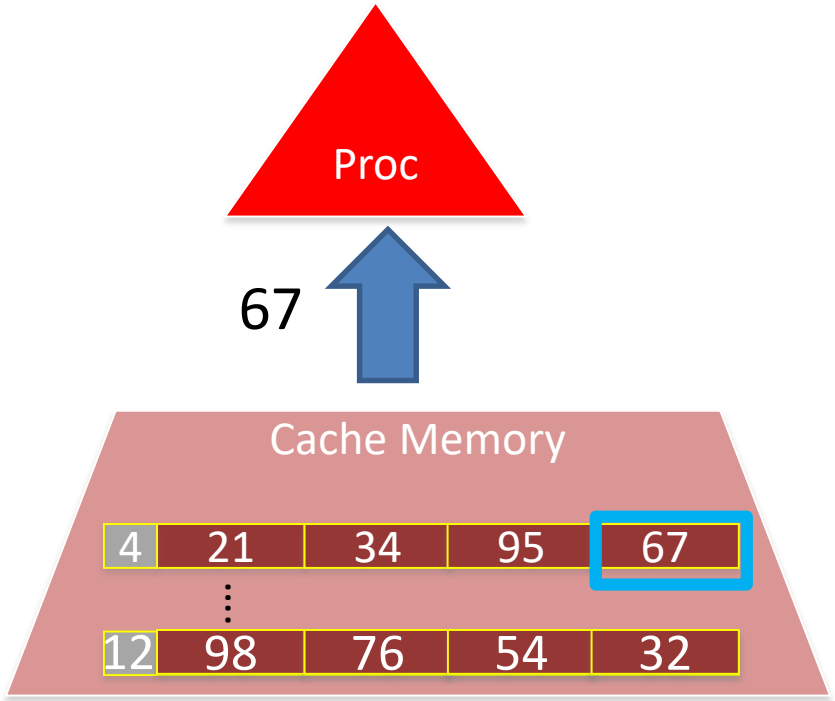
Recherche de données par le Processeur (cas échec = défaut de cache)

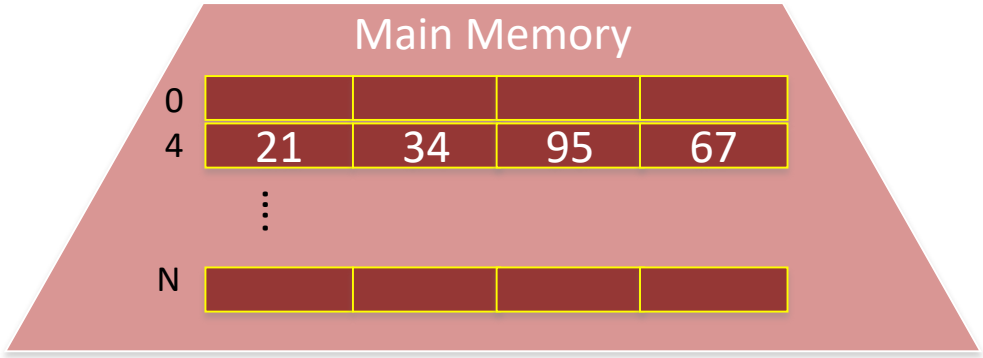
- ***Si non, il faut charger un bloc de la mémoire vers le cache (100ns)***



Recherche de données par le Processeur (cas échec = défaut de cache)

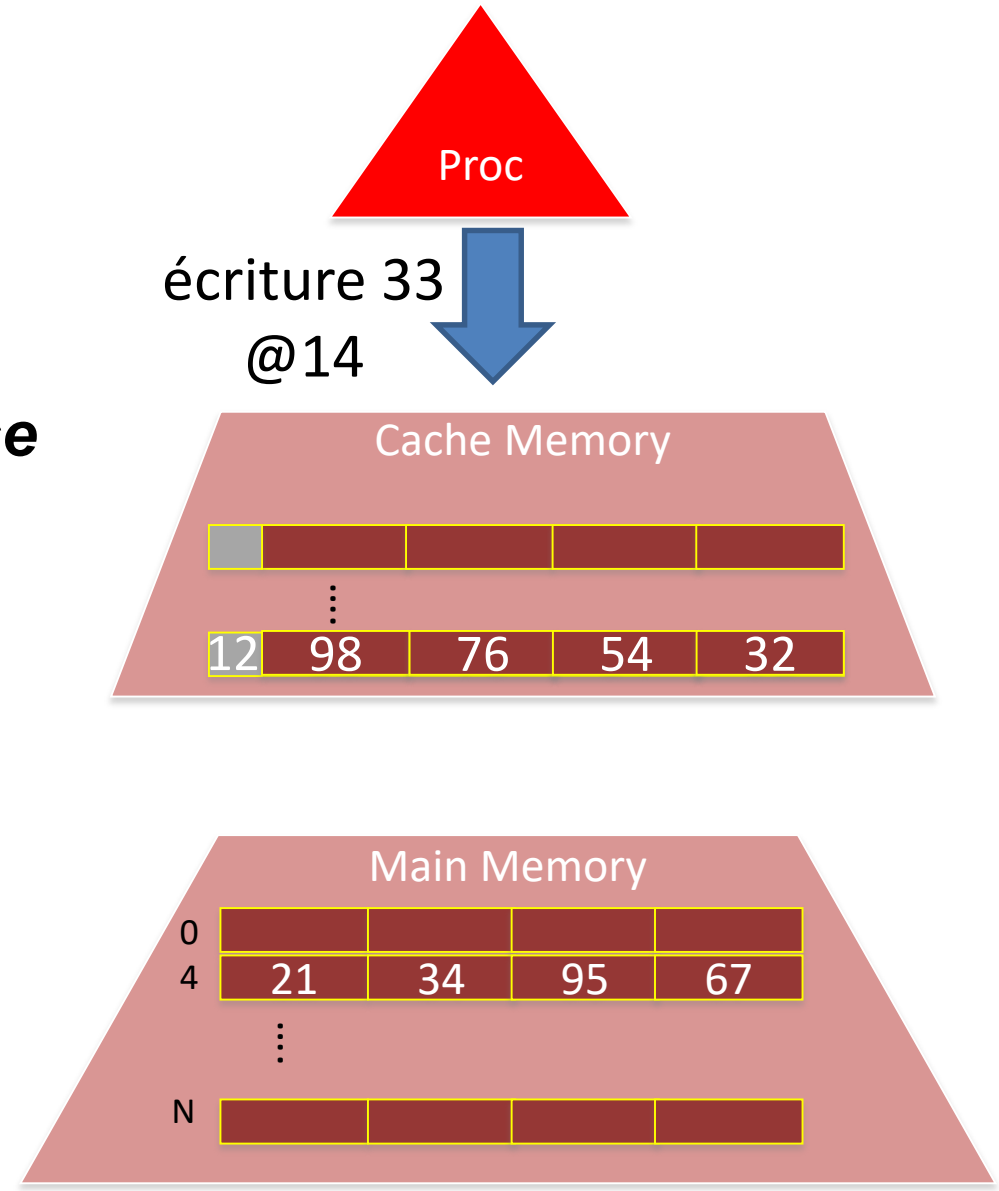
- ***le coût est proche d'une lecture sans cache***





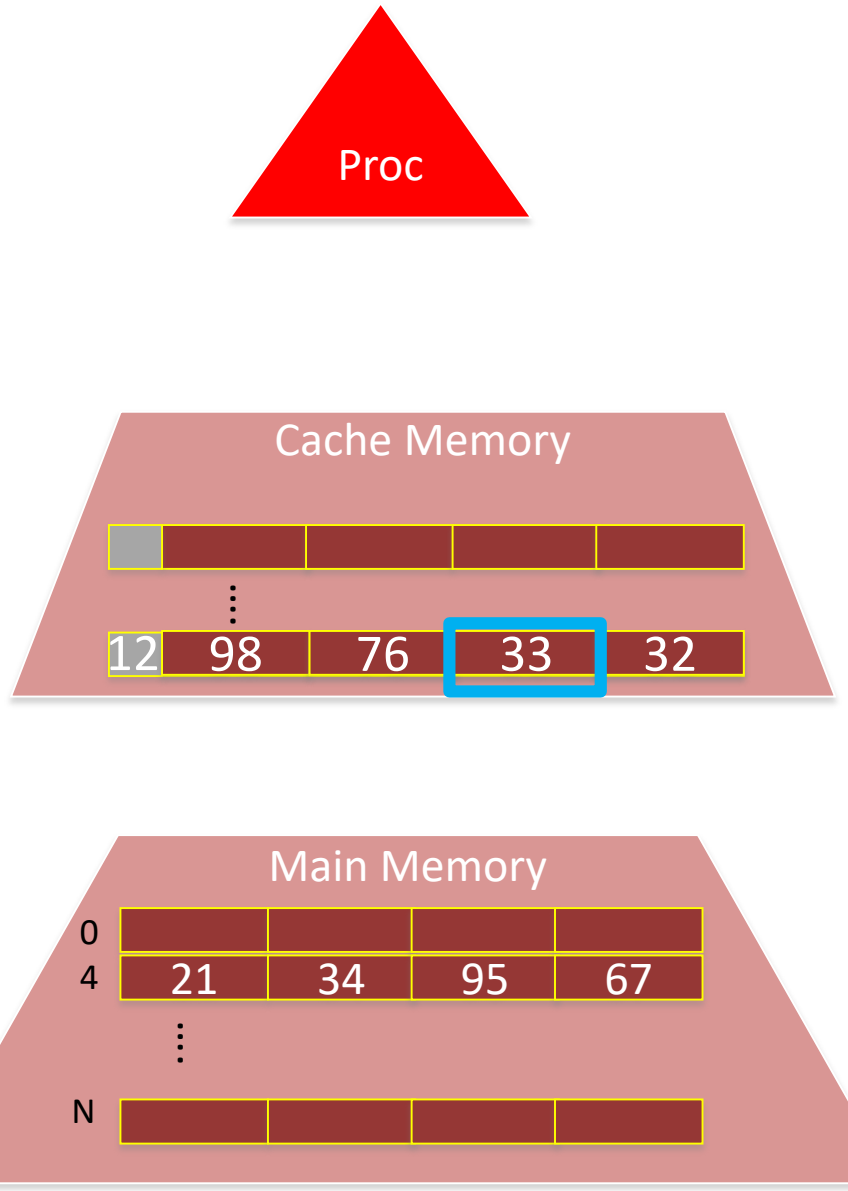
Ecriture d'une donnée par le Processeur

- ***Le processeur envoie l'adresse et la valeur du mot à écrire***



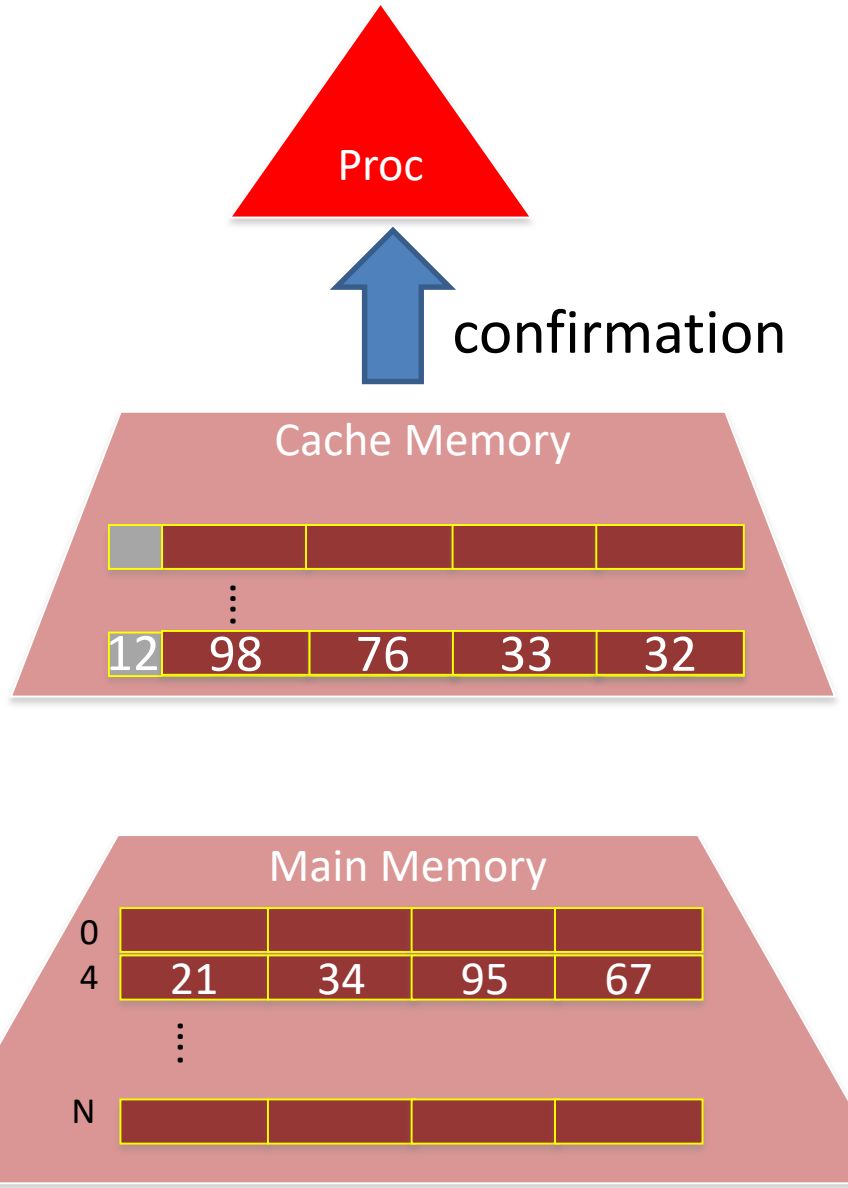
Ecriture d'une donnée par le Processeur

- Le cache vérifie si la donnée est présente dans le cache
- ***Si oui, la valeur est écrite dans le cache (1ns)***



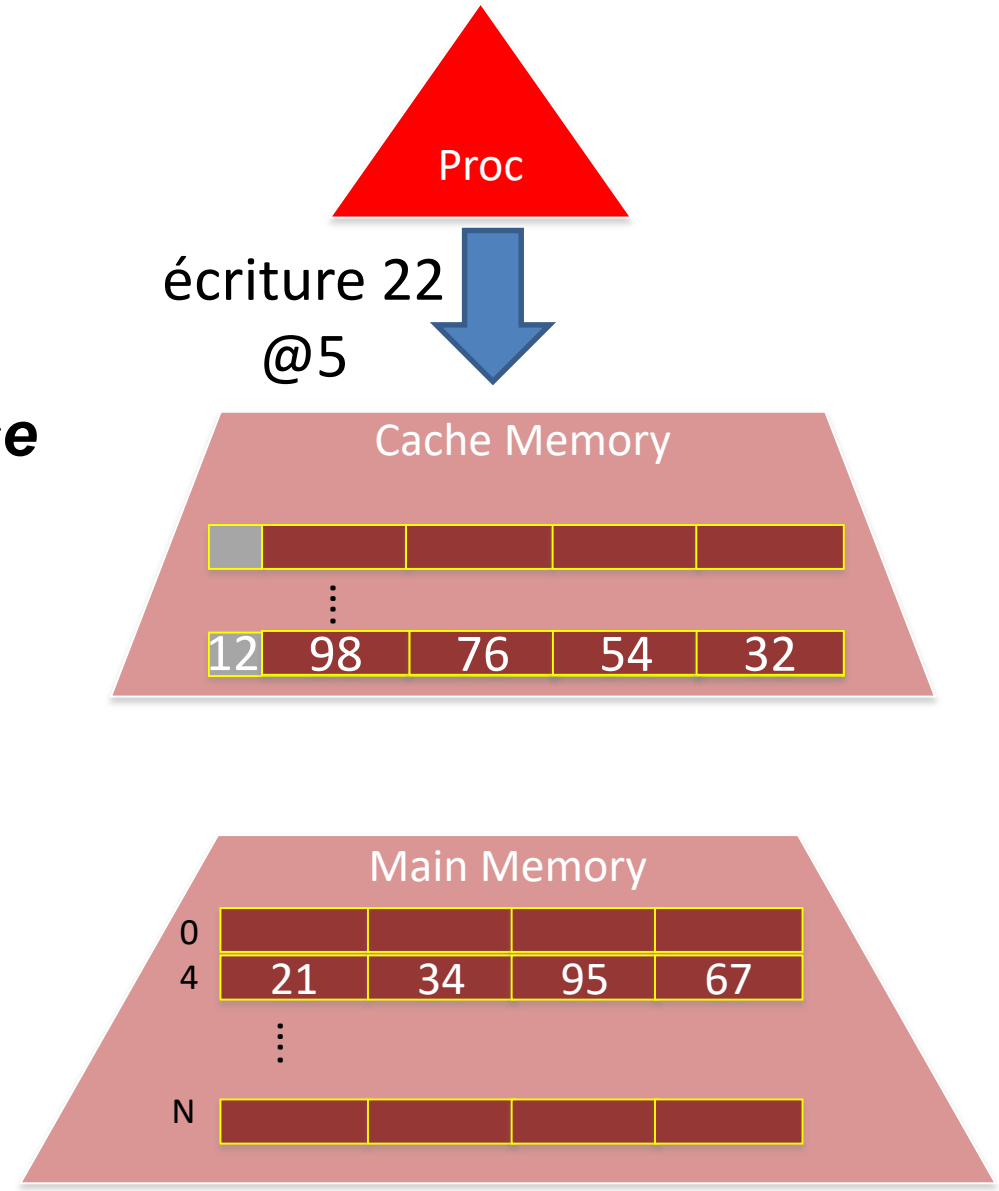
Ecriture d'une donnée par le Processeur

- ***envoie une confirmation au processeur.***



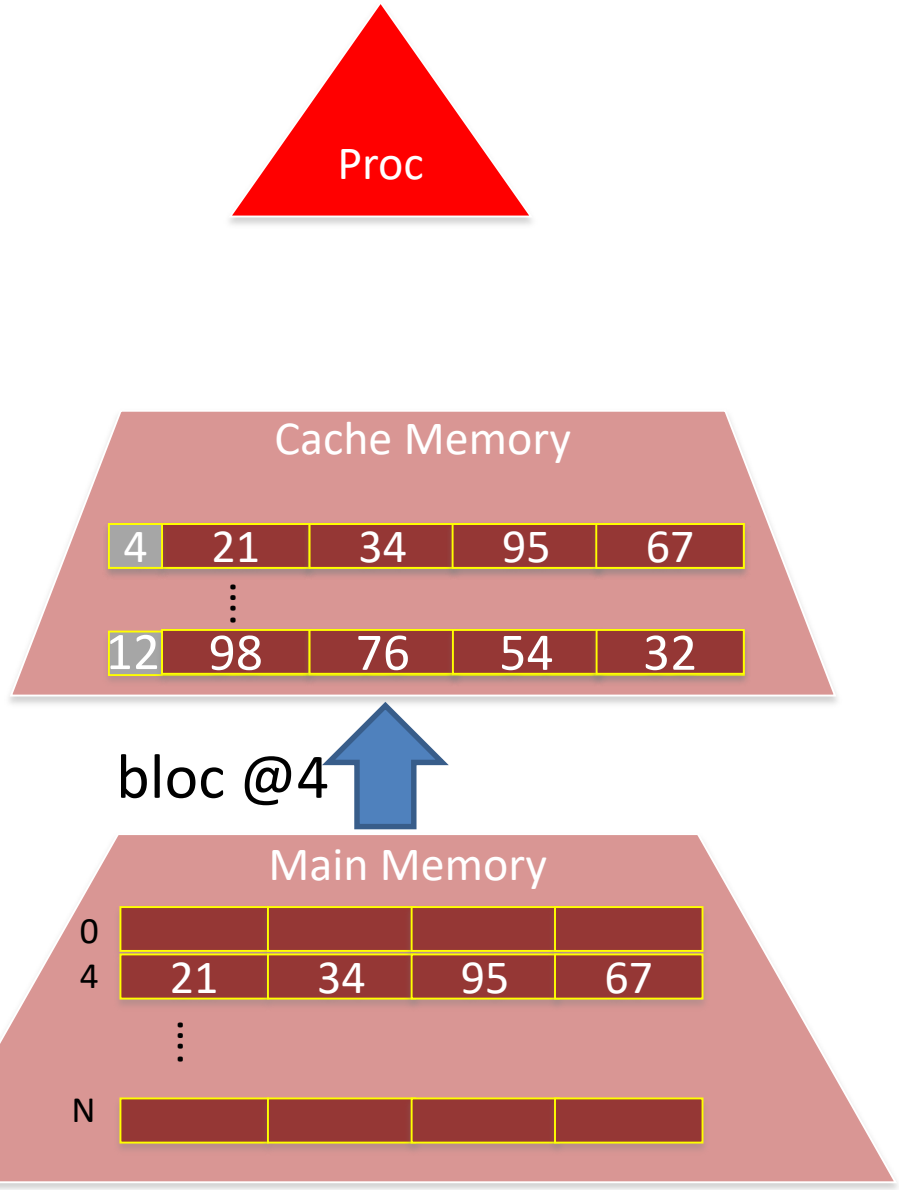
Ecriture d'une donnée par le Processeur (échec = défaut de cache)

- ***Le processeur envoie l'adresse et la valeur du mot à écrire***



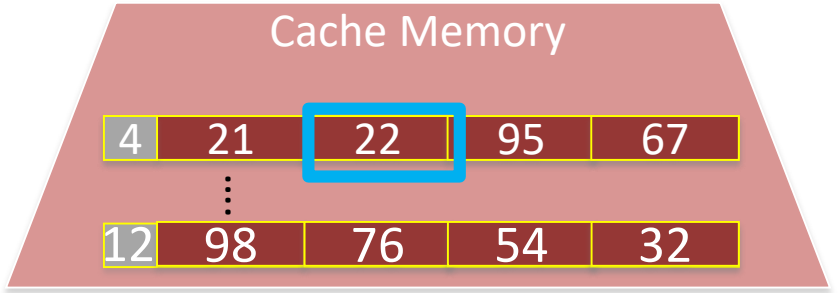
Ecriture d'une donnée par le Processeur (échec = défaut de cache)

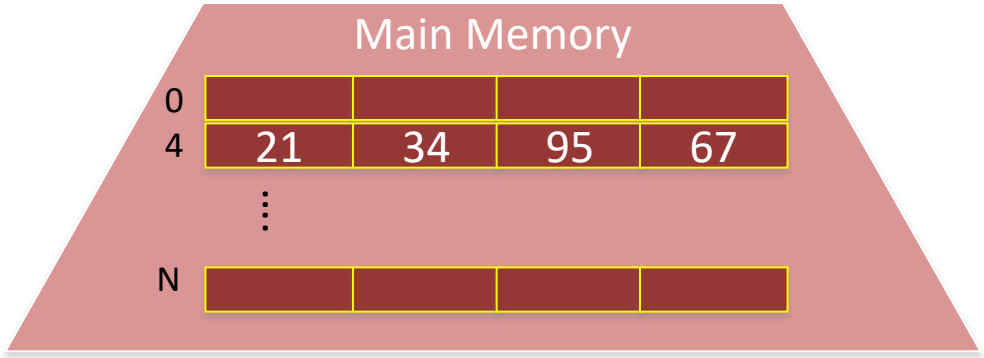
- Le cache vérifie si la donnée est présente dans le cache
- ***Si non, il faut charger un bloc de la mémoire vers le cache (100ns) puis écrire dans le mot.***



Ecriture d'une donnée par le Processeur (échec = défaut de cache)

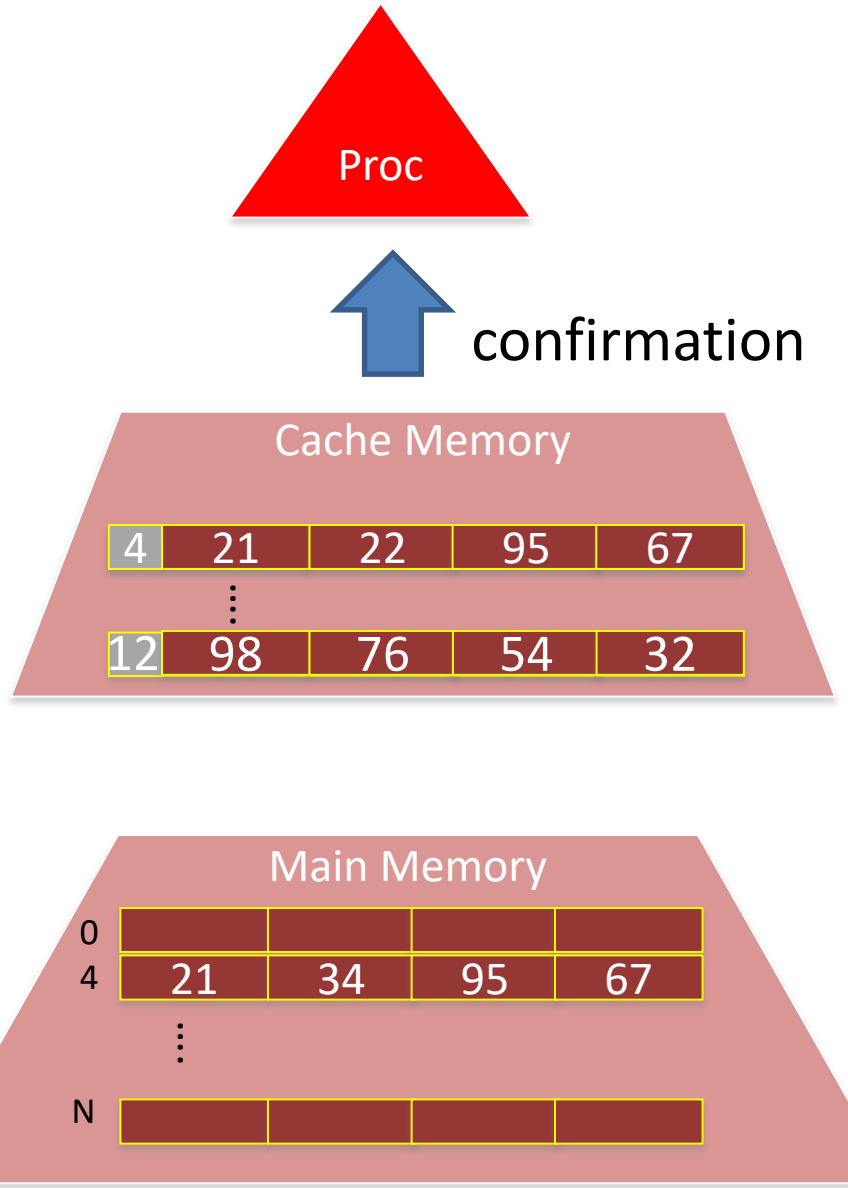
- Le cache vérifie si la donnée est présente dans le cache
- ***Si non, il faut charger un bloc de la mémoire vers le cache (100ns) puis écrire dans le mot.***





Ecriture d'une donnée par le Processeur (échec = défaut de cache)

- ***envoie une confirmation au processeur.***



Que faire si le cache est plein ?

- Il faut remplacer un bloc
- Lequel ?
- Une solution courante: celui qui a été utilisé le moins récemment (LRU = Least Recently Used)

► ***Ne pas oublier de recopier le bloc en mémoire***
le cache contient les données les plus récentes des variables

Plan

- Introduction : mémorisation et stockage
- Principe du cache: une mémoire grande et rapide
- Fonctionnement du cache avec le processeur et la mémoire centrale
- **Le compromis entre localité spatiale et localité temporelle**
- Impact de l'organisation des données sur les performances d'un algorithme
- Structuration du stockage des données

D'où provient la réussite dans l'accès au cache?

Lié à la localité des accès

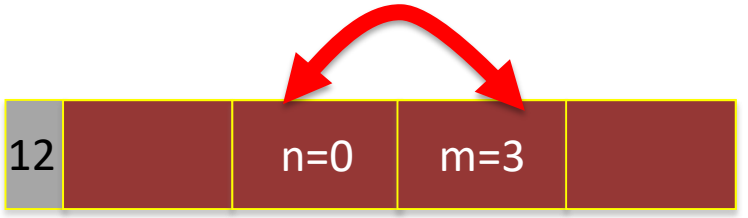
Pourquoi?

Variables déclarées en même temps sont proches en mémoire

Appelé ***"localité spatiale"***

Si un élément est accédé en mémoire,
son voisin a aussi de fortes chances d'être
accédé très prochainement

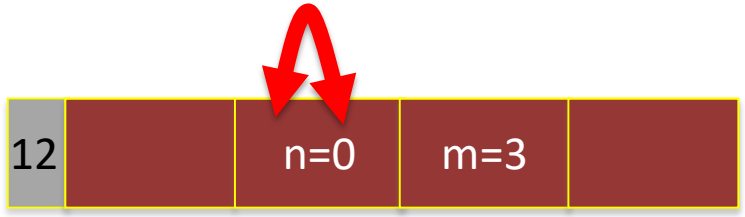
(ex: les données d'un tableau traité dans une boucle).



La même variable peut être accédée plusieurs fois pendant l'exécution

Appelé ***"localité temporelle"***

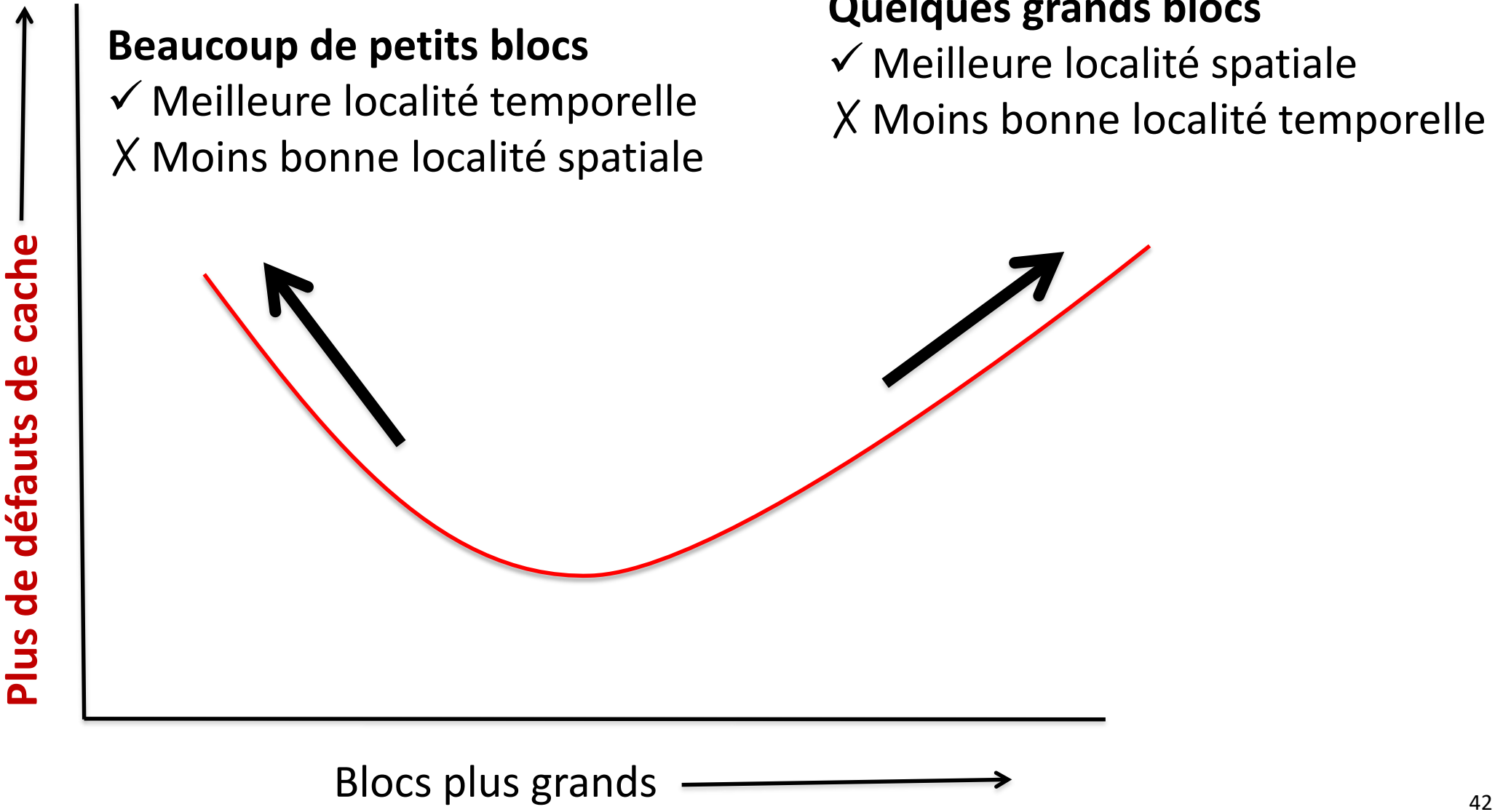
Une variable accédée récemment a de fortes
chances d'être accédée à nouveau dans le futur proche
(ex: compteur d'une boucle, variable accumulateur etc).



Localité spatiale vs. temporelle & Taille des blocs

Beaucoup de petits blocs
✓ Meilleure localité temporelle
✗ Moins bonne localité spatiale

Quelques grands blocs
✓ Meilleure localité spatiale
✗ Moins bonne localité temporelle



Plan

- Introduction : mémorisation et stockage
- Principe du cache: une mémoire grande et rapide
- Fonctionnement du cache avec le processeur et la mémoire centrale
- Le compromis entre localité spatiale et localité temporelle
- **Impact de l'organisation des données sur les performances d'un algorithme**
- Structuration du stockage des données

Impact de l'organisation des données sur les performances d'un algorithme

Addition des éléments de M

$$s \leftarrow \sum_{i=0}^{3}\sum_{j=0}^{3} M(i,j)$$

La valeur de la somme ne dépend pas de la façon dont les éléments sont ajoutés (ligne par ligne ou colonne par colonne)

$$s \leftarrow \sum_{j=0}^{3}\sum_{i=0}^{3} M(i,j)$$

Cependant, la performance de l'algorithme en temps d'execution peut dépendre de l'organization des données en mémoire et de l'ordre dans lequel on y accede.

Matrix M(i,j)

M(0,0)	M(0,1)	M(0,2)	M(0,3)
M(1,0)	M(1,1)	M(1,2)	M(1,3)
M(2,0)	M(2,1)	M(2,2)	M(2,3)
M(3,0)	M(3,1)	M(3,2)	M(3,3)

Algorithme et données: un conseil pour minimiser les défauts de cache

- Lorsqu'un algorithme travaille sur des données de type tableau à deux indices, il faut privilégier le parcours du tableau qui tire parti de la localité spatiale

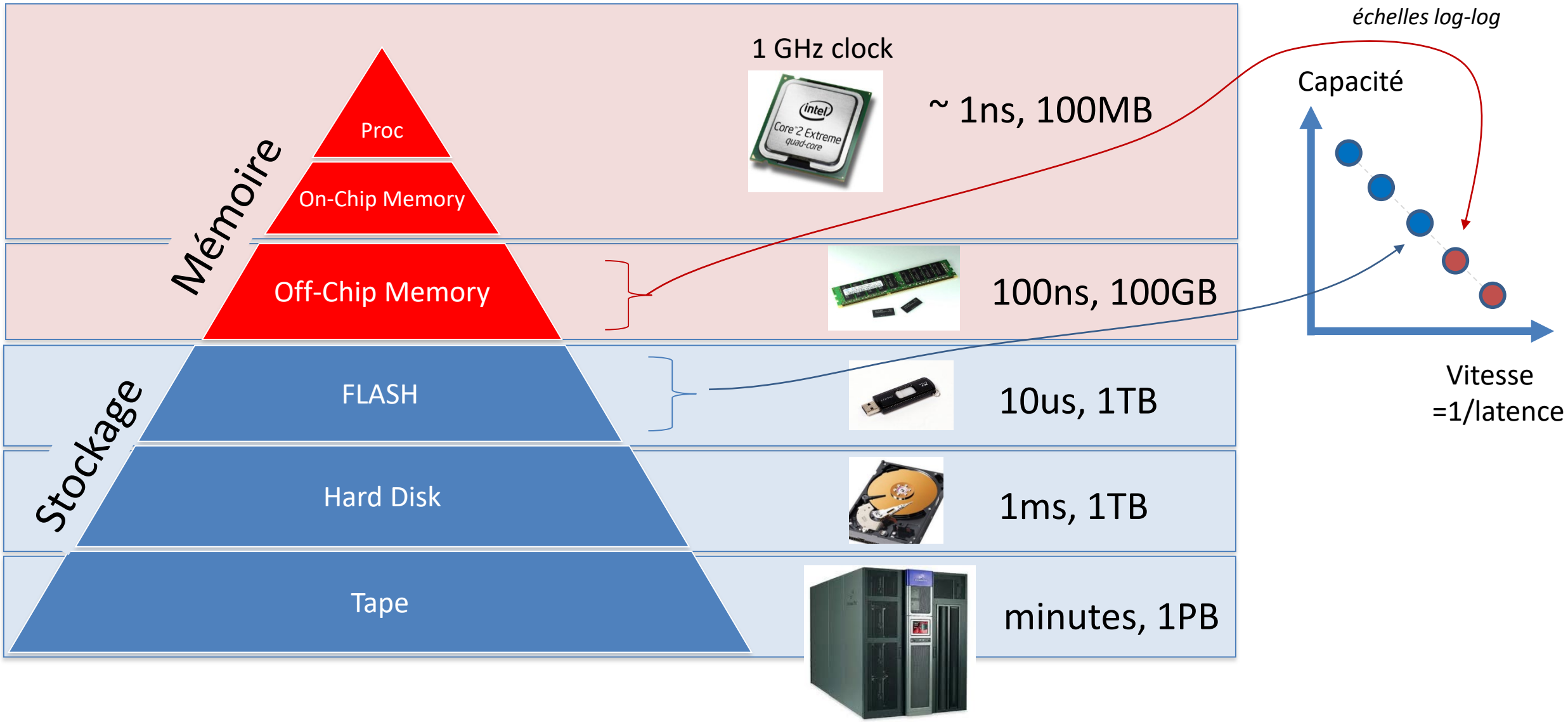
  - Si le tableau est rangé ligne par ligne en mémoire, il y a moins de défaut de cache quand on fait la somme des élément d'une ligne car ils sont généralement dans le *même bloc en mémoire cache*

  - Alors que les éléments appartenant à une même colonne ont une probabilité beaucoup plus grande d'être dans des blocs différents, ce qui produit *beaucoup plus de défauts de cache*

- En C++, quand c'est possible, il faut privilégier de faire une opération ligne-par-ligne plutot que colonne par colonne.

Plan

- Introduction : mémorisation et stockage
- Principe du cache: une mémoire grande et rapide
- Fonctionnement du cache avec le processeur et la mémoire centrale
- Le compromis entre localité spatiale et localité temporelle
- Impact de l'organisation des données sur les performances d'un algorithme
- **Structuration du stockage des données**


Mémoire
Proc
On-Chip Memory
Off-Chip Memory
Stockage
FLASH
Hard Disk
Tape
1 GHz clock
~ 1ns, 100MB
100ns, 100GB
10us, 1TB
1ms, 1TB
minutes, 1PB
échelles log-log
Capacité
Vitesse
=1/latence

2. Structuration du stockage des données

► Dans la hiérarchie de la **mémoire** on *calcule*

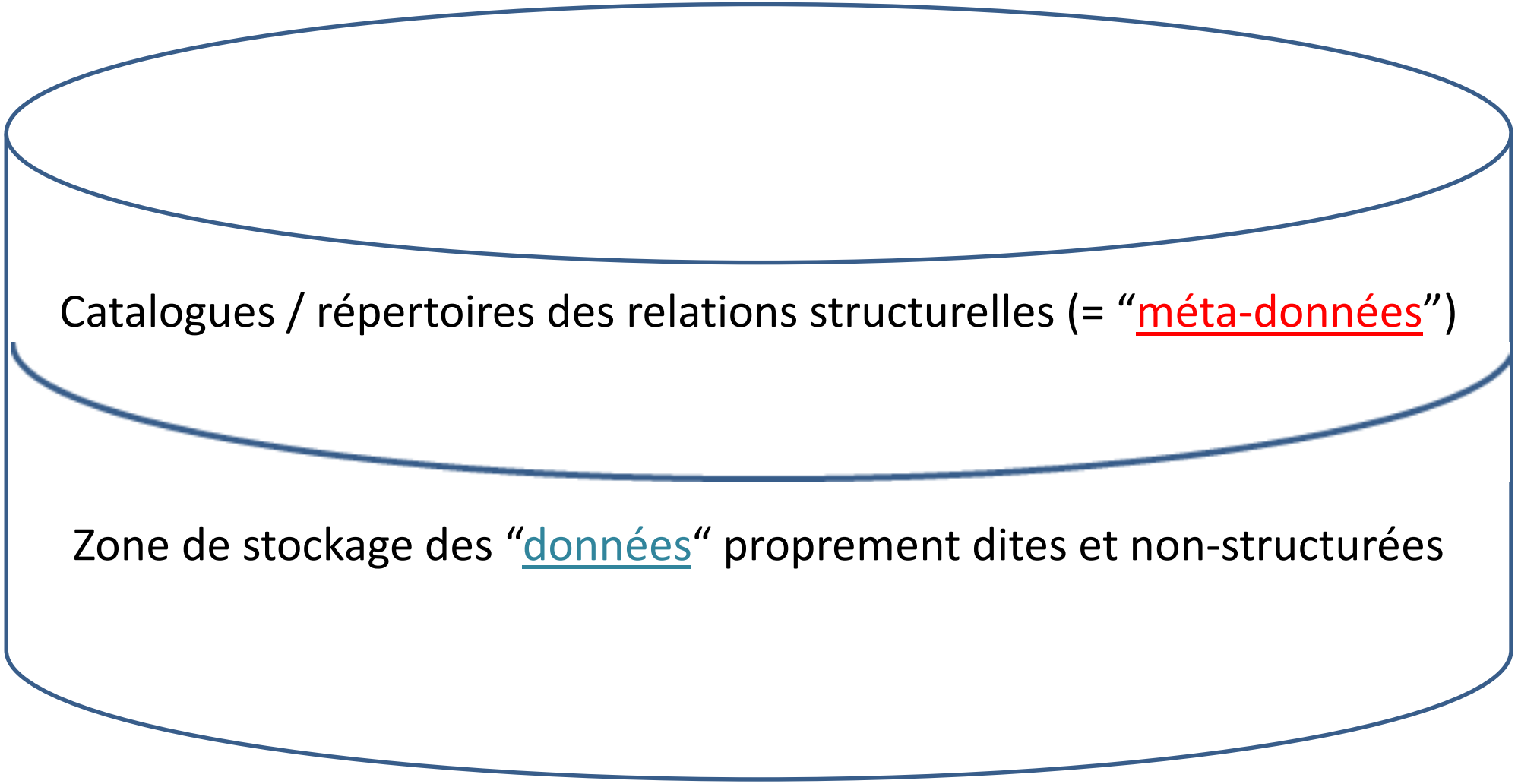
► Dans la hiérarchie du **stockage** on *stocke* et on *récupère* des données

- Les données sont organisées en fichiers (cf série1 C++)

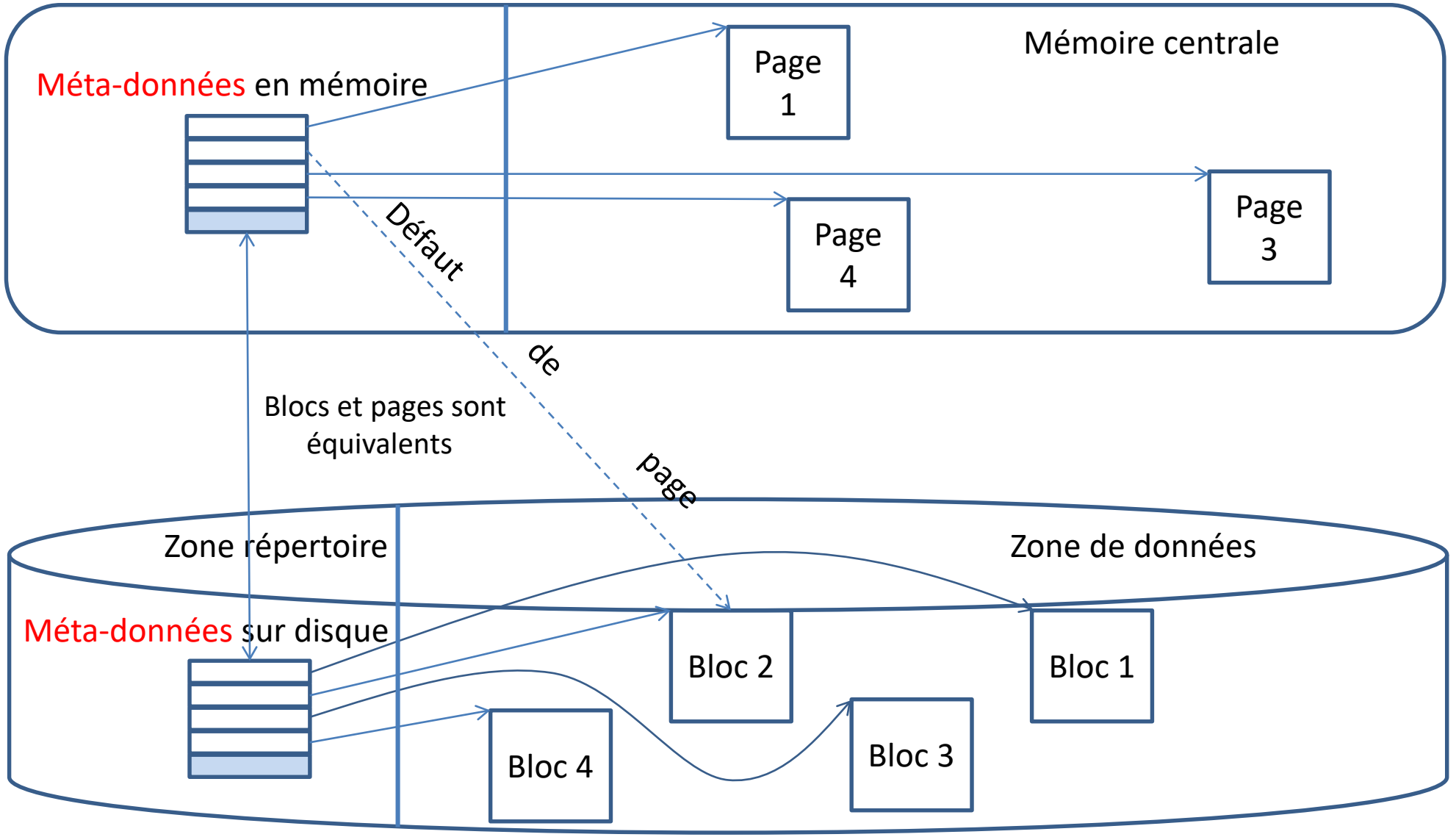
- Nous avons besoin de structure dans le stockage des fichiers
  - On distingue les *données brutes* des *méta-données* => **localisation** des données.

- La **bande passante** des échanges entre **mémoire** et **disque** impose un transfert par **bloc** (= page) de l'ordre de **4 KB** (cette taille n'est pas standardisée)
  - De ce fait, un fichier stocké sur le disque est décomposé en **un ensemble de blocs** de cette taille ; ces blocs vont contenir les données brutes.
  - Une ***carte*** du fichier contient la liste **des adresses des blocs** du fichier
  - Chaque disque contient une zone de *méta-données* rassemblant le *catalogue* des fichiers et la *carte* de chacun des fichiers.

Principe de base de la structuration des données stockées



Mécanisme de cache entre mémoire centrale et disque: role des méta-données pour gérer le transfert d'un bloc (=page)



Mécanisme de cache entre mémoire centrale et disque: role des méta-données pour gérer le transfert d'un bloc (=page)

(cf dessin du slide précédent)

- Lorsqu'un programme veut accéder à une donnée d'un fichier
  - Le bloc de 4KB contenant cette données va être transférée en mémoire centrale
  - MAIS, pour le tout premier bloc, la *carte* du fichier doit d'abord être copiée dans la zone des méta-données de la mémoire centrale.
  - A l'aide de cette carte la mémoire centrale sait quel bloc de 4KB recopier en mémoire centrale (appelé une « page »). Ensuite, de cette page est extrait le bloc de 512 octets demandé par la mémoire on-chip et contenant la donnée demandée.
  - Si ensuite le programme a besoin d'une autre donnée, elle sera peut être dans le cache de la mémoire on-chip, ou dans le bloc chargé en mémoire centrale (page).
  - Si ça n'est pas le cas, il y a *défaut de page* et, grâce à la carte du fichier déjà en mémoire centrale, la mémoire centrale sait quel bloc y charger (ex: bloc 2 dans slide précédent).

Résumé

- Hiérarchies de mémoires
  - Différence de technologie: vitesse vs. capacité
- Deux types de stockage
  - Mémoire pour le calcul
  - Mémoire de stockage pour l'archivage de données
- Cache & localité
  - Permet de rendre la mémoire grande & rapide
- Les algorithmes peuvent aussi conduire à des performances différentes
  - Par ex., 4x entre accès d'une matrice par lignes & colonnes
- Stockage : les méta-données mémorisent la structuration des données
  - Le mécanisme de cache entre mémoire et disque s'appuie sur les méta-données (la *carte* du fichier)

Information, Calcul et Communication

Réseaux

Faculté Informatique et Communications

A. Ailamaki, P. Janson, W. Zwaenepoel

Plan

- **La notion de protocole de communication**
- La commutation par paquets
- La structure d'un protocole et l'entête d'un paquet
- L'Internet
- La modularisation des protocoles
- Exemple: les protocols de l'Internet (TCP/IP)

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 électrique réel ou juste du bruit ?
  Comment le convertir en un signal numérique binaire ?
  Comment savoir où il commence et où il finit ?
  Comment savoir d'où il vient ?
  Comment interpréter ce qu'il contient ?

un protocole de communication
structure la transmission de données

Un protocole de communication

- Un jeu de règles qui structure une communication

La notion de protocole

- Toute communication est gouvernée par un protocole
- Y compris, la communication entre êtres humains, p. ex.,
  - Salle de cours
  - Au téléphone
  - Message écrit

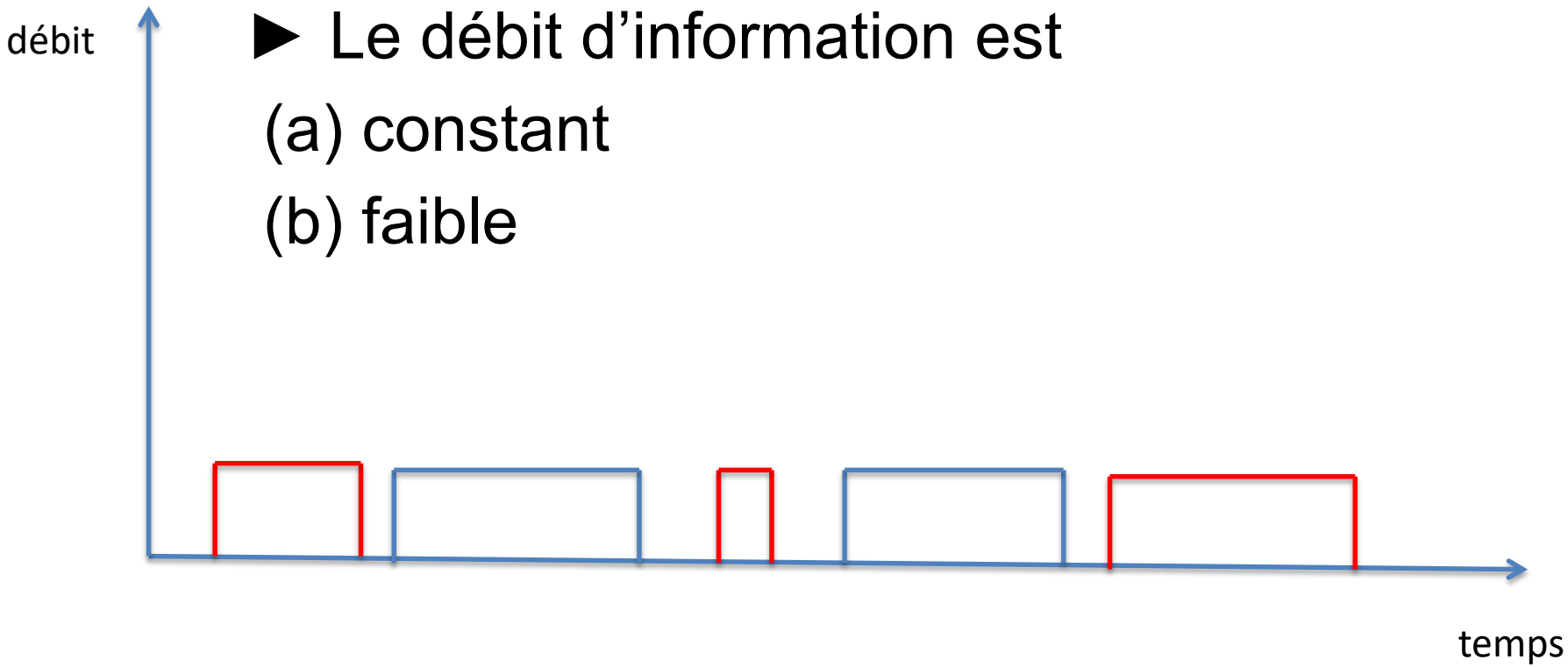
La différence avec la communication entre ordinateurs

- Un protocole entre êtres humains peut être vaguement défini
- Un protocole entre ordinateurs doit être fixé dans tous les details

Plan

- La notion de protocole de communication
- **La commutation par paquets**
- La structure d'un protocole et l'entête d'un paquet
- L'Internet
- La modularisation des protocoles
- Exemple: les protocoles de l'Internet (TCP/IP)

La communication par téléphone



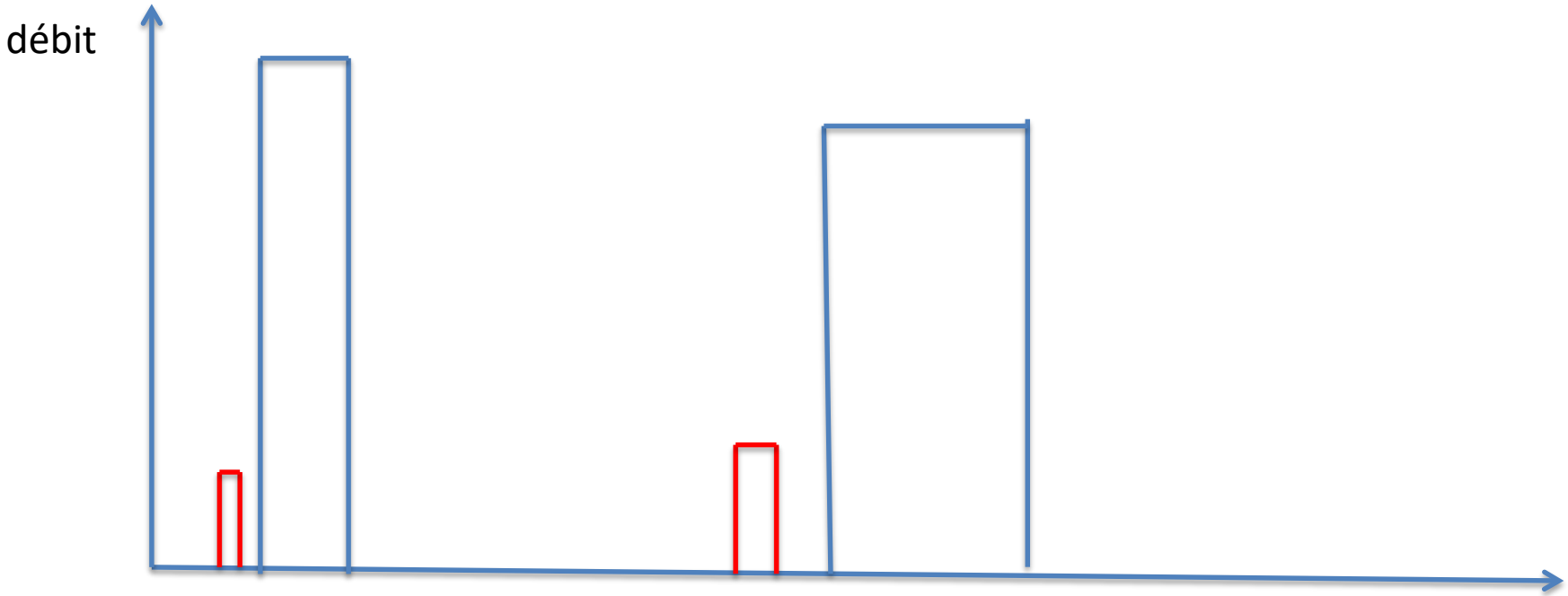
► Le débit d'information est

(a) constant

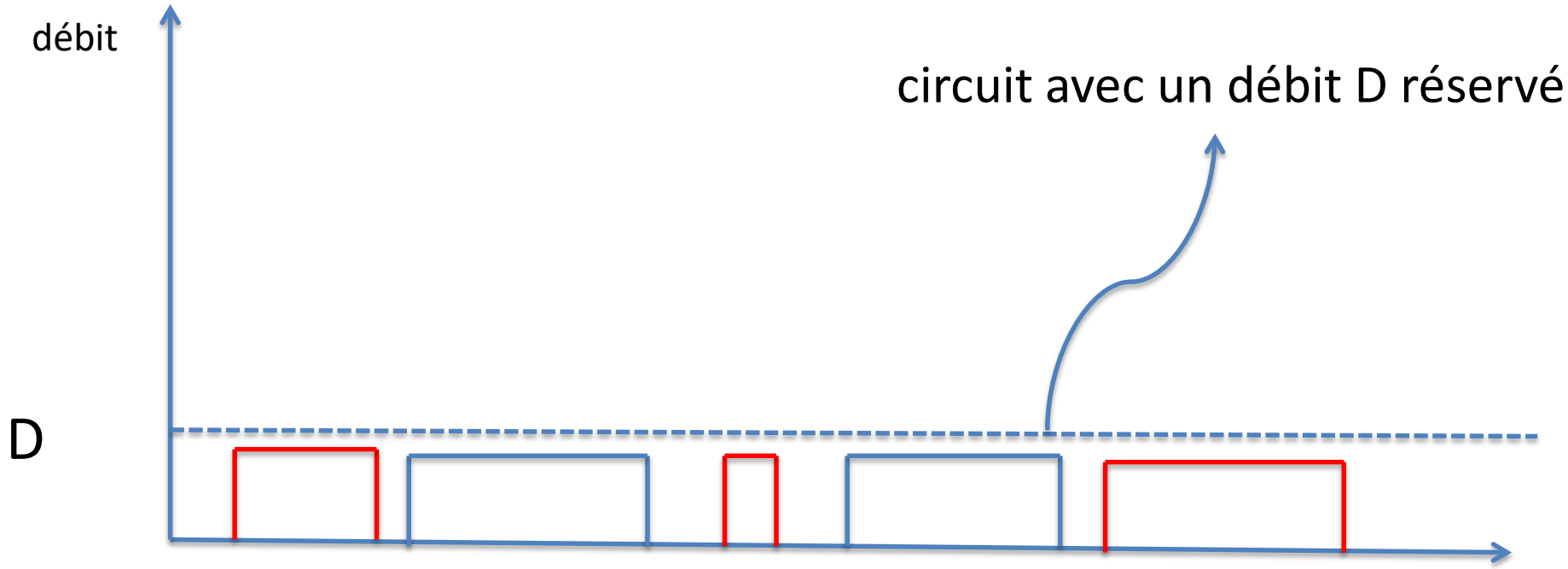
(b) faible

La communication entre ordinateurs

- L'information arrive en *rafale* (périodes de volume faible suivies par des hauts et soudains volumes d'information)
- Le débit est tres élevé quand il y a communication
- Le débit est nul autrement

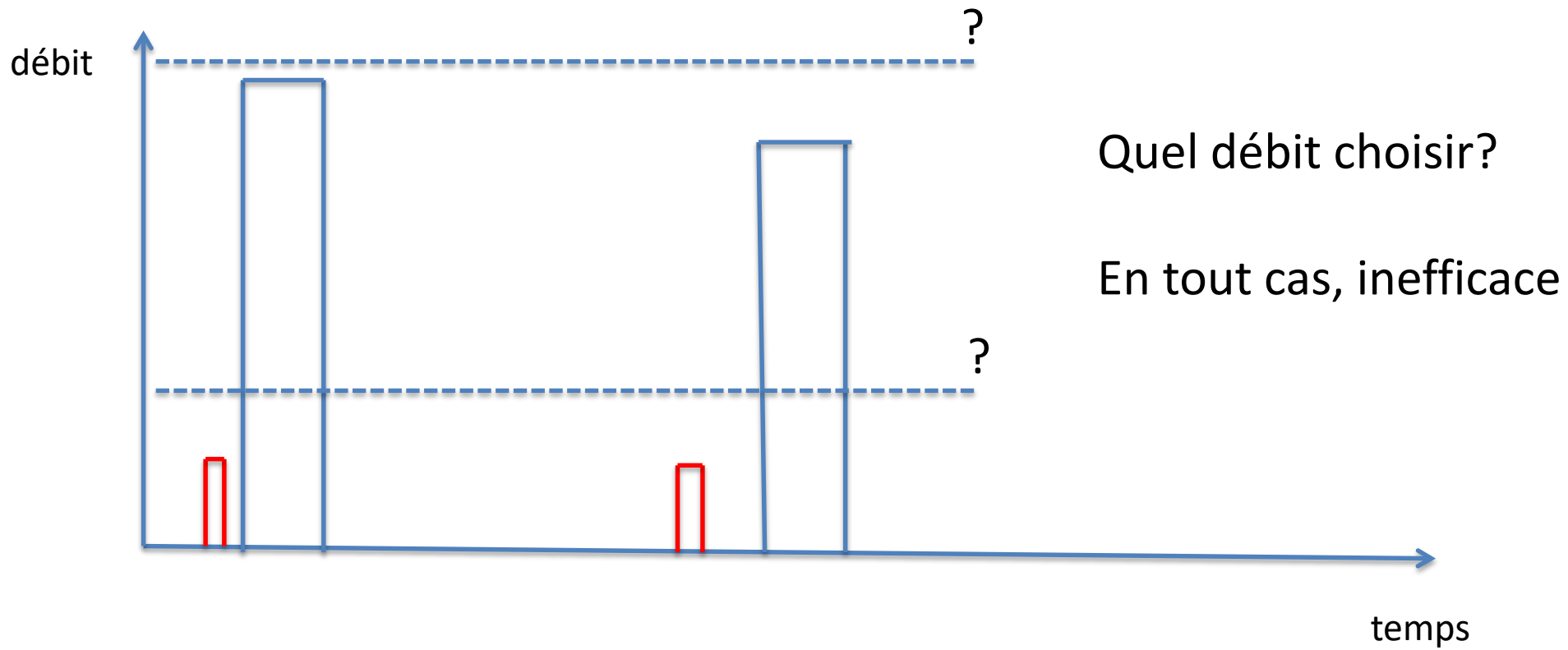


La commutation de circuits dans la téléphonie



- Réserver un “circuit” entre source et destination
- “Commutation de circuits”

Pas approprié pour la communication par ordinateur



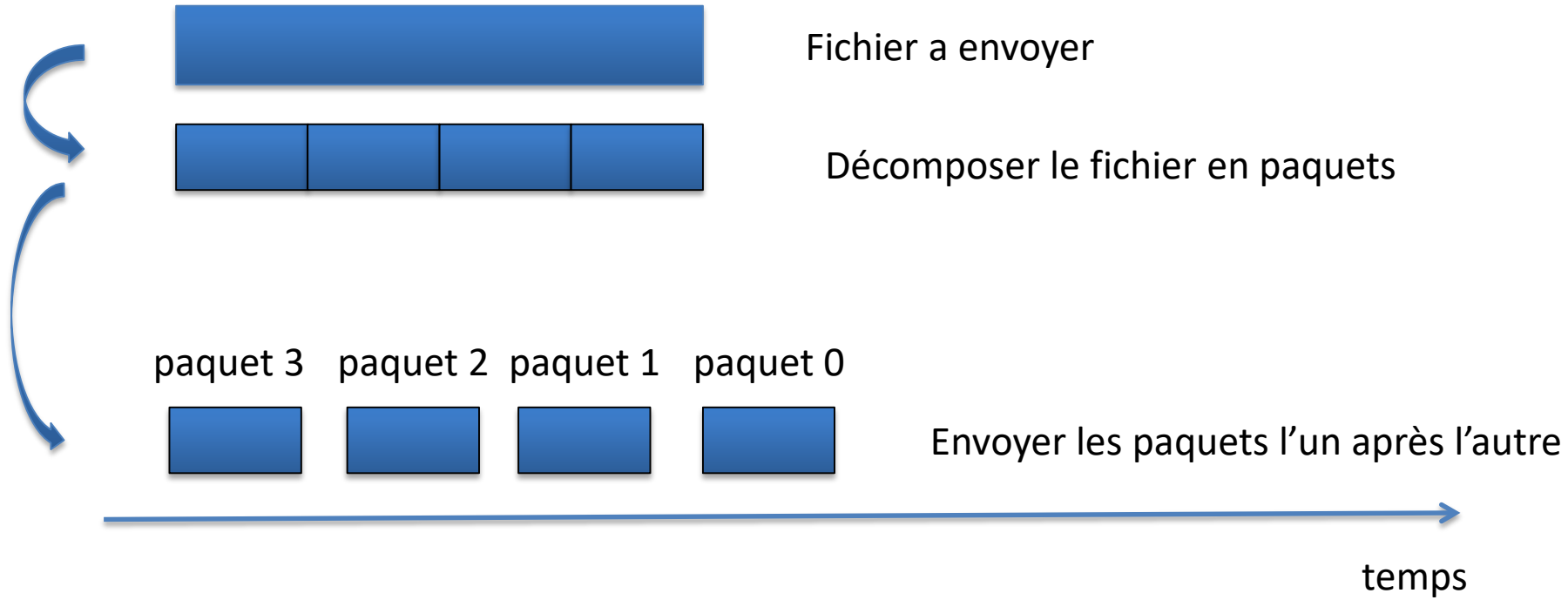
Quel débit choisir?

En tout cas, inefficace

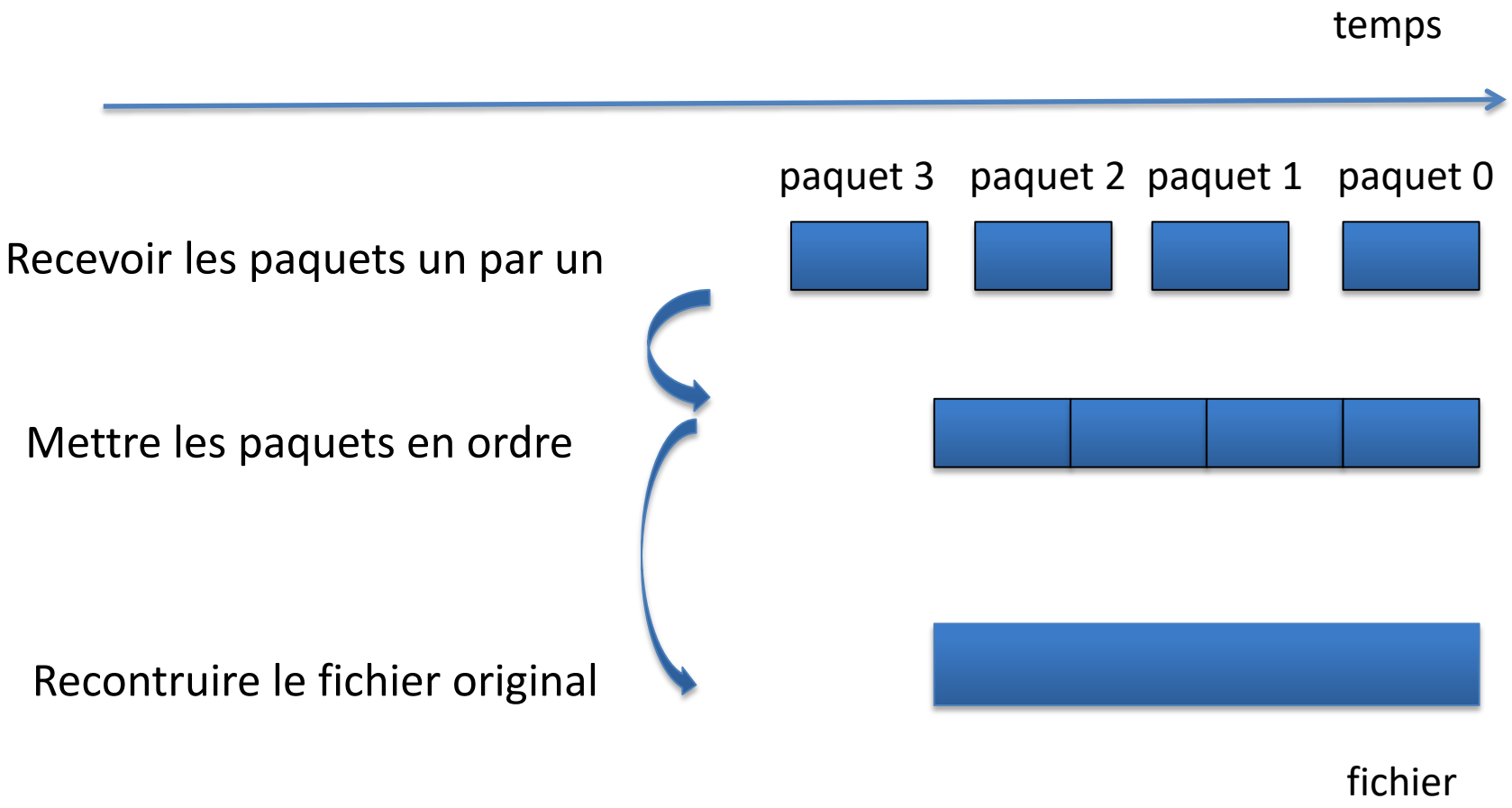
►La réservation d'un circuit serait très inefficace

Commutation par paquets

- On découpe l'information en paquets
- On les envoie à destination, un paquet à la fois
- Modèle "La Poste"



Exemple: comment recevoir un fichier



Commutation par paquets est omniprésente

- L'internet
- Presque tout réseau pour ordinateur
- Même la téléphonie (Skype!)
- Protocole: le jeu de règles
  - Exemple: WiFi, Ethernet, …

Plan

- La notion de protocole de communication
- La commutation par paquets
- **La structure d'un protocole et l'entête d'un paquet**
- L'Internet
- La modularisation des protocoles
- Exemple: les protocoles de l'Internet (TCP/IP)

Un exemple de protocole

- Disons que l'ordinateur `A` veut
  - Envoyer un paquet à l'ordinateur `B`
  - Savoir si le paquet est arrivé

Le principe d'un acquittement (ACK)

- `B` envoie à **`A`** un paquet de type spécial qui s'appelle paquet **d'acquittement** (ACK qui vient du mot anglais acknowledgement)

- ACK signifie "c'est ok, je l'ai recu"

Attention!

- **A** peut confondre un paquet d' acquittement avec un paquet des données!

- Il faut maintenant distinguer
  - Les paquets avec des **données**
  - Les paquets **d'acquittement**

- Comment faire?
  - On met dans le paquet une information de "type"

type = paquet de **données** ou paquet **d'acquittement**

Mais alors ?

- Que fait A quand l'acquittement n'arrive pas?

Le principe de la retransmission

- Après un temps t, A retransmet le paquet à B
- Jusqu'à ce qu'un acquittement soit reçu
- Apres un nombre d'essais max, A abandonne

Pendant ce temps, du coté de la destination...

- Comment B sait qu'un paquet est :
  - Une retransmission ?
  - Un nouveau paquet ?

- Il faut **identifier** le paquet

- On met dans le paquet un **numéro de séquence**

Inclusion du numéro de séquence

seq# = 0, 1, 2, …

Adressage

- $\mathbb{A}$ doit indiquer que le paquet est destiné à $\mathbb{B}$
- $\mathbb{A}$ doit indiquer qu'il est à l'origine du paquet
- Mettre les adresses de $\mathbb{A}$ et $\mathbb{B}$ dans le paquet

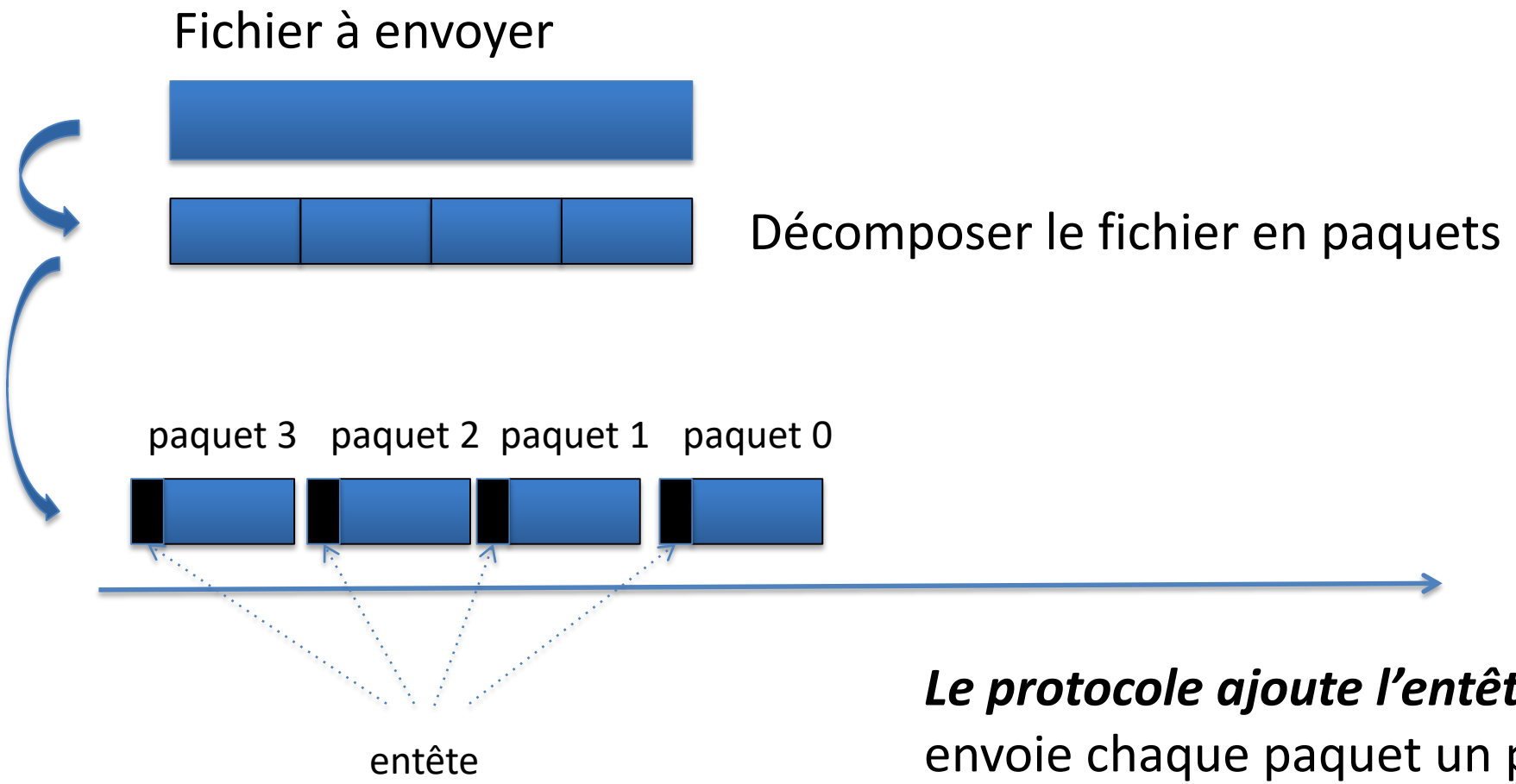
Inclusion d'adressage

L'entête (header) d'un paquet

- Chaque paquet consiste en :
  - Des **données**
  - D'un **entête**: type, numéro de séquence, taille, adresse source et destination, etc.

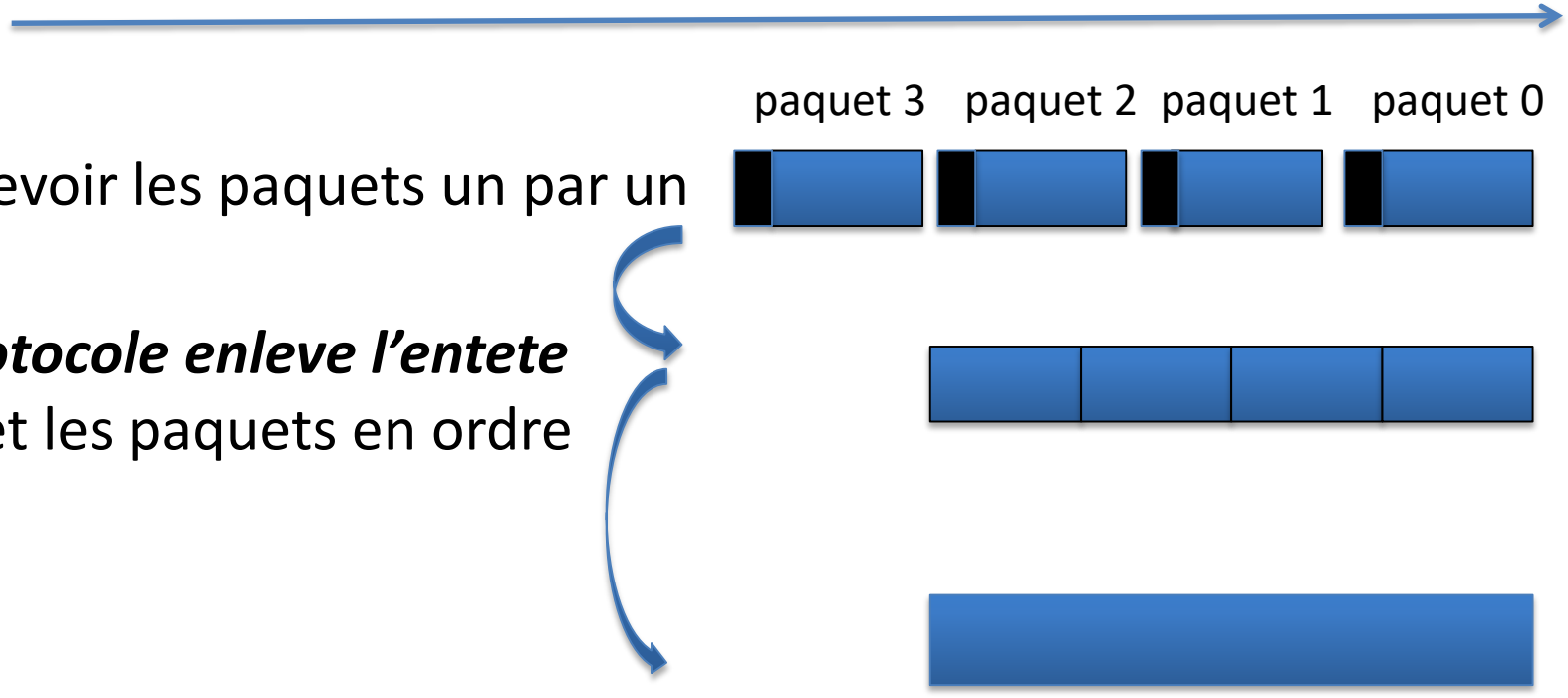
Structure du paquet

Envoi d'un fichier en paquets



Le protocole ajoute l'entête et envoie chaque paquet un par un

Réception d'un fichier en paquets



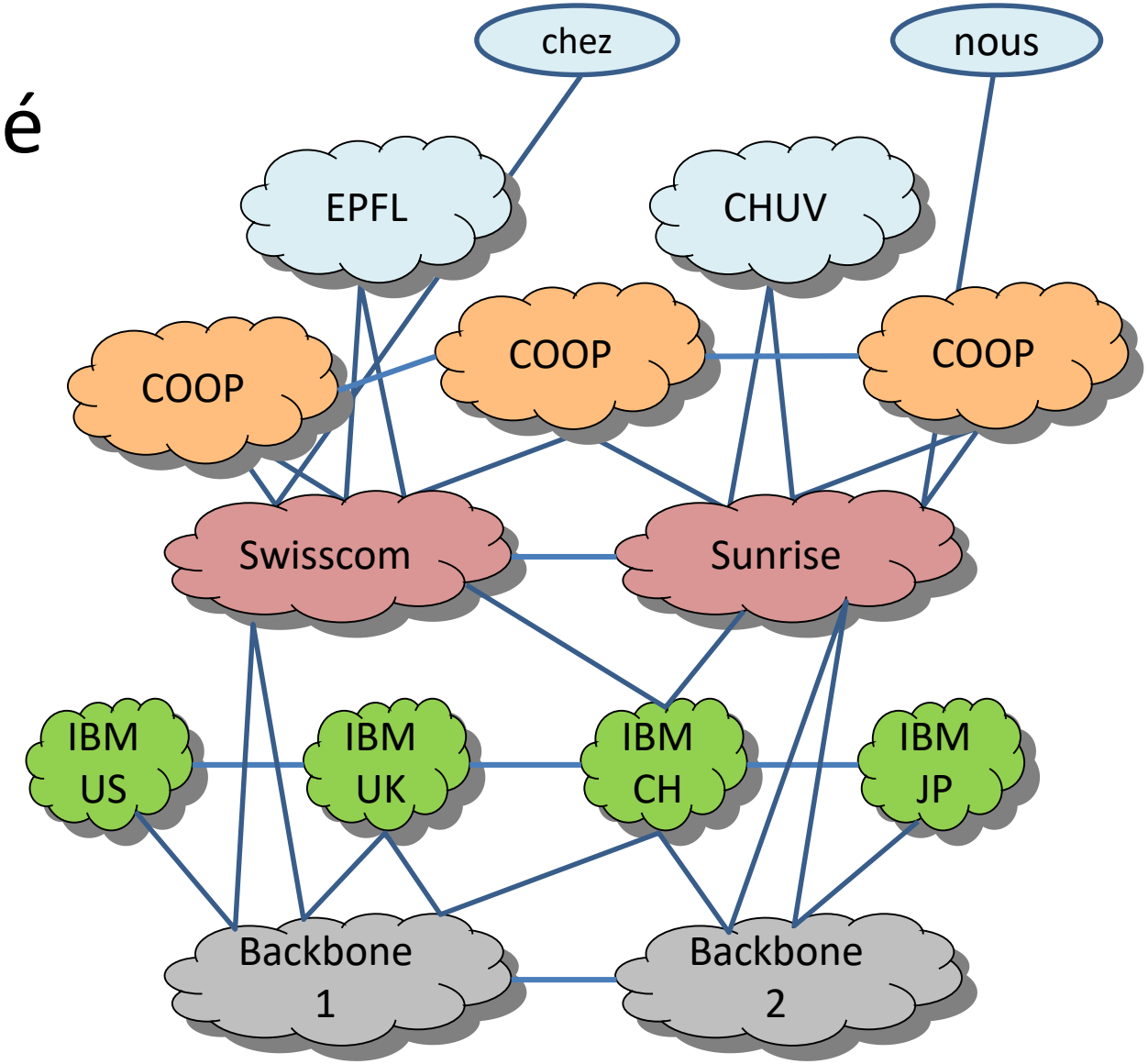
Ce qui permet de recontruire le fichier original

Plan

- La notion de protocole de communication
- La commutation par paquets
- La structure d'un protocole et l'entête d'un paquet
- **L'Internet**
- La modularisation des protocoles
- Exemple: les protocole de l'Internet (TCP/IP)

L'Internet

- L'internet est composé
  - Des réseaux nationaux
  - Des réseaux des ISP
  - De réseaux "backbone"
  - …



Comment gérer cette complexité?

- Chaque réseau a son propre protocol
  - WiFi, Ethernet, …
- Comment faire pour communiquer entre deux machines sur différents réseaux?
- Comment distinguer traffic Web d'autre traffic?
- Comment distinguer le traffic Web d'un utilisateur particulier?

Mise en oeuvre du Principe d'Abstraction
et de la décomposition top-down des problèmes
à l'échelle des systèmes de communications

Les couches = L'abstraction des protocoles

A chaque niveau son protocole – Exemple téléphonique

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

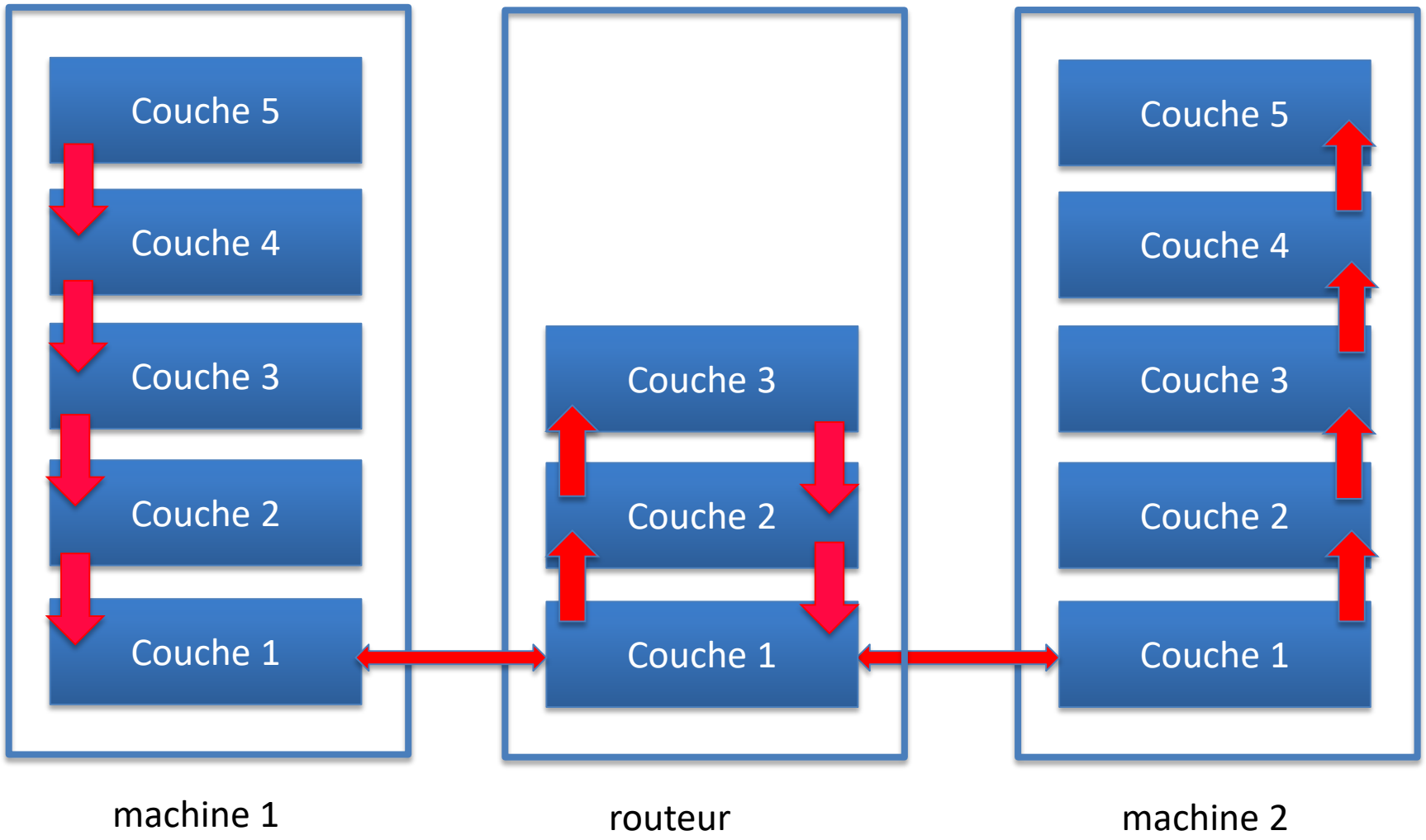
Plan

- La notion de protocole de communication
- La commutation par paquets
- La structure d'un protocole et l'entête d'un paquet
- L'interface et le pilote réseau
- L'Internet
- **La modularisation des protocoles**
- Exemple: les protocoles de l'Internet (TCP/IP)

La modularisation: c’est quoi?

- Le protocole est divisé en couches
- Chaque couche a
  - Sa propre fonction
  - Son propre entête dans le paquet transmis
  - Son propre logiciel
- Les couches se reposent l’une sur l’autre

La réalité est plus compliquée



Le principe d'encapsulation - Transmission

- Chaque couche a son entête
- La fonction de la couche N:
  - Prend le paquet venant de la couche supérieure N+1
    (qui inclut les entêtes des couches N+1, N+2, …)
  - Met un nouveau entête pour sa propre couche
- La couche N>1 passe le paquet au niveau N-1
- La couche N=1 transmet le paquet sur le réseau

Plan

- La notion de protocole de communication
- La commutation par paquets
- La structure d'un protocole et l'entête d'un paquet
- L'Internet
- La modularisation des protocoles
- **Exemple: les protocoles de l'Internet (TCP/IP)**

Structure d'Internet – Protocoles

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	

Couche 3 – “Réseau”

- Entre deux machines n’importe où
- IP (Internet Protocol)
- Responsable pour le routage

La couche IP en plus de détail

- Envoyer un paquet
  - D’une machine quelconque
  - A une machine quelconque

- Une machine $\mathbb{A}$ a une adresse IP **unique**, IP_A
- Deux versions:
  - IPv4: 32 bits - 2^{32} (~4.10^9) systèmes
  - IPv6: 128 bits - ~$256.10^9.10^9.10^9.10^9$ systèmes

A veut envoyer un paquet à B

- La couche IP de A met IP_A et IP_B dans l'entête IP

Le problème du routage

- Comment trouver une voie de A à B?
- Comment trouver le chemin le plus court?

Routage IP

- Un "tableau de routage" pour chaque noeud
  - Pour chaque destination, quel chemin à suivre
  - Quelle est la distance à la destination par ce chemin

Exemple de réseau

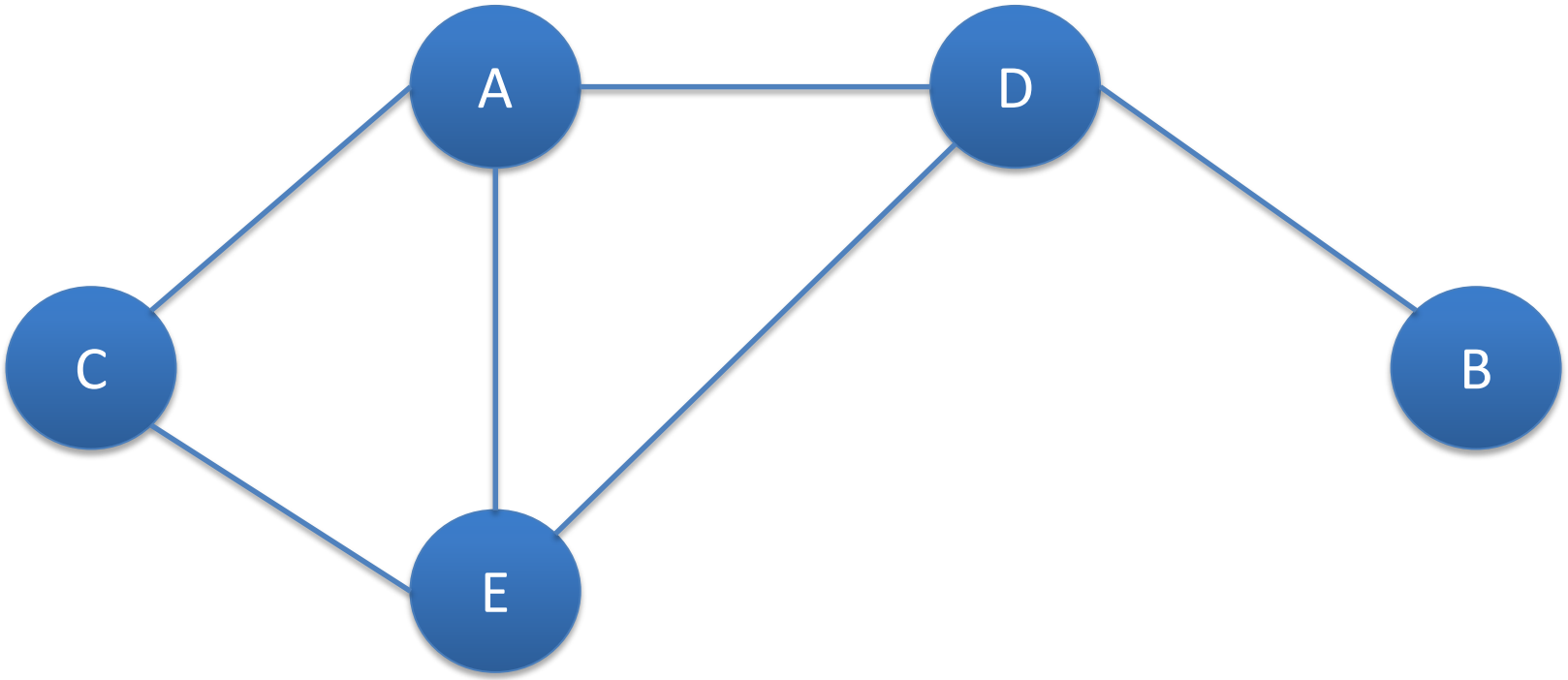


Table de routage de A

dest	voie	longeur
B	D	2
C	C	1
D	D	1
E	E	1

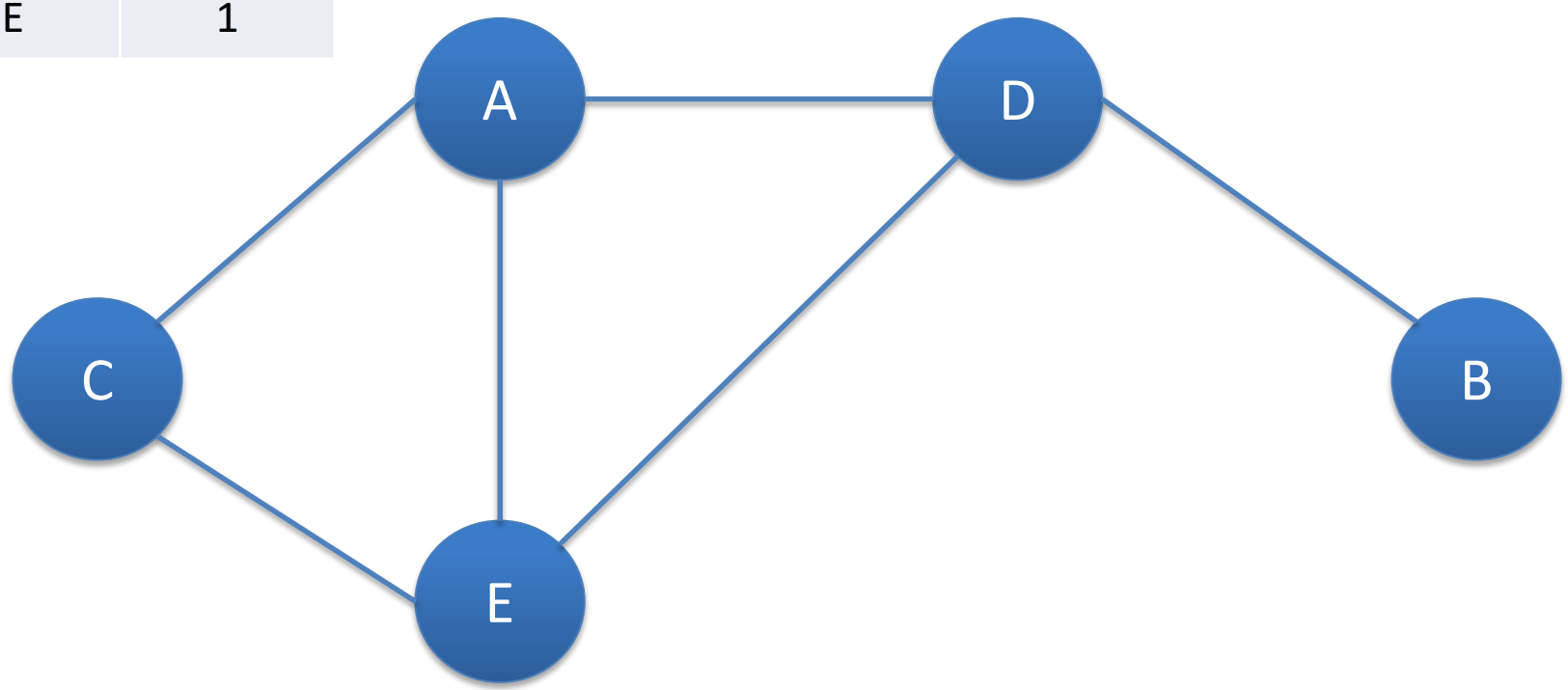


Table de routage de D

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

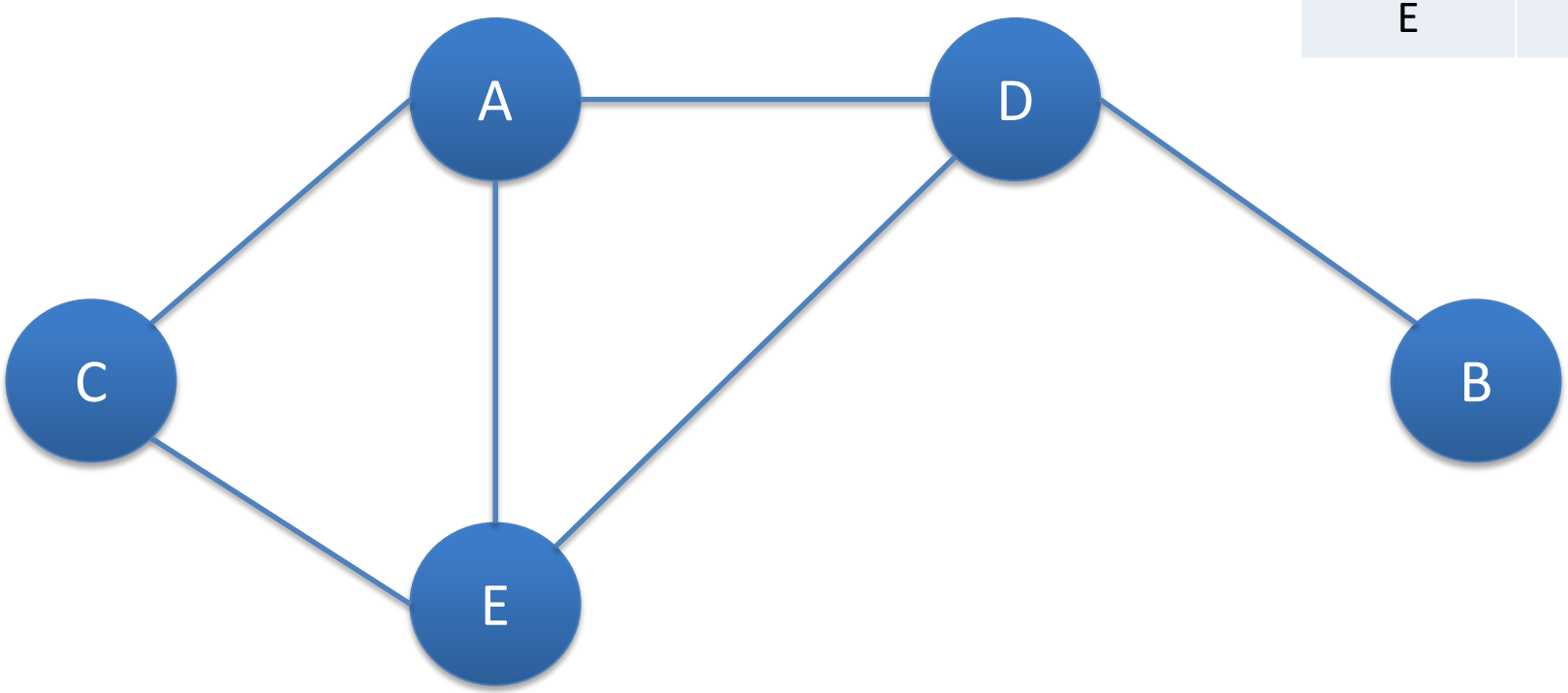
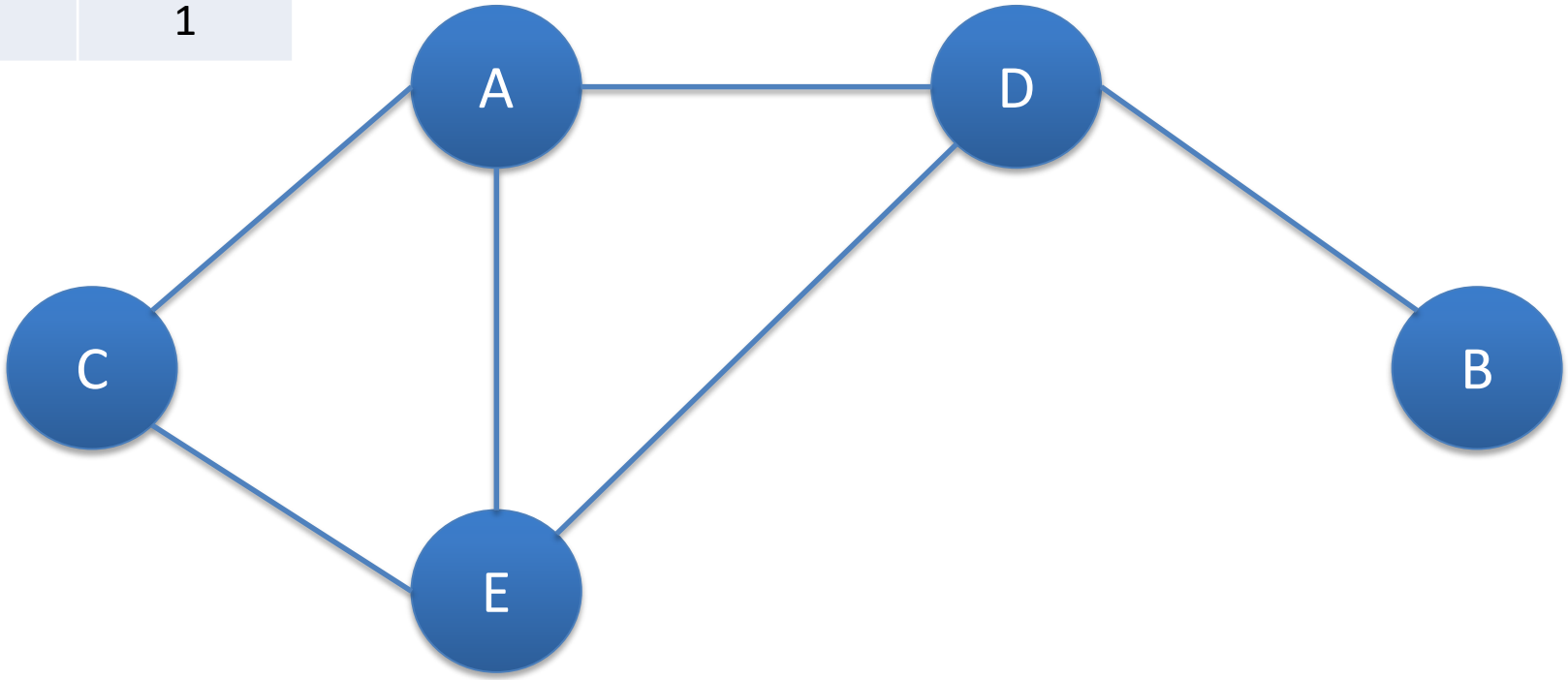


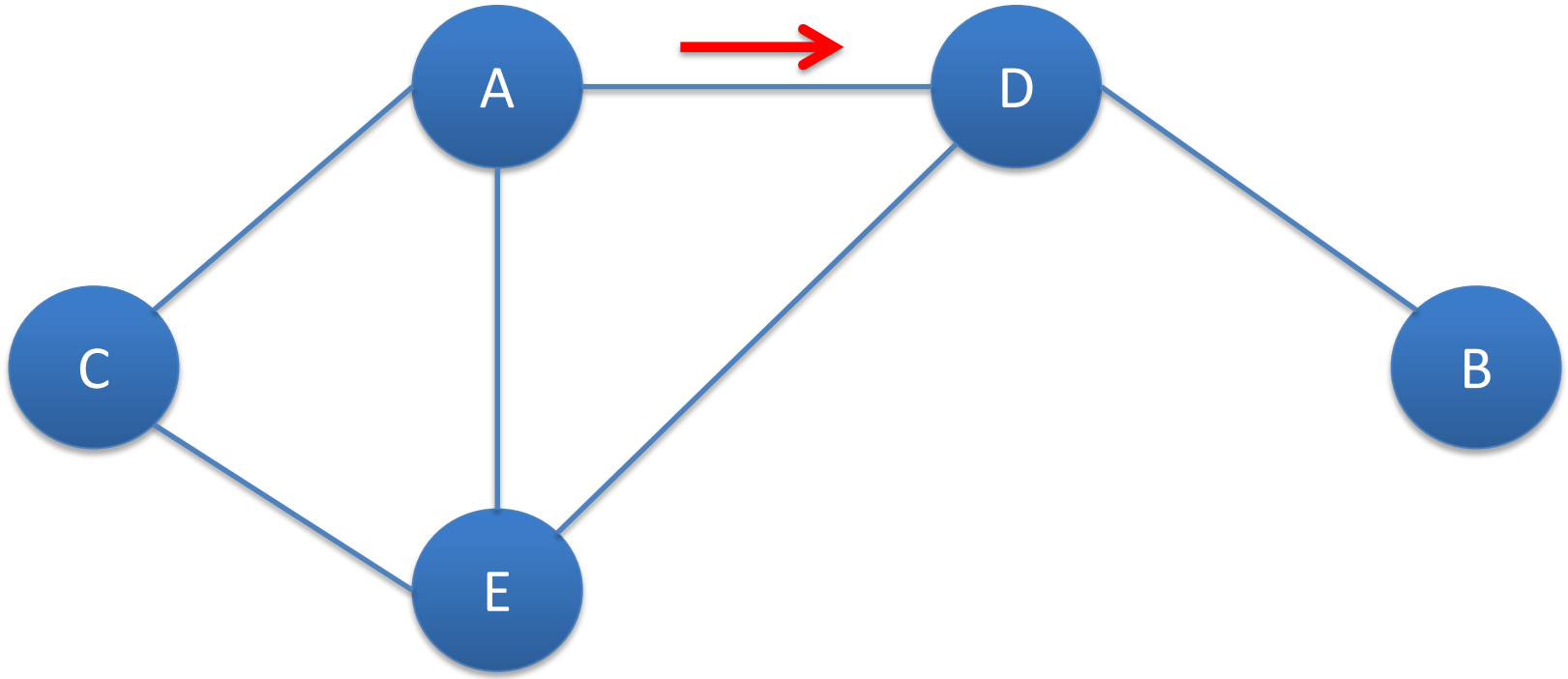
Table de routage de C

dest	voie	longeur
A	A	1
B	A/E	3
D	A/E	2
E	E	1



A veut envoyer un paquet à B

dest	voie	longeur
B	D	2
C	C	1
D	D	1
E	E	1

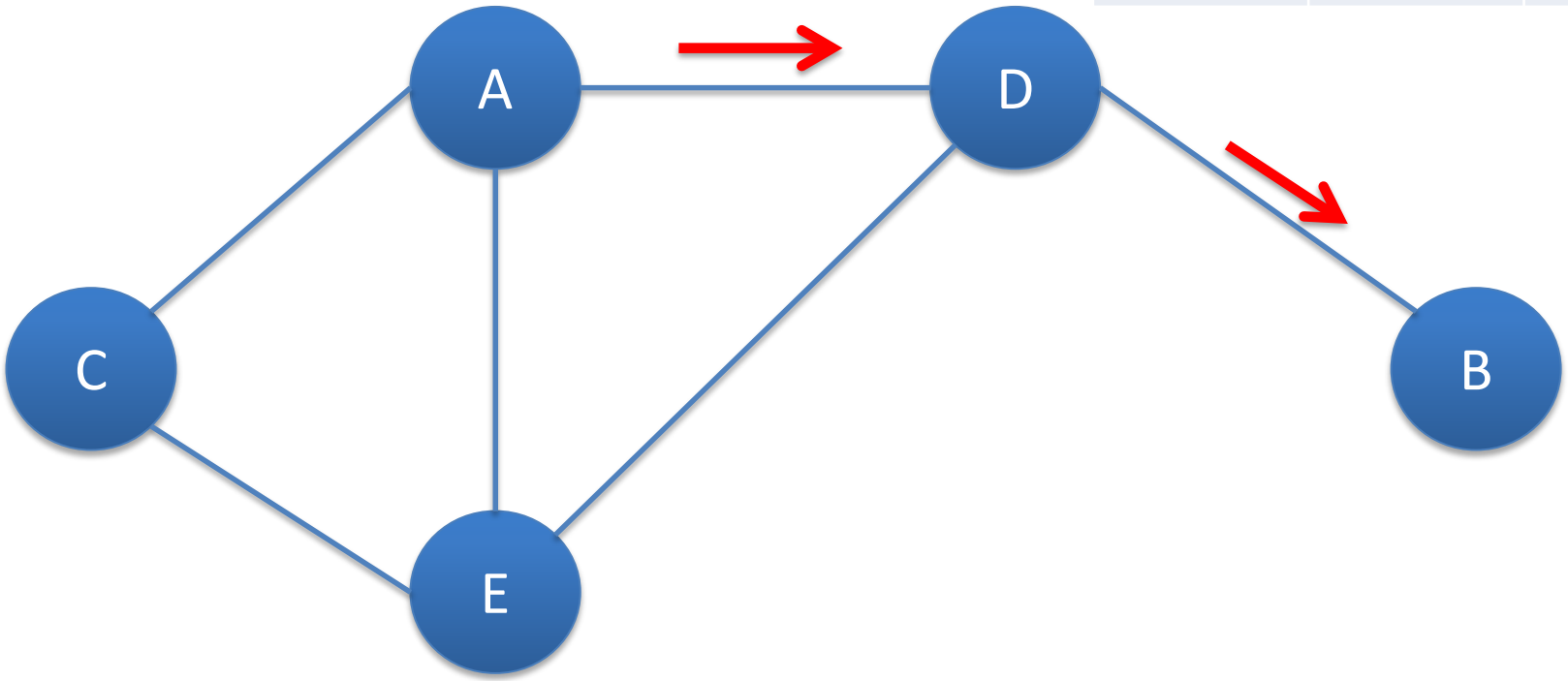


A chaque étape

- On regarde l'adresse IP de la destination
- On regarde le tableau de routage
- On envoie le paquet au noeud indiqué dans la table pour cette destination

D forward le paquet à B

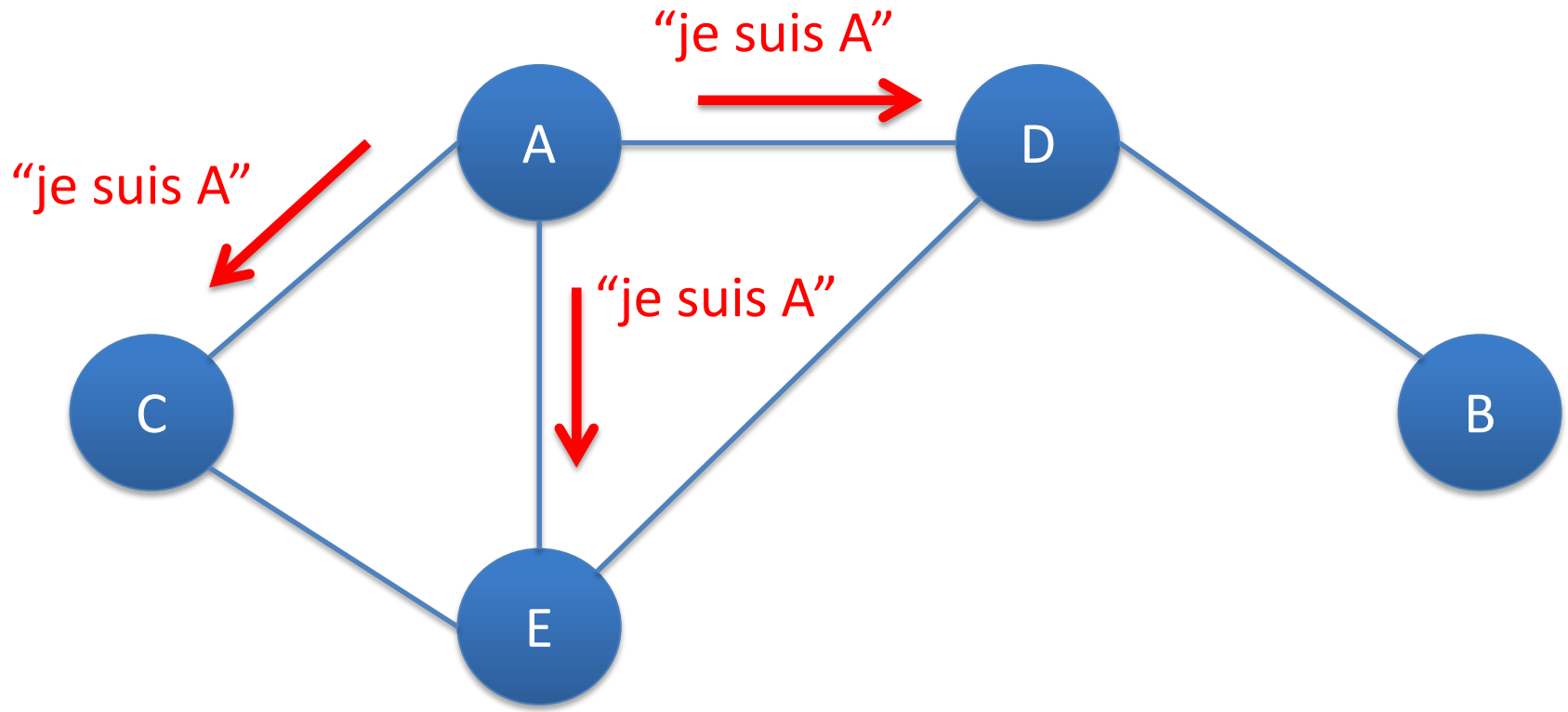
dest	voie	longeur
A	A	1
B	B	1
C	A/E	2
E	E	1



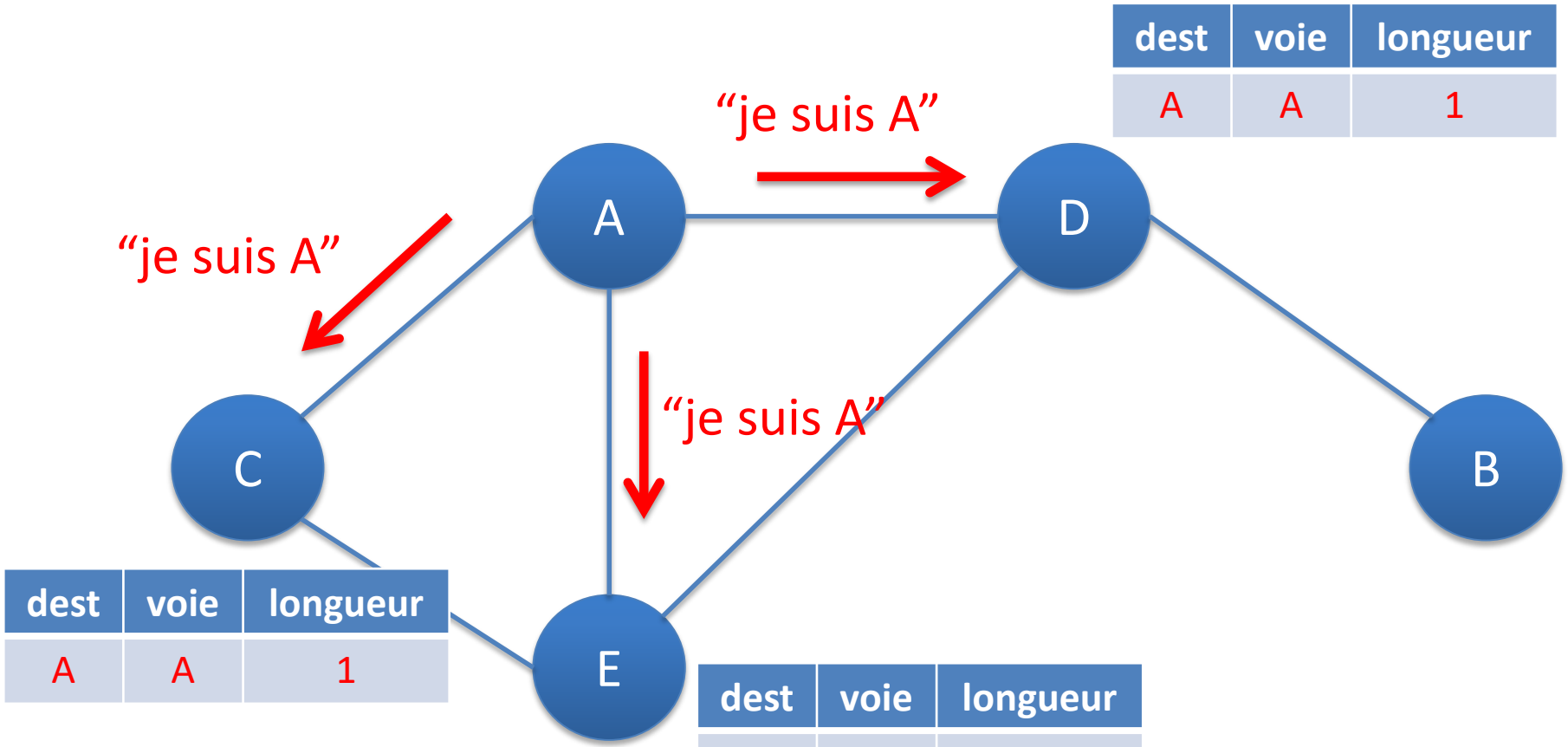
Comment construire le tableau de routage?

- Le routage se fait par “ouï-dire”
- Chaque noeud annonce à ses voisins la “longueur” des chemins qu’il connaît
- Chaque noeud retient et propage le chemin le plus court parmi ceux annoncés par ses voisins

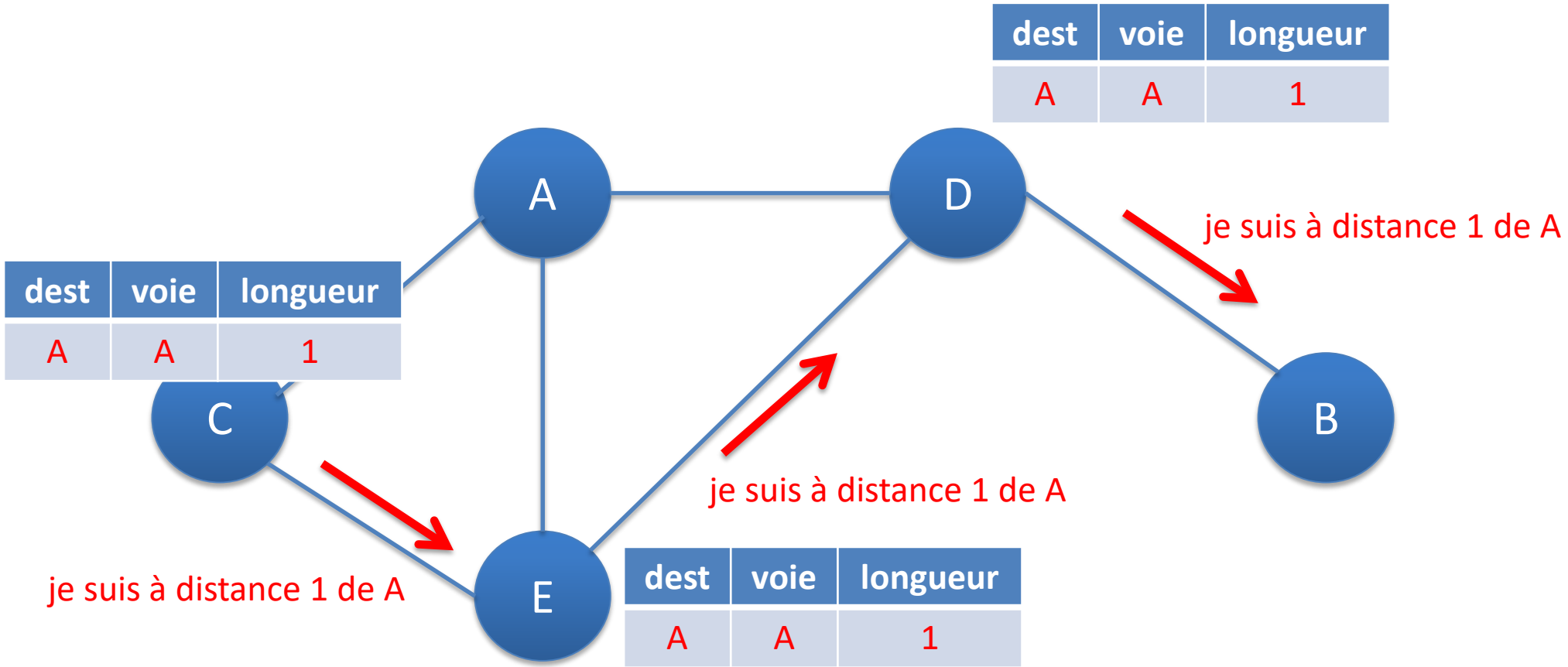
Par exemple



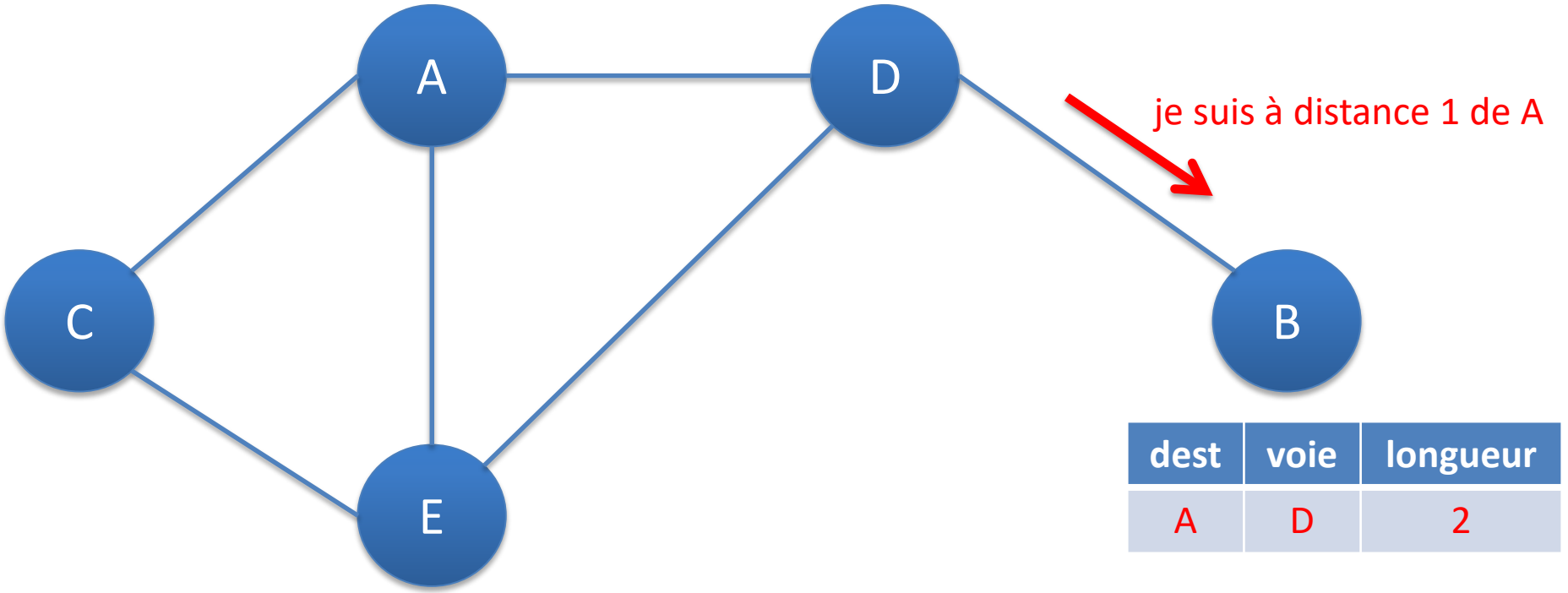
Par exemple


"je suis A"
"je suis A"
"je suis A"
A
D
C
B
E
dest voie longueur
A A 1
dest voie longueur
A A 1
dest voie longueur
A A 1

Par exemple


dest voie longueur
A A 1
A
D
je suis à distance 1 de A
dest voie longueur
A A 1
C
B
je suis à distance 1 de A
je suis à distance 1 de A
E
dest voie longueur
A A 1

Par exemple



dest	voie	longueur
A	A	1

dest	voie	longueur
A	D	2

Résumé

- La notion de protocole de communication
  - Jeu de règles pour la transmission des données au sein d'un seul ordinateur ou entre plusieurs ordinateurs

- La commutation par paquets
  - Grouper les données dans des paquets de taille fixe
  - Plusieurs types des paquets sont nécessaires pour l'implementation des protocoles
  - Entêtes des paquets

- Les couches des protocoles
- Exemple: le protocole de l'Internet (TCP/IP)