

# Les réseaux

## 3

### Couche Réseau

D'après le livre :

*Analyse structurée des réseaux*

Jim Kurose, Keith Ross  
*Pearson Education*

Adaptation : **AbdelAli ED-DBALI** ([AbdelAli.Ed-Dbali@lifo.univ-orleans.fr](mailto:AbdelAli.Ed-Dbali@lifo.univ-orleans.fr))



# Chapitre 4 : Couche Réseau

## Buts du chapitre:

- ♦ comprendre les principes derrière les services de la couche réseau :
  - ♦ routage (sélection de chemins) hiérarchique
  - ♦ IP (Internet Protocol)
  - ♦ IPv6

# Chapitre 4 : feuille de route

## 4.1 Introduction

## 4.2 Principes du routage

## 4.3 Routage hiérarchique

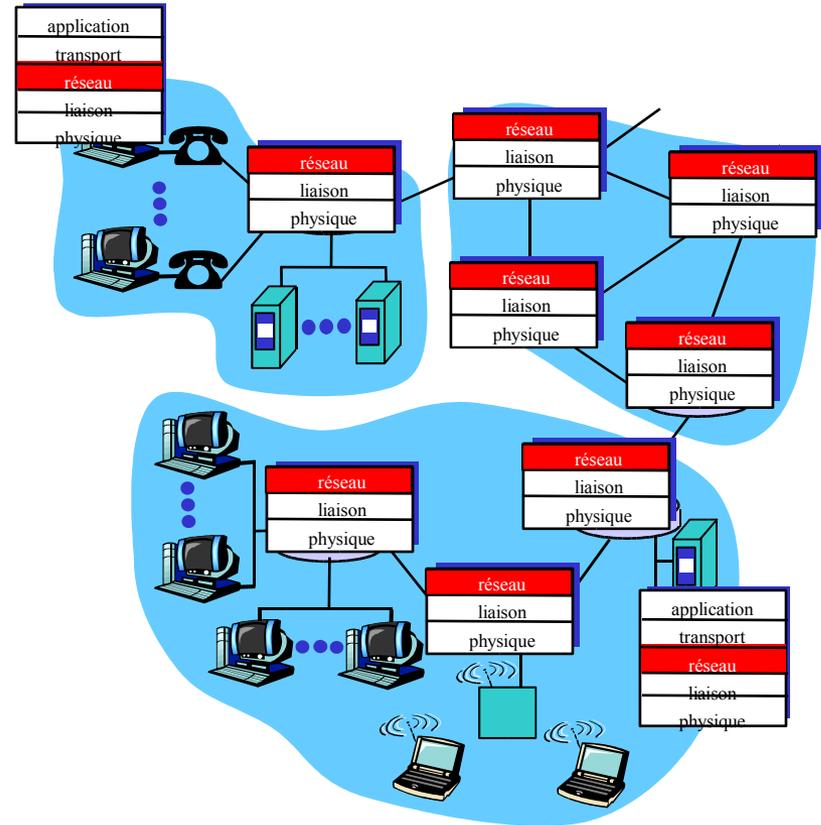
## 4.4 IP (Internet Protocol)

## 4.5 Routage dans l'Internet

## 4.6 IPv6

# Fonctions de la couche réseau

- ♦ acheminer les paquets de l'émetteur jusqu'au récepteur
- ♦ les protocoles de la couche réseau résidents dans *tout* poste, routeur



## Trois fonctions importantes :

- ♦ *détermination de chemin* : route prise par les paquets depuis la source à la dest.  
Algorithmes de routage
- ♦ *transit* : déplacer des paquets en entrée vers la bonne sortie du routeur
- ♦ *call setup* : certains réseaux demandent l'établissement d'une connexion le long du chemin de routeurs avant le transfert de données

# Modélisation du service

**Q:** Quel *modèle de service* pour le transport, dans un “canal”, de paquets depuis l'émetteur au récepteur?

- ♦ Garantie de bande passante?
- ♦ préservation du timing inter-paquets?
- ♦ livraison sans perte?
- ♦ livraison dans l'ordre?
- ♦ feedback à l'émetteur sur la congestion?

abstraction du service

**La** plus importante abstraction prévue par la couche réseau :

*circuit virtuel*  
ou  
*datagramme?*

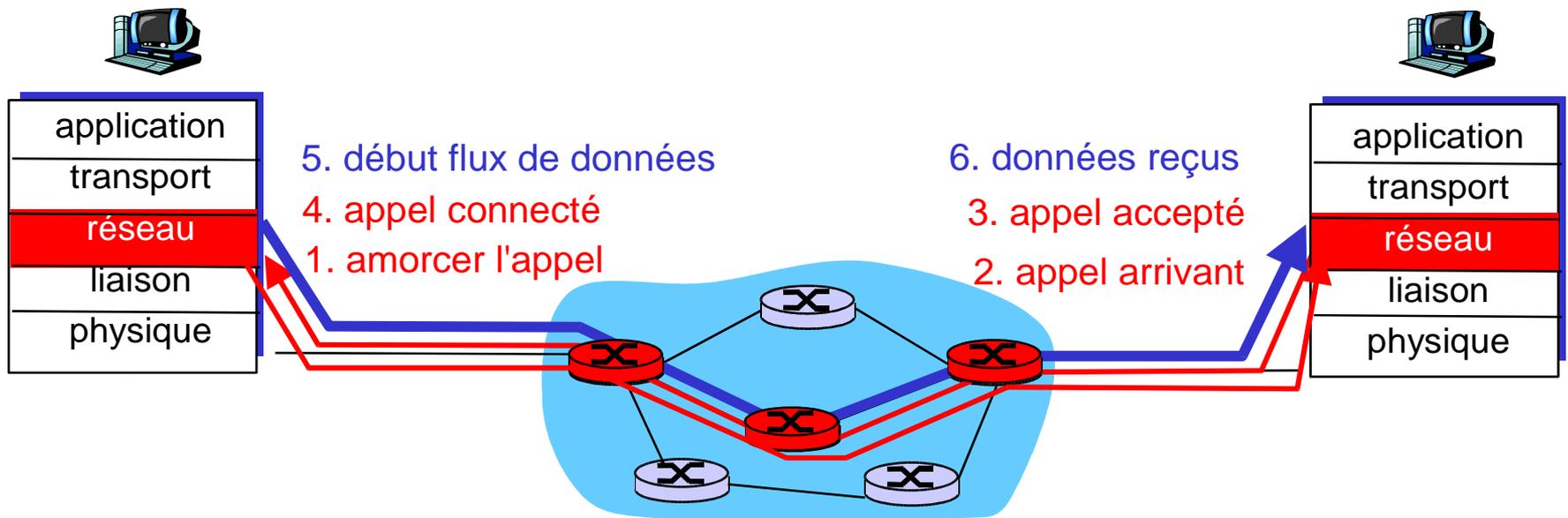
# Circuit virtuel

“*le chemin source-à-destination se comporte à peu près comme le circuit tél.*”

- ♦ performance
  - ♦ actions réseau le long du chemin source-à-destination
- 
- ♦ procédure d'appel avant le *transfert* des données
  - ♦ chaque paquet transporte l'identificateur du CV (pas d'id pour le poste destination)
  - ♦ *chaque* routeur sur le chemin source-dest maintien l'état pour chaque connexion établie
    - ♦ la connexion de la couche transport n'implique que les deux systèmes terminaux
  - ♦ les ressources du routeur (bande passante, buffers) peuvent être *allouées* au CV
    - ♦ s'approcher de la performance du circuit réel

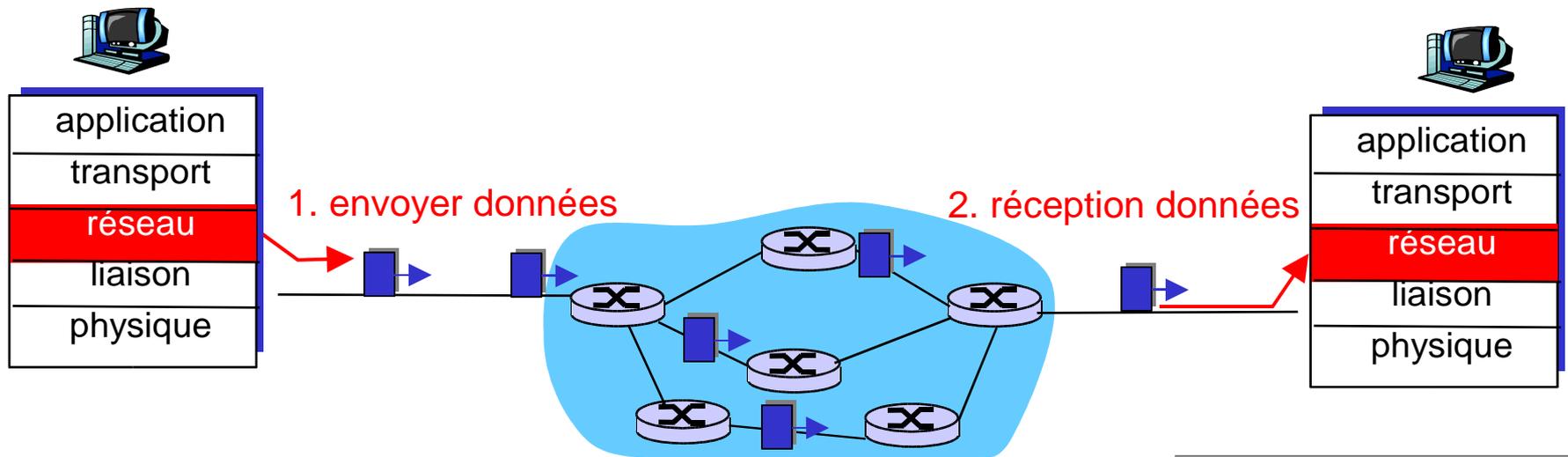
# Circuits virtuels : protocoles de signalisation

- ♦ utilisés pour initialiser, maintenir et détruire les CV
- ♦ utilisés dans ATM, frame-relay, X.25
- ♦ non utilisés dans l'Internet d'aujourd'hui



# Réseaux datagrammes : le modèle Internet

- ♦ pas d'appel au niveau de la couche réseau
- ♦ routeurs : pas d'état à propos les connections bout-à-bout
  - ♦ pas de concept «connexion» au niveau du réseau
- ♦ les paquets acheminés en utilisant l'adresse de destination
  - ♦ les paquets entre la source et la destination peuvent prendre des chemins différents



# Modèles des services de la couche réseau

Architecture réseau	Modèle du service	Garanties ?				Feedback de congestion
		Bande pass	Perte	Ordre	Timing	
Internet	meilleur effort	non	non	non	non	non (inféré via les pertes)
ATM	CBR	taux constant	oui	oui	oui	pas de congestion
ATM	VBR	taux garanti	oui	oui	oui	pas de congestion
ATM	ABR	minimum garanti	non	oui	non	oui
ATM	UBR	non	non	oui	non	non

- ♦ Le modèle Internet s'étend : Intserv, Diffserv

# Réseau datagramme ou CV : pourquoi ?

## Internet

- ♦ échange de données entre ordinateurs
  - ♦ service “élastique”, aucune demande stricte de synchronisation
- ♦ systèmes terminaux «futés» (ordinateurs)
  - ♦ s'adaptent, contrôlent, détectent les erreurs
  - ♦ intérieur du réseau simple, complexité aux «bords»
- ♦ plusieurs types de liens
  - ♦ caractéristiques différentes
  - ♦ service uniforme et difficile

## ATM

- ♦ dérivé de la téléphonie
- ♦ conversation humaine :
  - ♦ timing strict, demande la fiabilité
  - ♦ besoin de service garanti
- ♦ systèmes terminaux «muets»
  - ♦ téléphones
  - ♦ complexité à l'intérieur du réseau

# Chapitre 4 : feuille de route

## 4.1 Introduction

## 4.2 Principes du routage

- routage «link state»
- routage «distance vector»

## 4.3 Routage hiérarchique

## 4.4 IP (Internet Protocol)

## 4.5 Routage dans l'Internet

## 4.6 IPv6

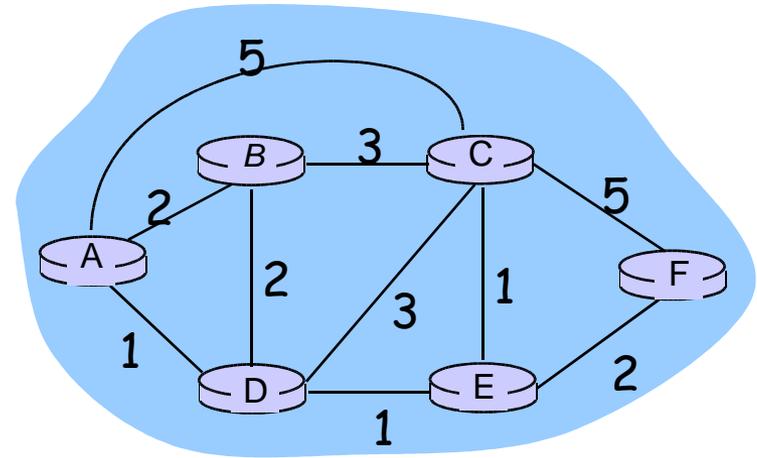
# Routage

## Protocole de routage

**But :** déterminer le “bon” chemin (suite de routeurs) à travers le réseau de la source à la dest.

abstraction à l'aide de graphes :

- ♦ les noeuds sont les routeurs
- ♦ les arcs sont les liens physiques
  - ♦ coût du lien : délai, coût en €, ou le niveau de congestion



- ♦ “bon” chemin :
  - ♦ typiquement : chemin à coût minimum
  - ♦ autres définitions possibles

# Classification des algorithmes de routage

## Information globale ou décentralisée ?

### Globale :

- ♦ tous les routeurs connaissent la topologie complète, info coût des liens
- ♦ algorithmes “link state”

### Décentralisée :

- ♦ un routeur connaît les voisins physiquement connectés, coût des liens voisins
- ♦ processus itératif de calcul, échange des infos avec les voisins
- ♦ algorithmes “distance vector”

## Statique ou dynamique ?

### Statique :

- ♦ les routes changent lentement durant le temps

### Dynamique :

- ♦ les routes changent plus fréquemment
  - ♦ mise-à-jour périodique
  - ♦ en réponse au changement du coût d'un lien

# Un Algorithme de routage «Link-State»

## Algorithme de Dijkstra

- ♦ topologie du réseau, coûts des liens connus dans tous les noeuds
  - ♦ accompli via la “diffusion de l'état du lien” (*link state broadcast*)
  - ♦ tous les noeuds ont la même information
- ♦ calcule le chemin de plus faible coût à partir d'un noeud («source») vers tous les autres
- ♦ donne la **table de routage** pour ce noeud
- ♦ itératif : après k itérations, connaît le plus faible coût vers k dest.

# Un Algorithme de routage «Link-State»

## Notations :

- ♦  $c(i,j)$ : coût du lien du noeud  $i$  au noeud  $j$ .  
(coût infini si non voisins directs)
- ♦  $D(v)$ : valeur courante du coût du chemin de la source à la dest.  $v$
- ♦  $p(v)$ : noeud précédent le long du chemin de la source à  $v$
- ♦  $N$ : ensemble de noeuds pour lesquels le chemin de plus faible coût est connu définitivement

# Algorithme de Dijkstra

1 **Initialisation:**

2  $N = \{A\}$

3 **pour** tous les noeuds  $v$

4 **si**  $v$  adjacent à  $A$

5 **alors**  $D(v) = c(A,v)$

6 **sinon**  $D(v) = \infty$

7

8 **boucle**

9 trouver  $w \notin N$  tel que  $D(w)$  est le minimum

10  $N = N \cup \{w\}$

11 mettre à jour  $D(v)$  pour tout  $v$  adjacent à  $w$  et  $v \notin N$  :

12  $D(v) = \min( D(v), D(w) + c(w,v) )$

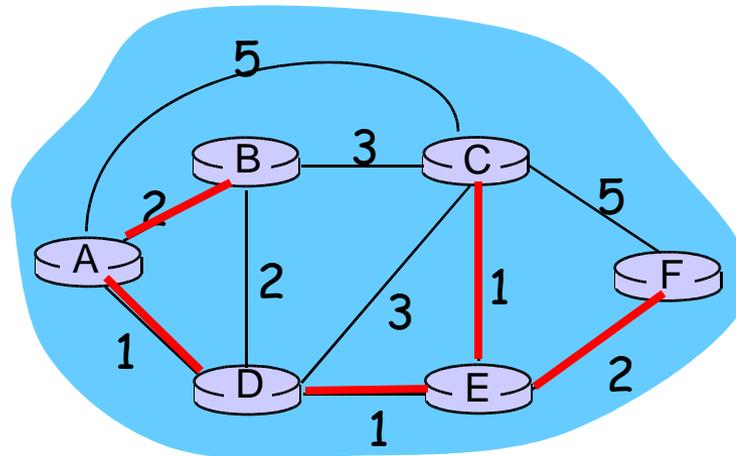
13 /\* le nouveau coût jusqu'à  $v$  est soit l'ancien coût, soit le plus

14 faible coût du chemin jusqu'à  $w$  plus le coût de  $w$  à  $v$  \*/

15 **jusqu'à** ( $N =$  l'ensemble de tous les noeuds)

# Algorithme de Dijkstra : exemple

étape	N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	$\infty$	$\infty$
1	AD	2,A	4,D		2,D	$\infty$
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					



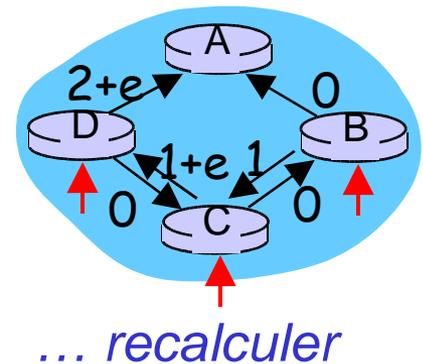
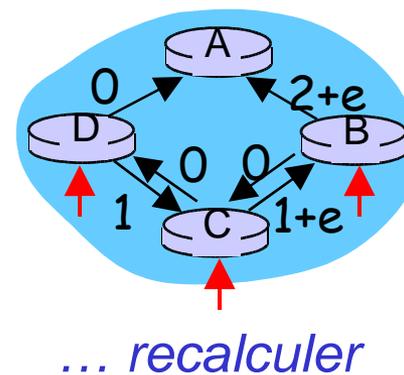
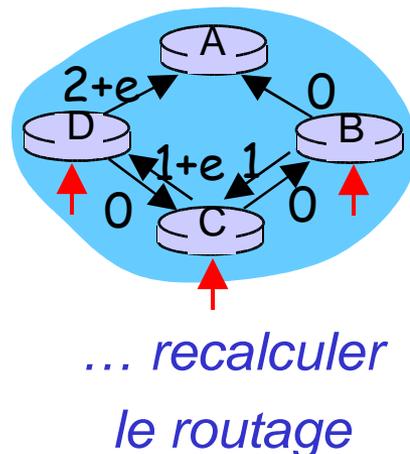
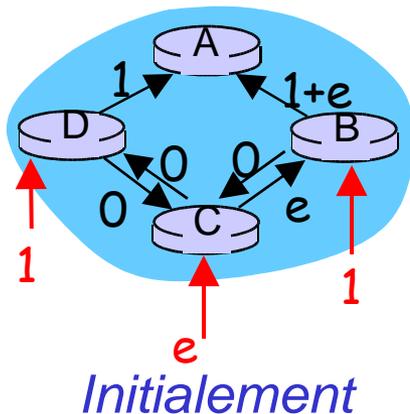
# Algorithme de Dijkstra : discussion

**Complexité** :  $n$  noeuds

- chaque itération : besoin de visiter tous les noeuds  $w \notin N$
- $n*(n+1)/2$  comparaisons :  $O(n^2)$
- implémentations plus efficaces possibles:  $O(n \log n)$

**Oscillations possible:**

- ex. coût du lien = quantité du trafic porté



# Algorithme de routage : *Distance Vector*

## itératif :

- ♦ tourne jusqu'à l'arrêt de l'échange d'infos entre noeuds
- ♦ *termine seul* : pas de “signal” pour stopper

## asynchrone :

- ♦ les noeuds n'échangent pas d'info et n'itèrent pas dans une étape verrouillée!

## distribué :

- ♦ chaque noeud communique *seulement* avec ses voisins directs

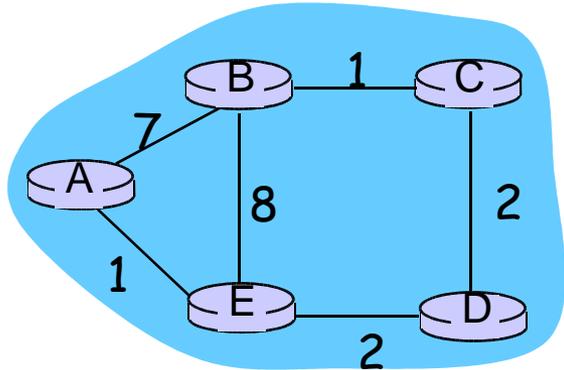
# Algorithme de routage : *Distance Vector*

## Structure de données : Table de distance

- ♦ chaque noeud a sa propre table :
  - ♦ ligne pour chaque destination possible
  - ♦ colonne pour chacun de ses voisins directs
- ♦ exemple : dans le noeud X, pour la dest. Y via le voisin Z :

$$\begin{aligned} D^X(Y,Z) &= \text{distance de X à} \\ &= Y, \text{ via Z comme prochain saut} \\ &= c(X,Z) + \min_w \{D^Z(Y,w)\} \end{aligned}$$

# Table de distance : exemple



$$D^E(C,D) = c(E,D) + \min_w \{D^D(C,w)\}$$

$$= 2+2 = 4$$

$$D^E(A,D) = c(E,D) + \min_w \{D^D(A,w)\}$$

$$= 2+3 = 5 \text{ *boucle!*}$$

$$D^E(A,B) = c(E,B) + \min_w \{D^B(A,w)\}$$

$$= 8+6 = 14 \text{ *boucle!*}$$

coût vers la destination via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destination

# Table de distance → table de routage

coût vers la destination via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destination

lien sortant à utiliser, coût

A	A,1
B	D,5
C	D,4
D	D,2

destination

Table de distance



Table de routage

# Routage *Distance Vector* : vue d'ensemble

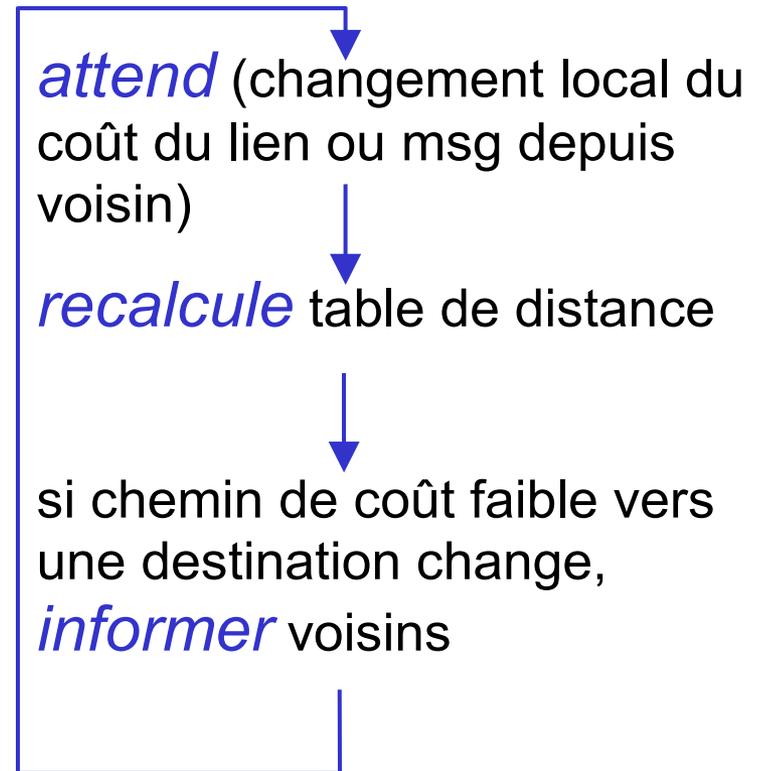
## Itératif, asynchrone :

- chaque itération locale causée par :
  - changement local du coût du lien
  - message de la part d'un voisin : son plus court chemin change depuis le voisin

## Distribué :

- chaque noeud informe ses voisins uniquement quand son propre plus court chemin à une destination change
  - les voisins informent alors leurs voisins si nécessaire

## Chaque noeud :



# Algorithme *Distance Vector* :

*A tous les noeuds X :*

- 1 **Initialisation :**
- 2 ***pour tous*** noeuds adjacents  $v$  :
- 3      $D^X(*,v) = \infty$      /\* l'opérateur \* signifie "pour toutes les lignes" \*/
- 4      $D^X(v,v) = c(X,v)$
- 5 ***pour toutes*** les destinations  $y$
- 6     envoyer  $\min_w D^X(y,w)$  à tout voisin /\*  $w$  : tout voisin de  $X$  \*/

# Algorithme *Distance Vector* : suite

8 **boucle**

9 **attendre** (jusqu'à ce que je vois le changement du coût d'un lien voisin V  
10 ou jusqu'à ce que je reçoive la m-à-j d'un voisin V)

11

12 **si** ( $c(X,V)$  dévie de  $d$ )

13 */\* ajouter d aux coûts de toutes les dest via le voisin v \*/*

14 */\* note: d peut être positif ou négatif \*/*

15 pour toutes les destinations  $y$  :  $D^X(y,V) = D^X(y,V) + d$

16

17 **sinon si** (m-à-j reçue de V par rapport à la destination Y)

18 */\* plus court chemin de V à un Y a changé \*/*

19 */\* V a envoyé une nouvelle valeur de son  $\min_w D^V(Y,w)$  \*/*

20 */\* appelons cette nouvelle valeur reçue "nouv\_val" \*/*

21 pour la seule destination Y :  $D^X(Y,V) = c(X,V) + \text{nouv\_val}$

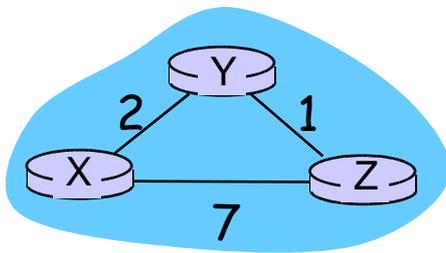
22

23 **si** nous avons une nouvelle val.  $\min_w D^X(Y,w)$  pour une dest. Y

24 envoyer la nouvelle val. de  $\min_w D^X(Y,w)$  à tous les voisins

25 **pour toujours**

# Algorithme *Distance Vector* : exemple



		cost via	
		Y	Z
d e s t	D <sup>X</sup>		
	Y	2	∞
Z	∞	7	

		cost via	
		Y	Z
d e s t	D <sup>X</sup>		
	Y	2	8
Z	3	7	

		cost via	
		Y	Z
d e s t	D <sup>X</sup>		
	Y		
Z			

		cost via	
		X	Z
d e s t	D <sup>Y</sup>		
	X	2	∞
Z	∞	1	

		cost via	
		X	Z
d e s t	D <sup>Y</sup>		
	X	2	8
Z	9	1	

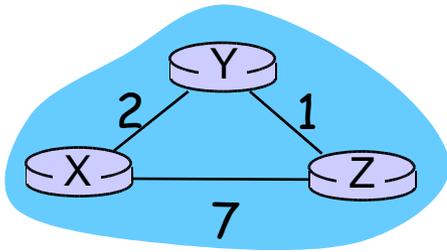
		cost via	
		X	Z
d e s t	D <sup>Y</sup>		
	X		
Z			

		cost via	
		X	Y
d e s t	D <sup>Z</sup>		
	X	7	∞
Y	∞	1	

		cost via	
		X	Y
d e s t	D <sup>Z</sup>		
	X	7	3
Y	9	1	

		cost via	
		X	Y
d e s t	D <sup>Z</sup>		
	X		
Y			

# Algorithme *Distance Vector* : exemple



		cost via	
		Y	Z
d e s t	D <sup>X</sup>		
	Y	2	∞
Z	∞	7	

		cost via	
		X	Z
d e s t	D <sup>Y</sup>		
	X	2	∞
Z	∞	1	

		cost via	
		X	Y
d e s t	D <sup>Z</sup>		
	X	7	∞
Y	∞	1	

		cost via	
		Y	Z
d e s t	D <sup>X</sup>		
	Y	2	8
Z	3	7	

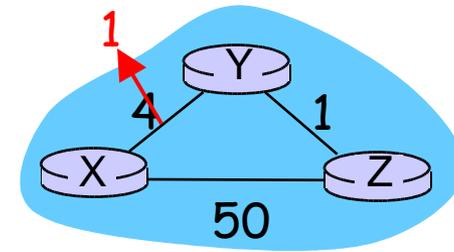
$$D^X(Y,Z) = c(X,Z) + \min_w \{D^Z(Y,w)\} \\ = 7 + 1 = 8$$

$$D^X(Z,Y) = c(X,Y) + \min_w \{D^Y(Z,w)\} \\ = 2 + 1 = 3$$

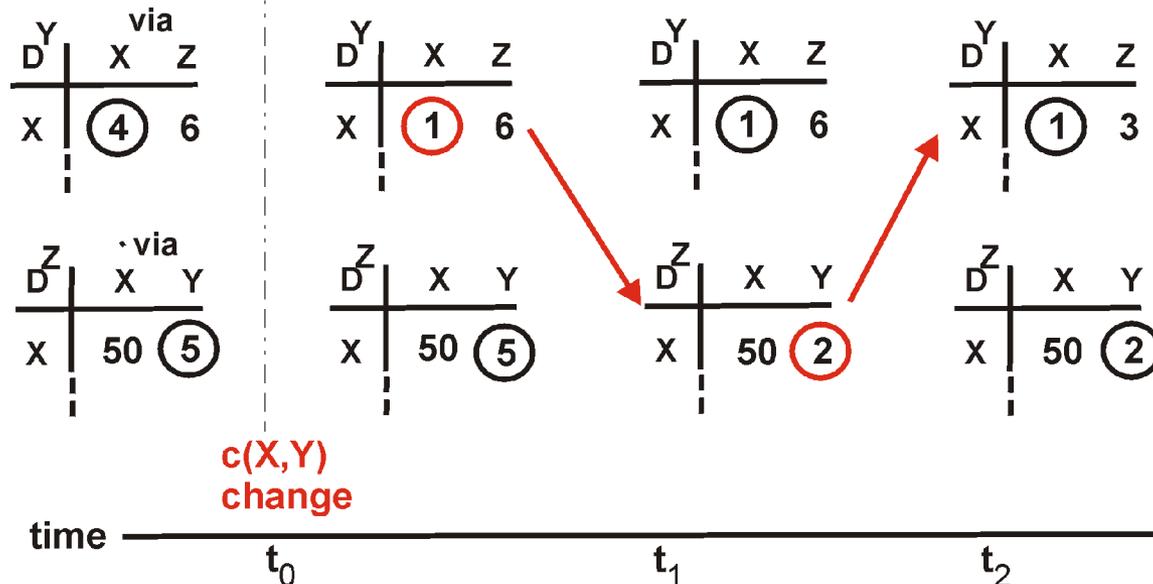
# Distance Vector : changements du coût du lien

## Changements du coût du lien :

- le noeud détecte le changement du coût du lien local
- m-à-j la table de distance (ligne 15)
- si le coût change dans le chemin le plus court, informer les voisins (lignes 23,24)



“les bonnes nouvelles se propagent vite”

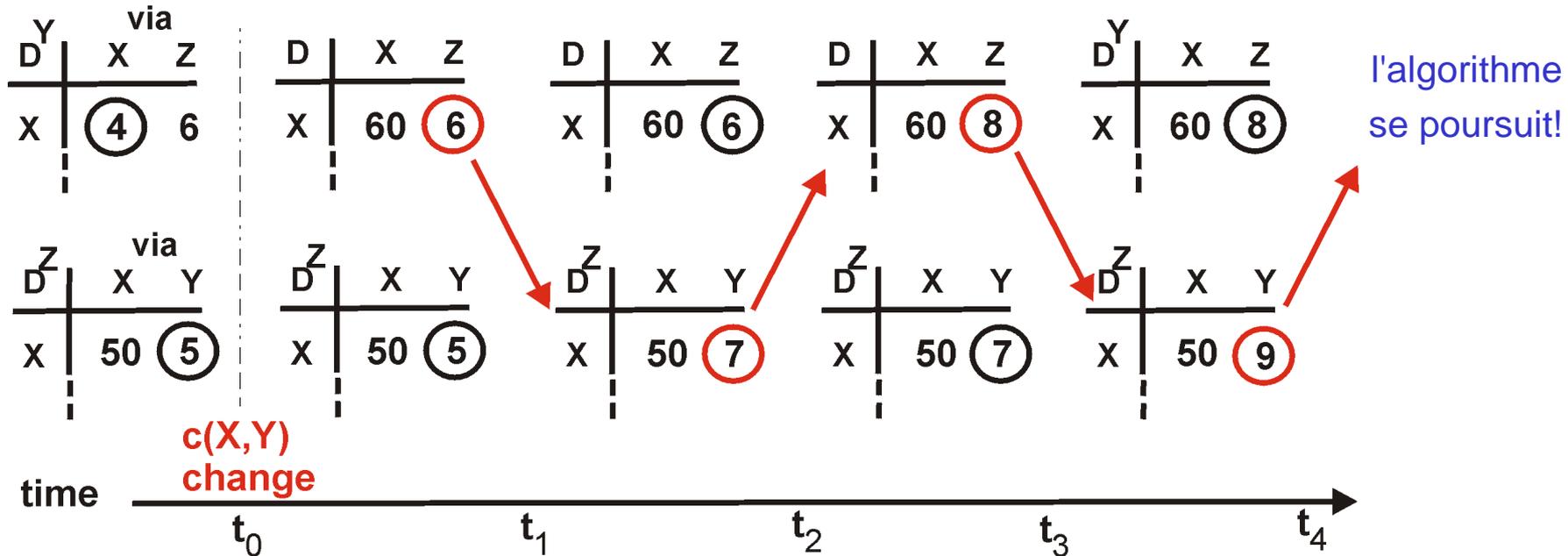
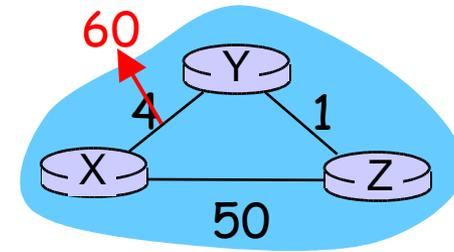


l'algorithme termine

# Distance Vector : changements du coût du lien

## Changements du coût du lien :

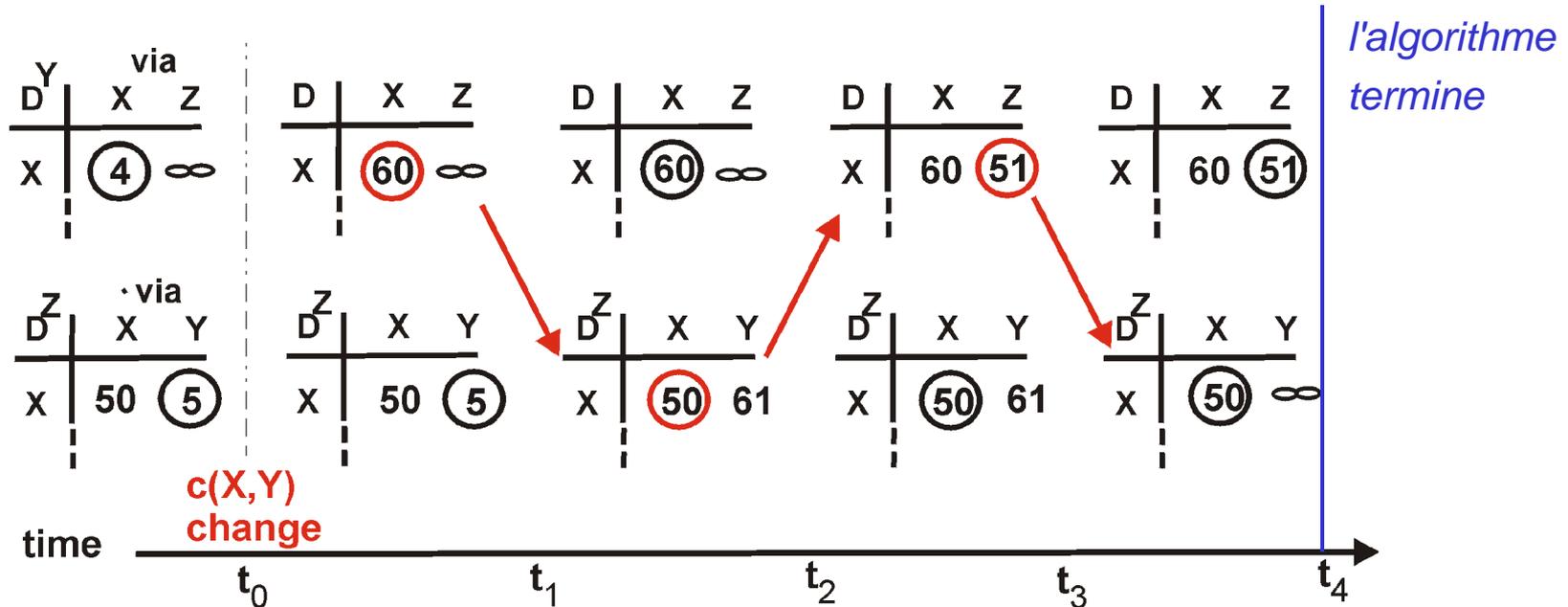
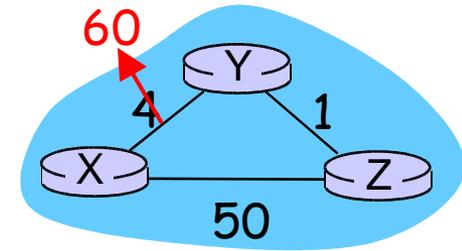
- les bonnes nouvelles se propagent vite
- les mauvaises nouvelles se propagent lentement – problème de “calcul à l’infini” !



# Distance Vector : mensonge au voisin

Si Z route via Y pour atteindre X :

- Z dit à Y que sa distance à X est infinie (donc Y ne peut pas router à X via Z)
- cela résoudra-t-il complètement le problème du calcul infini ?



# Comparaison des algorithmes LS et DV

## Complexité des messages

- LS: avec  $n$  noeuds,  $E$  liens,  $O(nE)$  msgs envoyés
- DV: échange entre voisins uniquement
  - temps de convergence variable

## Vitesse de Convergence

- LS:  $O(n^2)$ 
  - peut donner lieu à des oscillations
- DV: temps de convergence variable
  - le routage peut boucler
  - problème de calcul à l'infini

**Robustesse:** que ce passe-t-il si le routeur fonctionne mal ?

- LS: - un noeud peut publier un coût *incorrect* d'un *lien*
- chaque noeud calcule uniquement sa propre table

- DV: - un noeud peut publier un coût *incorrect* d'un *chemin*
- chaque table d'un noeud est utilisée par les autres
    - les erreurs se propagent à travers le réseau

# Chapitre 4 : feuille de route

4.1 Introduction

4.2 Principes du routage

4.3 Routage hiérarchique

4.4 IP (Internet Protocol)

4.5 Routage dans l'Internet

4.6 IPv6

4.7 Routage multicast

4.8 Mobilité

# Routage hiérarchique

Notre étude du routage est idéaliste

- ♦ tous les routeurs identiques
- ♦ réseau “plat”

... *pas vrai* dans la pratique

**échelle** : avec 200 millions de

destinations :

- ♦ ne peut pas stocker toutes les dest dans les tables de routage !
- ♦ l'échange des tables de routage risque d'inonder les liens !

**autonomie administrative**

- ♦ internet = réseau des réseaux
- ♦ chaque admin d'un réseau veut contrôler le routage dans son propre réseau

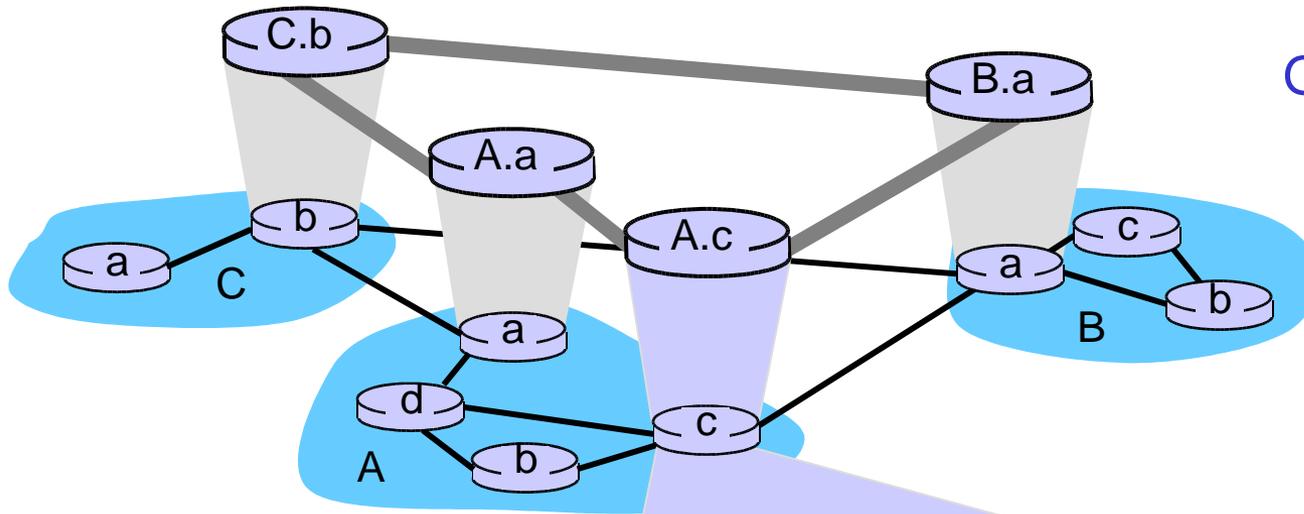
# Routage hiérarchique

- ♦ agréger des routeurs au sein de régions : “**systèmes autonomes**” (AS : **autonomous systems**)
- ♦ les routeurs dans le même AS exécutent le même protocole de routage
  - ♦ protocole **de routage “intra-AS”**
  - ♦ les routeurs d'AS différents peuvent exécuter des protocoles de routage intra-AS différents

## *routeurs «gateways»*

- ♦ des routeurs spéciaux dans les AS
- ♦ exécutent le protocole de routage intra-AS avec les autres routeurs de l'AS
- ♦ responsables aussi du routage vers les destinations hors l'AS
  - ♦ exécutent un protocole de routage **inter-AS** avec les autres routeurs «portes»

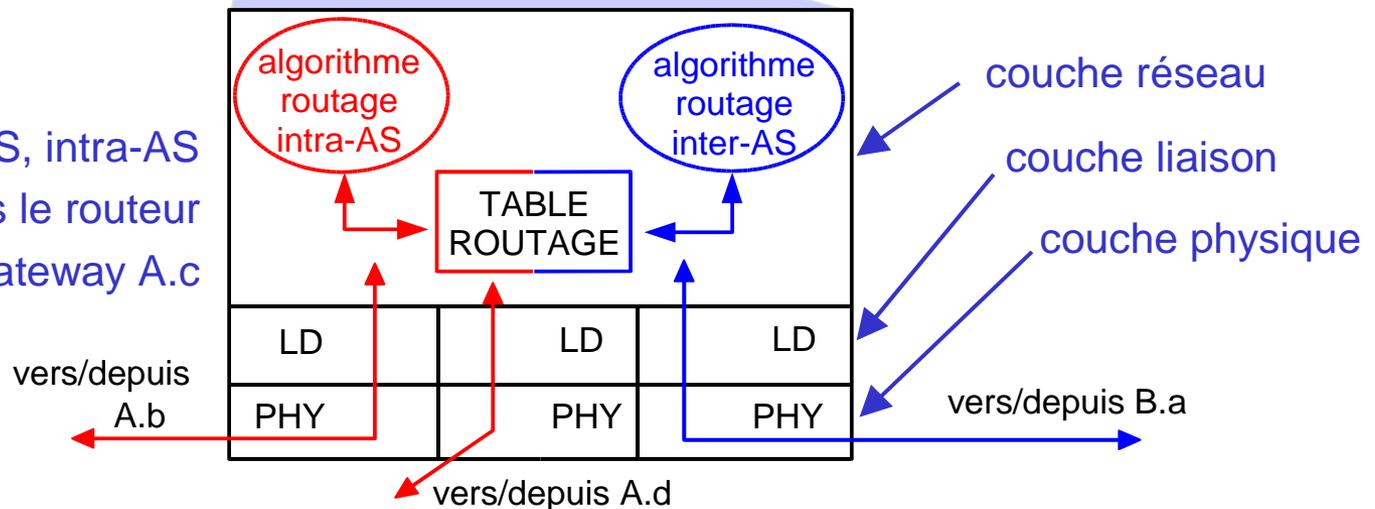
# Routage Intra-AS et Inter-AS



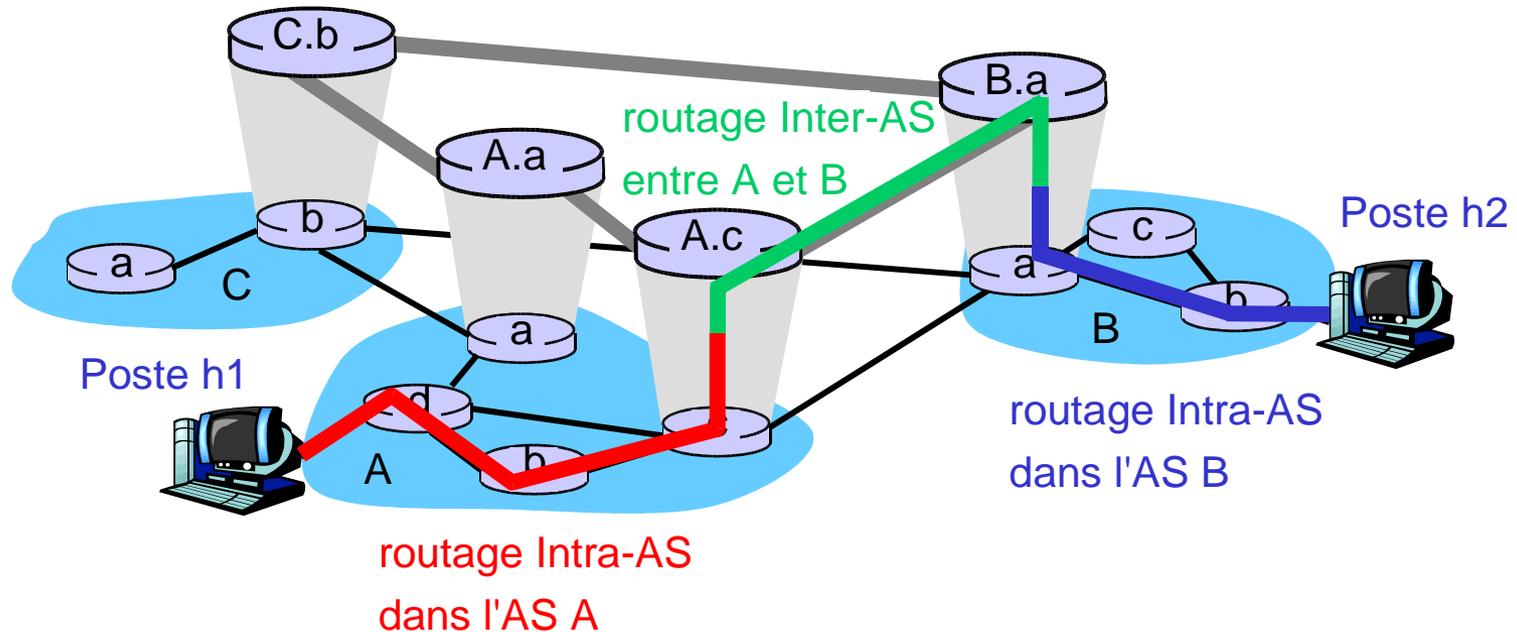
## Gateways :

- routage inter-AS entre eux
- routage intra-AS entre routeurs du même AS

routage inter-AS, intra-AS dans le routeur gateway A.c



# Routage Intra-AS et Inter-AS



- Nous examinerons de façon spécifique les protocoles de routage inter-AS et intra-AS de l'Internet

# Chapitre 4 : feuille de route

4.1 Introduction      4.2 Principes du routage      4.3 Routage hiérarchique

## 4.4 IP (Internet Protocol)

- Adressage IPv4
- Acheminement des datagrammes de la source à la dest.
- Format des datagrammes
- Fragmentation IP
- ICMP: Internet Control Message Protocol
- DHCP: Dynamic Host Configuration Protocol
- NAT: Network Address Translation

4.5 Routage dans l'Internet

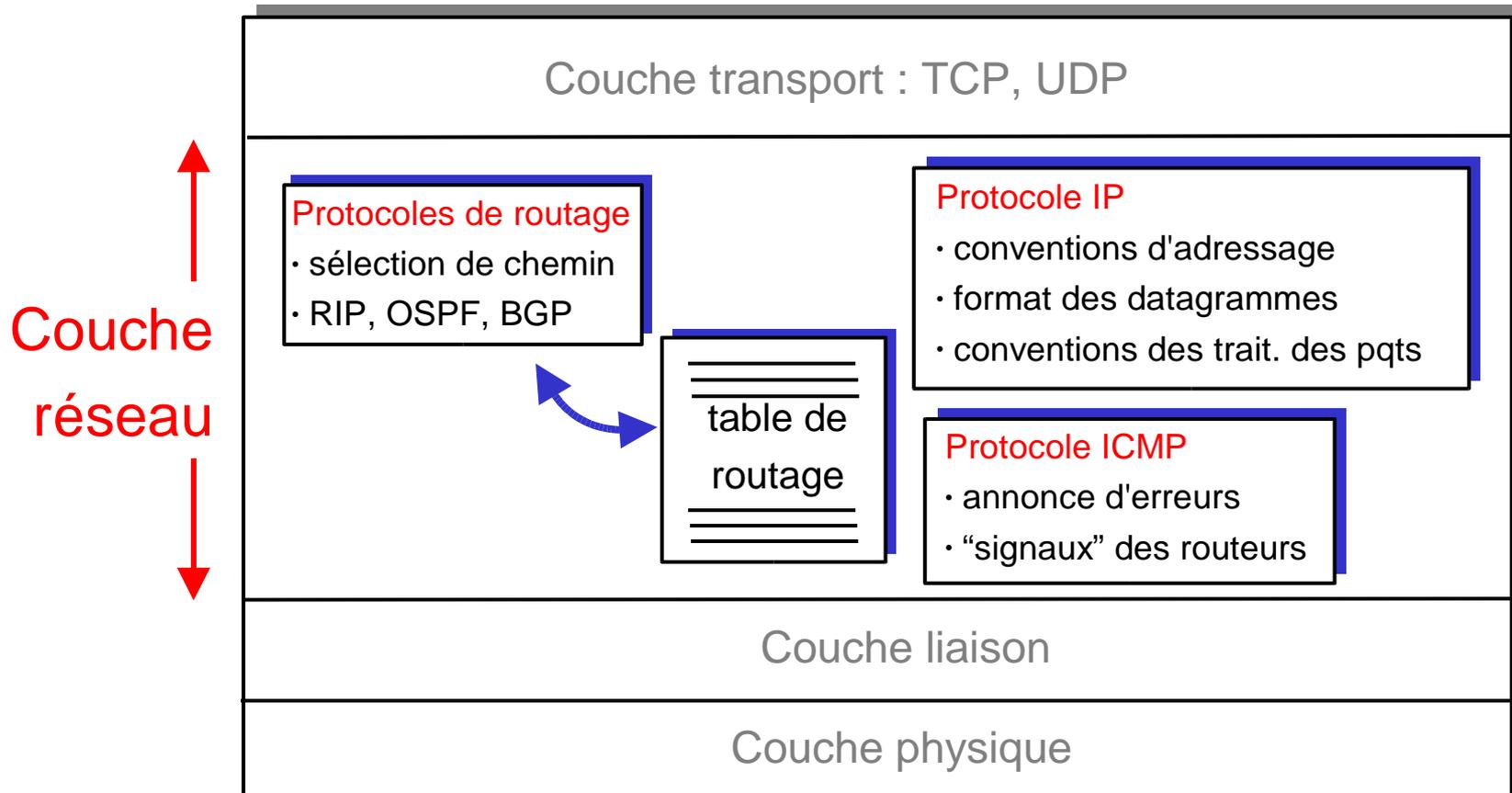
4.6 IPv6

4.7 Routage multicast

4.8 Mobilité

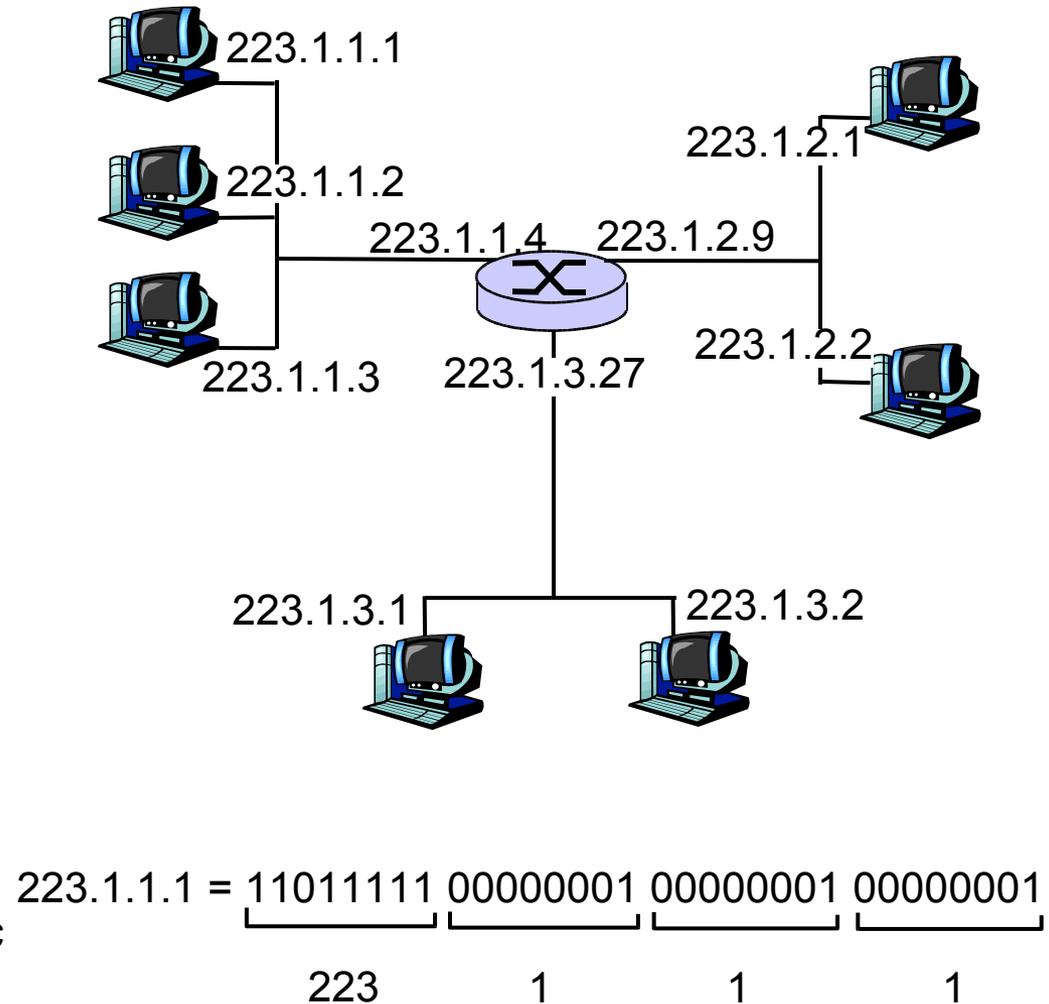
# La couche réseau de l'Internet

Fonctions de la couche réseau des postes et routeurs :



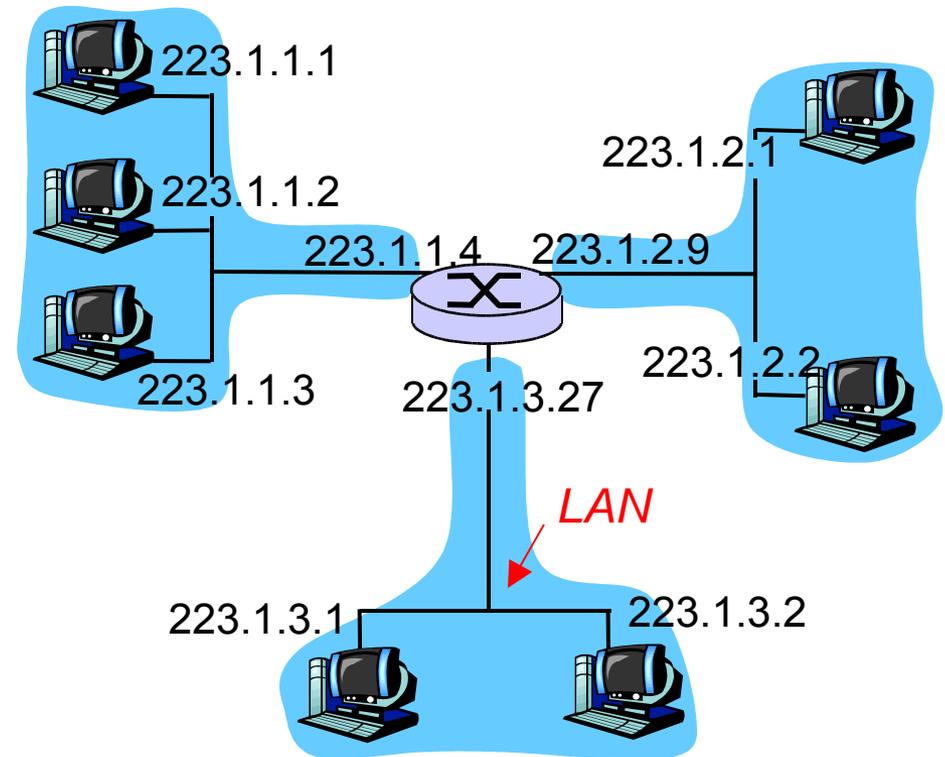
# Adressage IP : introduction

- **adresse IP** : identificateur 32-bits pour un poste, *interface* de routeur
- **interface** : connexion entre poste/routeur et un lien physique
  - typiquement, tous les routeurs ont plusieurs interfaces
  - les postes peuvent aussi avoir plusieurs interfaces
  - adresses IP associés avec chaque interface



# Adressage IP

- ◆ Adresse IP :
  - ◆ partie réseau (bits d'ordre supérieur)
  - ◆ partie poste (bits d'ordre inférieur)
- ◆ *Qu'est ce qu'un réseau ?* (du point de vue des adresses IP)
  - ◆ interfaces avec la même partie réseau de l'adresse IP
  - ◆ peuvent atteindre physiquement les autres interfaces sans passer par un routeur



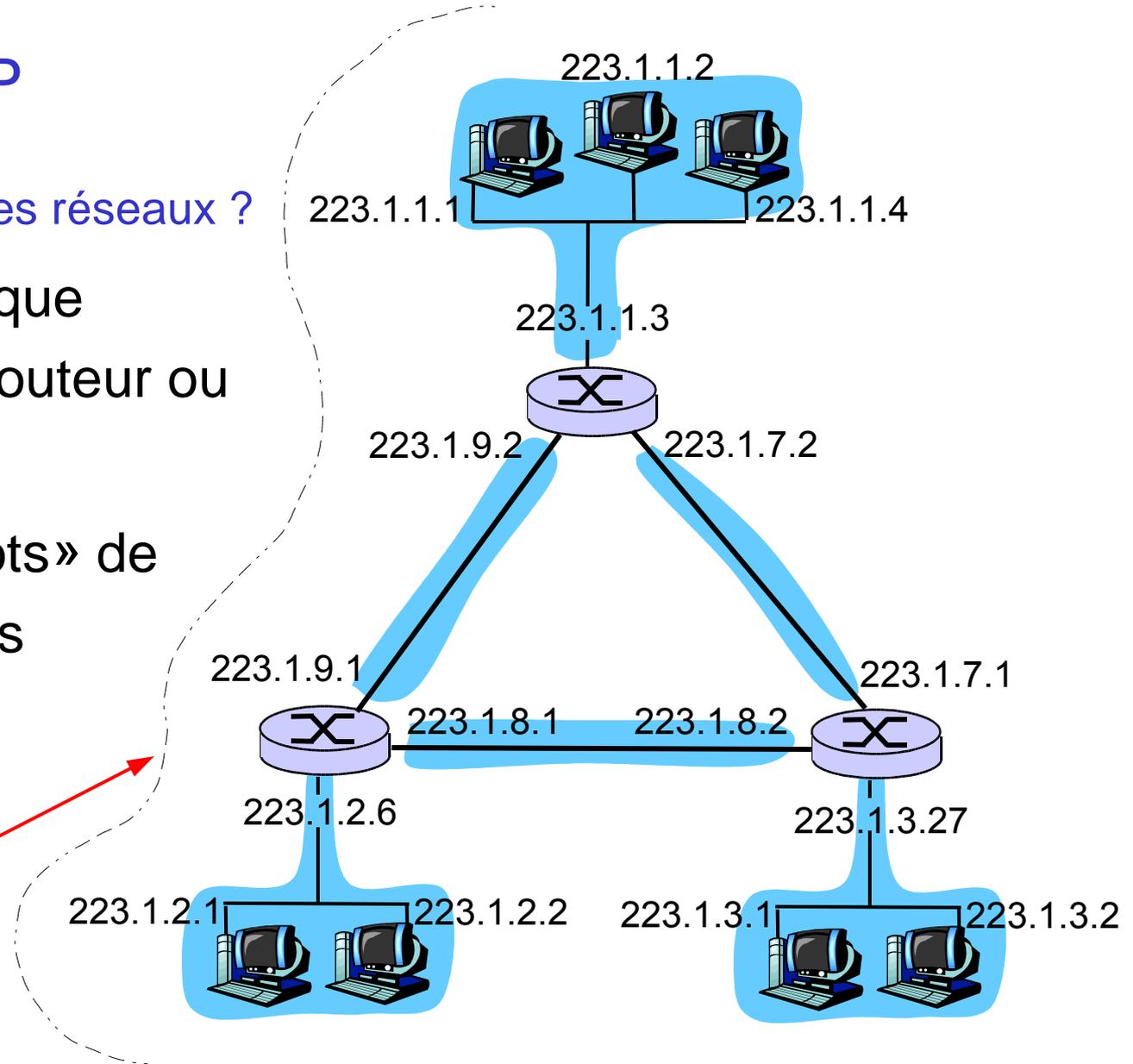
Réseau de 3 réseaux IP  
(pour les adresses IP commençant par 223,  
les 24 premiers bits représentent l'adresse réseau)

# Adressage IP

Comment trouver les réseaux ?

- ♦ détacher chaque interface de routeur ou poste
- ♦ créer des «îlots» de réseaux isolés

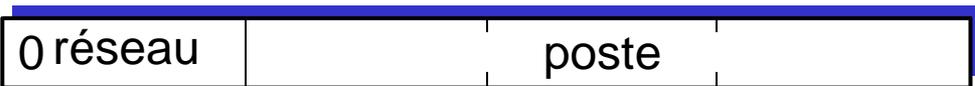
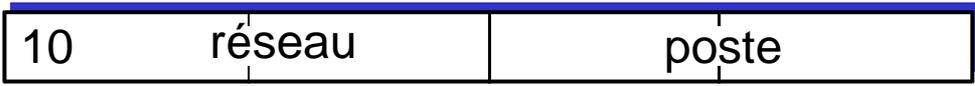
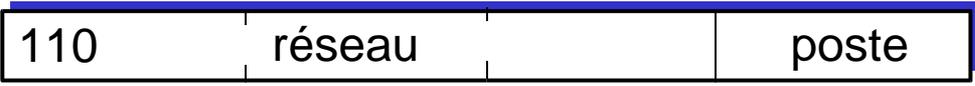
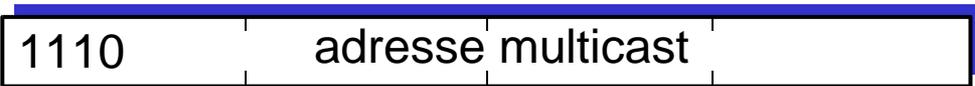
Interconnexion de six réseaux



# Adresses IP

étant donné la notion de «réseau», ré-examinons les adresses IP :

“classes” d'adressage :

classe	adresse	intervalle d'adresses
A		1.0.0.0 → 127.255.255.255
B		128.0.0.0 → 191.255.255.255
C		192.0.0.0 → 223.255.255.255
D		224.0.0.0 → 239.255.255.255

 32 bits

# Adresses IP : adresses spéciales

**Format d'adresse :** bits partie réseau | bits partie poste

- ✓ Adresse du **réseau** : *partie réseau*      00.....0 (*tous les bits à 0*)
- ✓ Adresse de **diffusion** : *partie réseau*      11.....1 (*tous les bits à 1*)  
(adresse de *broadcast*)
- ✓ **1ère adr.** de poste : *partie réseau*      00.....01
- ✓ **dernière adr.** de poste: *partie réseau*      11.....10

## Exemple :

adresse réseau : 192.168.42.0  
adresse de diffusion : 192.168.42.255  
1ère adr. de poste : 192.168.42.1  
dernière adr. de poste : 192.168.42.254

# Adressage IP : CIDR

- ♦ Adressage par classes :
  - ♦ utilisation inefficace de l'espace d'adressage, épuisement de l'espace d'adressage
  - ♦ ex. : réseau de classe B alloue suffisamment d'adresses pour 65k postes, même s'il y a seulement 2k postes dans le réseau
- ♦ **CIDR: Classless InterDomain Routing**
  - ♦ longueur arbitraire de la partie réseau de l'adresse
  - ♦ format d'adresse : **a.b.c.d/x**, où x est le nombre de bits dans la portion réseau de l'adresse



136.23.16.0/23

(ou 136.23.16.0 masque réseau = 255.255.254.0)

# Adresses IP : comment en avoir une ?

Q: Comment un poste reçoit une adresse IP ?

- ♦ Codée en “dur” par l'administrateur dans un fichier
  - ♦ Win : control-panel→network→configuration→tcp/ip→properties
  - ♦ UNIX : /etc/rc.config
  - ♦ LINUX : /etc/sysconfig/network-scripts/\*
- ♦ **DHCP**: **D**ynamic **H**ost **C**onfiguration **P**rotocol : obtenir dynamiquement une adresse à partir d'un serveur
  - ♦ “plug-and-play” (plus de détail dans la suite)

# Adresses IP : comment en avoir une ?

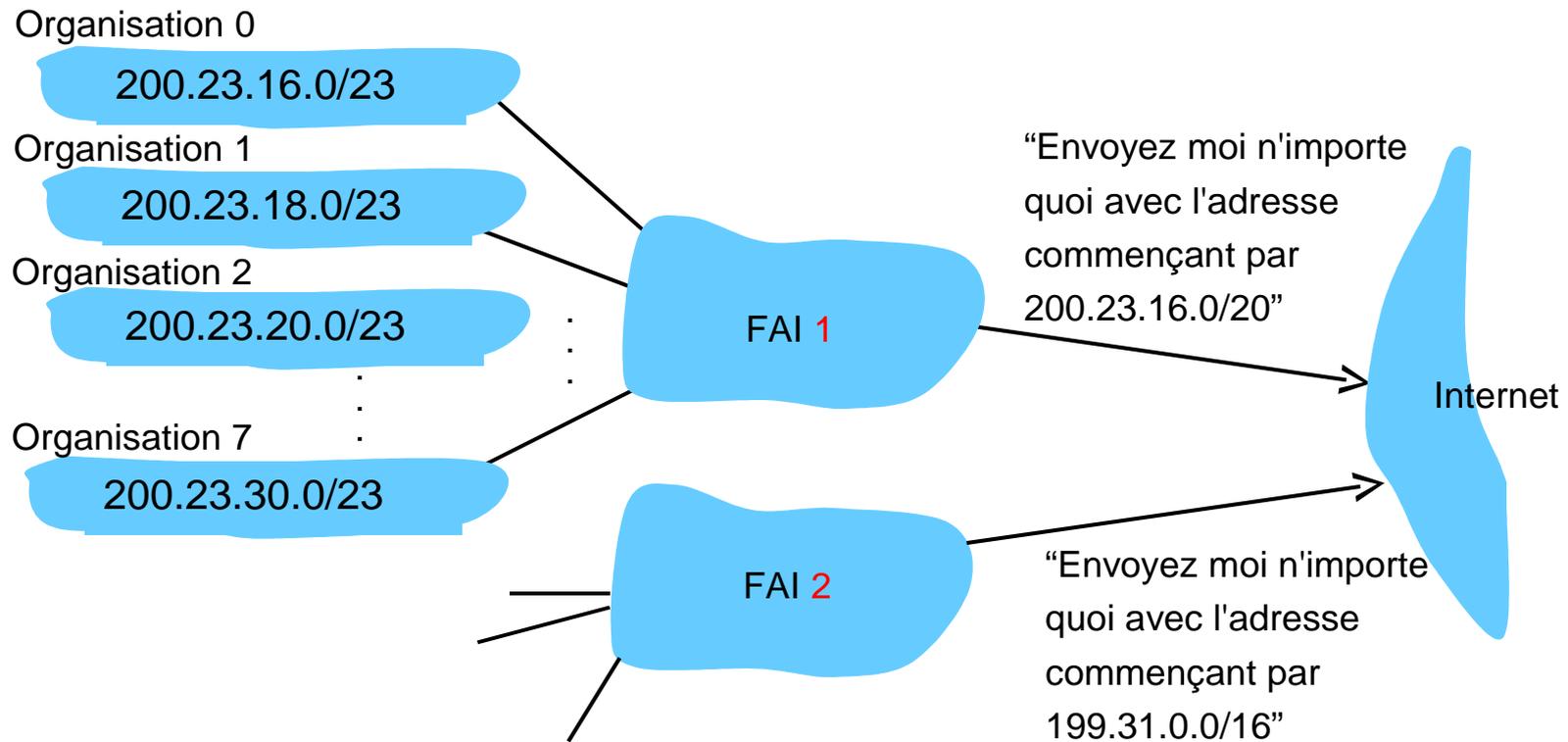
**Q:** Comment le *réseau* reçoit la partie réseau de l'adresse IP ?

**R:** reçoit la portion allouée par l'espace d'adressage du FAI

Bloc du FAI	<u>11001000</u>	<u>00010111</u>	<u>00010000</u>	00000000	200.23.16.0/20
Organisation 0	<u>11001000</u>	<u>00010111</u>	<u>00010000</u>	00000000	200.23.16.0/23
Organisation 1	<u>11001000</u>	<u>00010111</u>	<u>00010010</u>	00000000	200.23.18.0/23
Organisation 2	<u>11001000</u>	<u>00010111</u>	<u>00010100</u>	00000000	200.23.20.0/23
...	.....		....	....	
Organisation 7	<u>11001000</u>	<u>00010111</u>	<u>00011110</u>	00000000	200.23.30.0/23

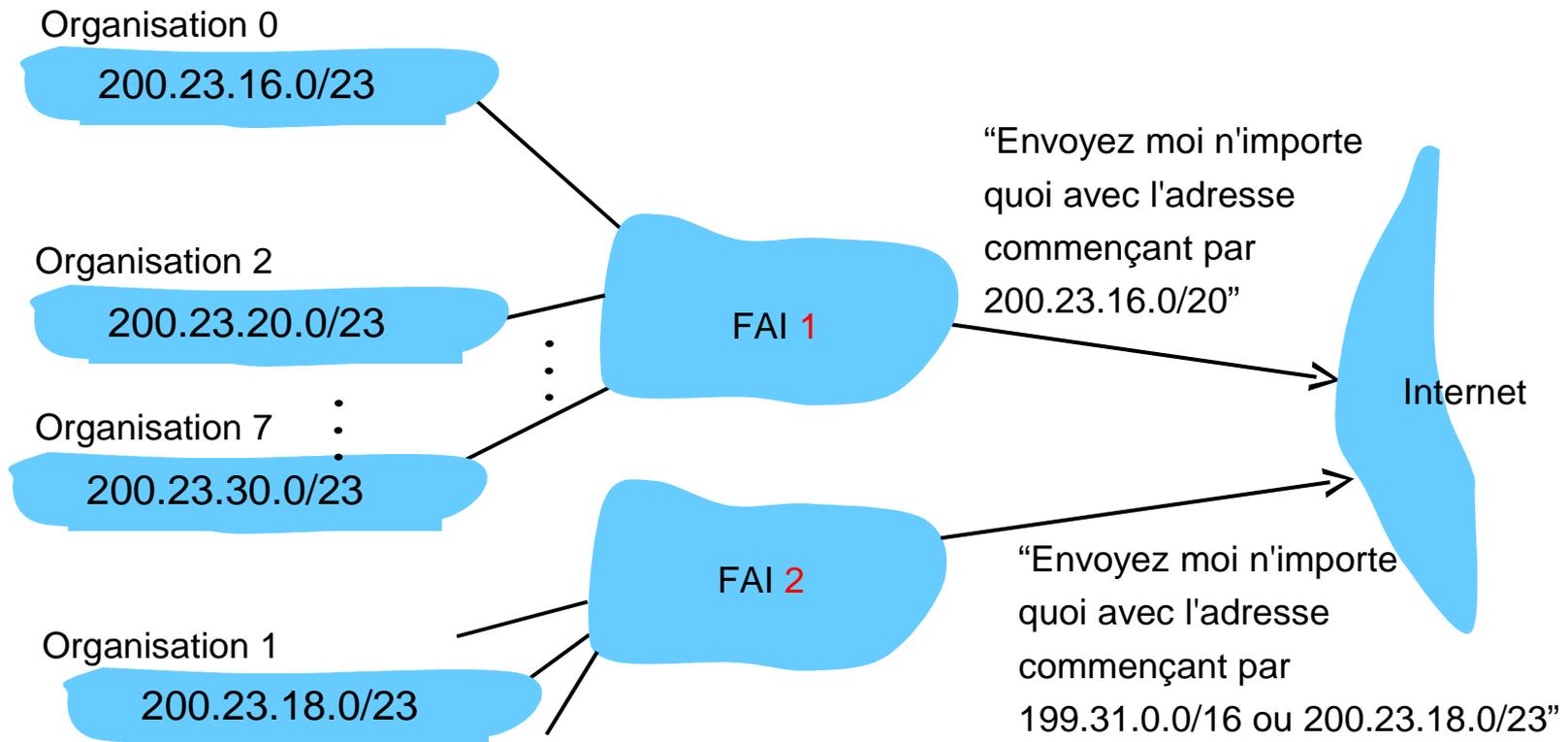
# Adressage hiérarchique : agrégation de routes

L'adressage hiérarchique permet la publication efficace de l'information de routage :



# Adressage hiérarchique : routes plus spécifiques

Le FAI 2 a une route plus spécifique pour *Organisation 1*



# Adressage IP : un dernier mot...

Q: Comment un FAI obtient-il un bloc d'adresses ?

R: **ICANN**: Internet **C**orporation for **A**ssigned  
**N**ames and **N**umbers

- ♦ alloue des adresses
- ♦ gère les DNS
- ♦ attribue les noms de domaines, résout les conflits

# Acheminer un datagramme : de la source à la dest.

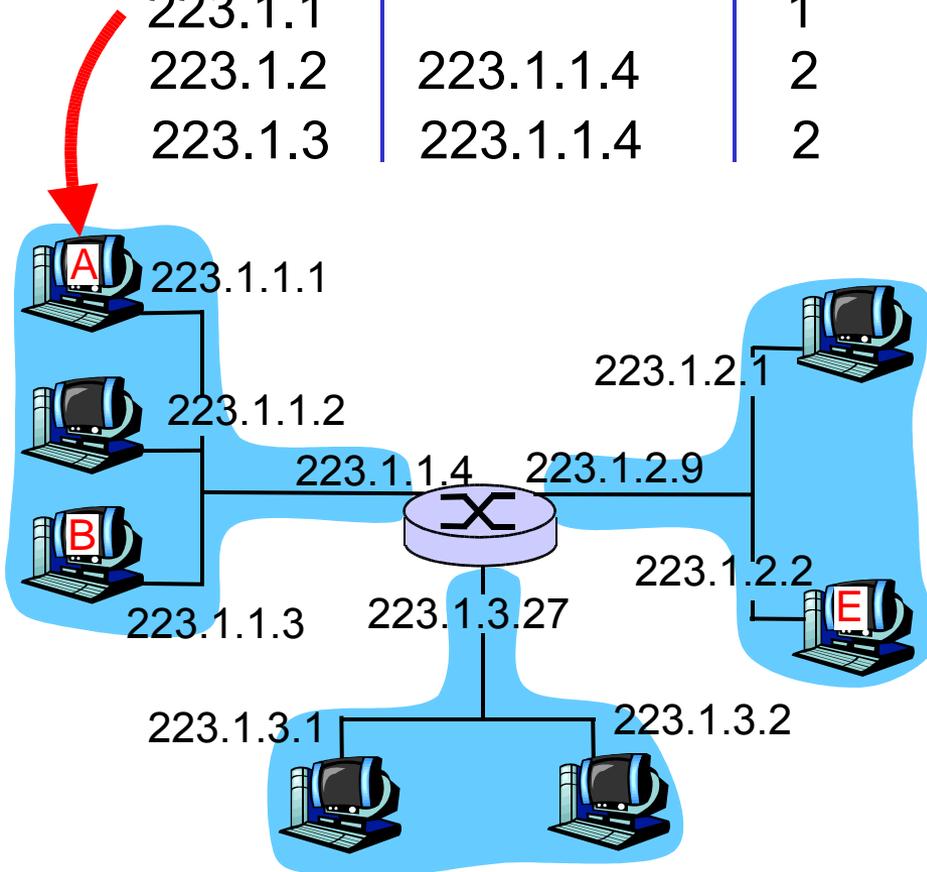
datagramme IP:

champs	adr IP	adr IP	données
divers	source	dest	

- le datagramme reste **inchangé** durant le voyage de la source vers la destination
- utilisation des champs d'adresse

*Table de routage dans A*

rés. dest.	prochain routeur	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



# Acheminer un datagramme : de la source à la dest.

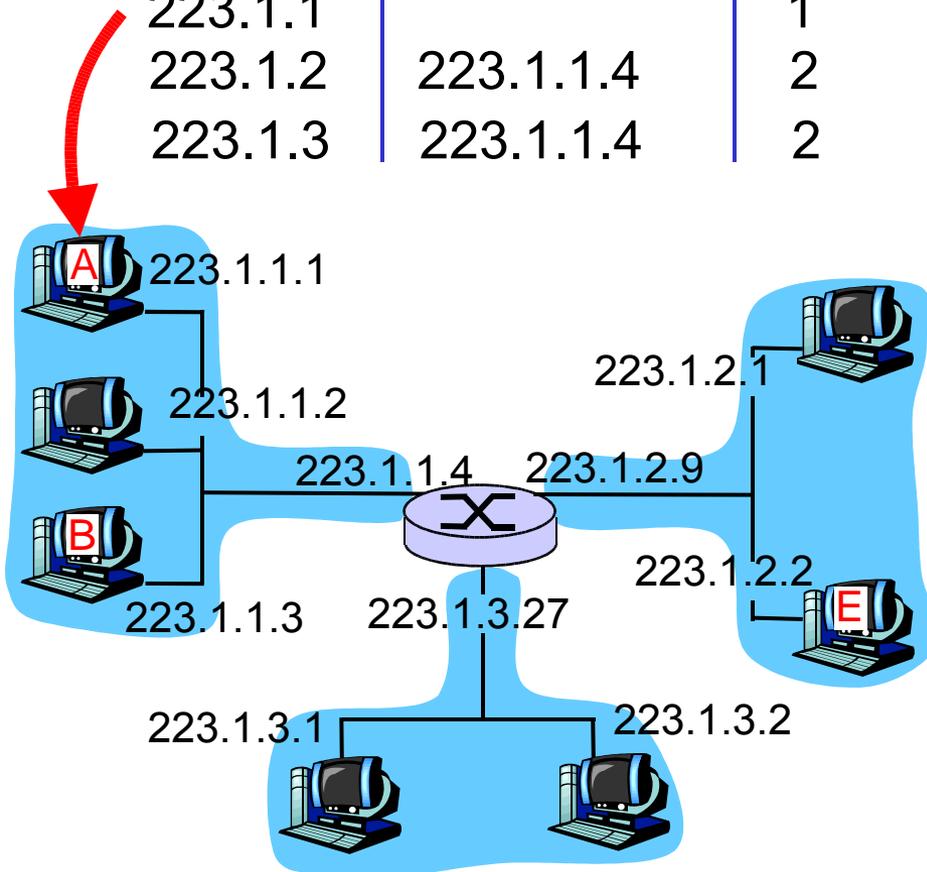
champs divers	223.1.1.1	223.1.1.3	données
---------------	-----------	-----------	---------

de A, envoyer un datagramme IP vers B :

- chercher l'adresse réseau de B dans la table de routage
- B trouvé dans le même réseau que A
- la couche liaison envoie le datagramme (dans une trame liaison) directement à B
  - B et A sont directement connectés

*Table de routage dans A*

rés. dest.	prochain routeur	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



# Acheminer un datagramme : de la source à la dest.

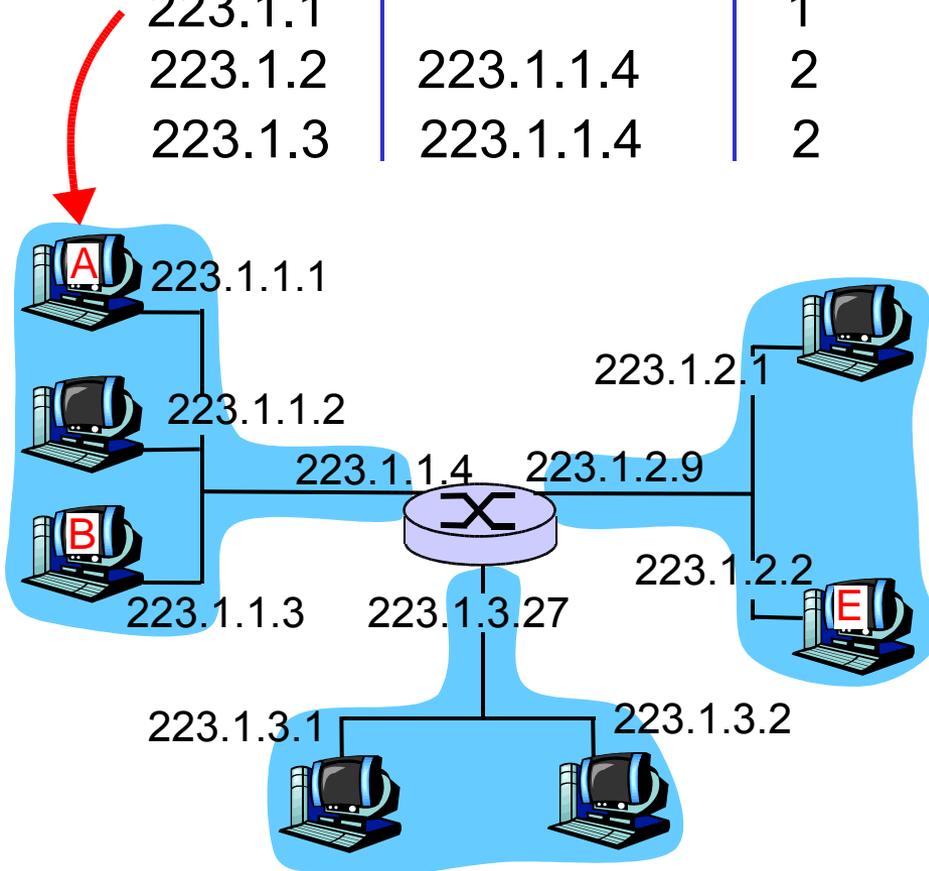
champs	223.1.1.1	223.1.2.2	données
divers			

Source A, dest. E :

- chercher l'adresse réseau de E dans la table de routage de A
- E sur un réseau différent
  - A et E indirectement attachés
- table de routage : le prochain routeur vers E est 223.1.1.4
- la couche liaison envoie le datagramme au routeur 223.1.1.4 (dans une trame liaison)
- le datagramme arrive à 223.1.1.4
- à suivre...

*Table de routage dans A*

rés. dest.	prochain routeur	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



# Acheminer un datagramme : de la source à la dest.

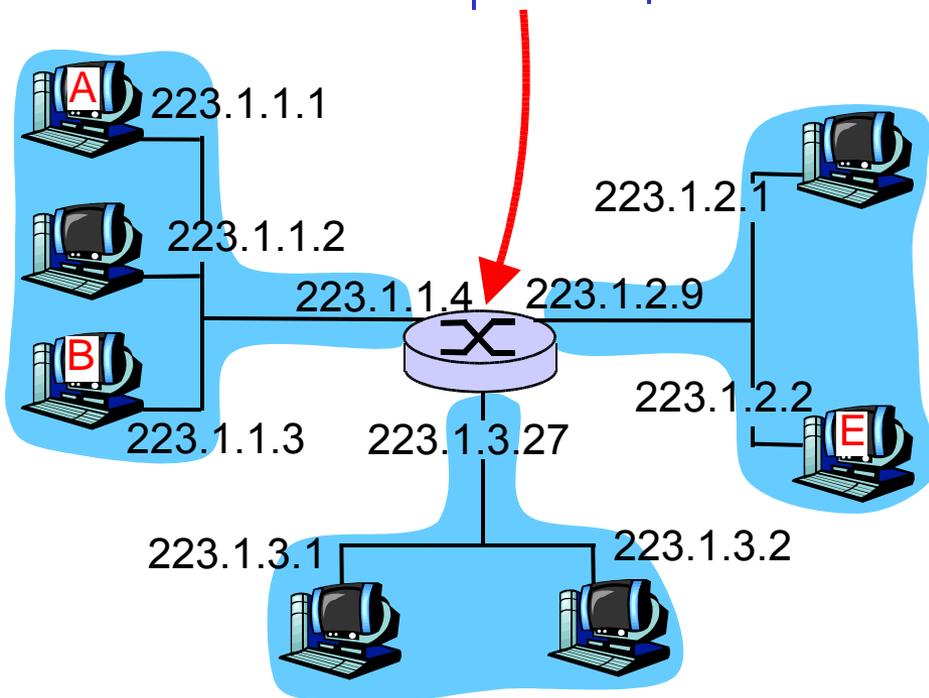
champs	223.1.1.1	223.1.2.2	données
divers			

Arrivée à 223.1.1.4, dest. à 223.1.2.2

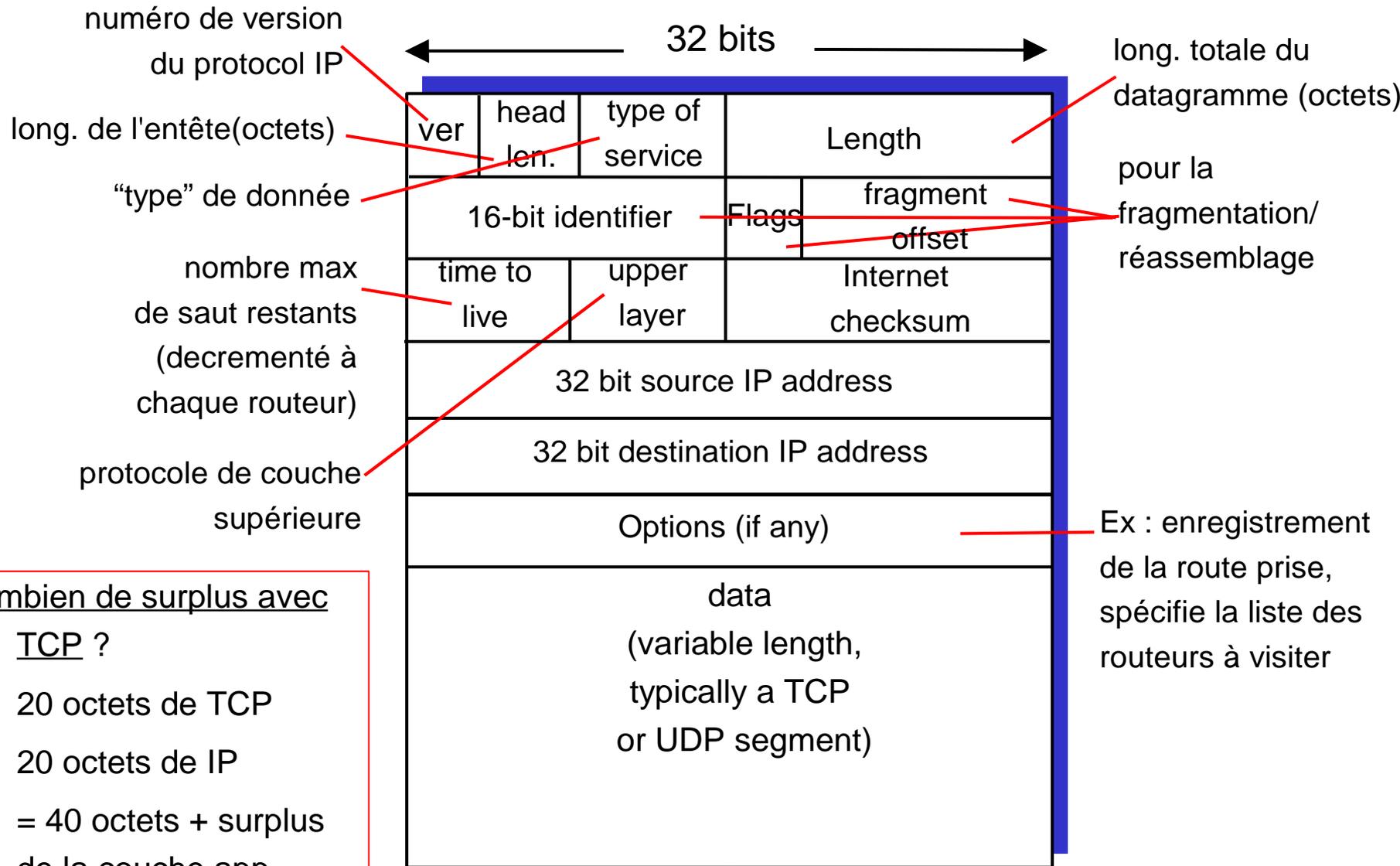
- chercher l'adresse réseau de E dans la table de routage du routeur
- E est sur le même réseau que l'interface 223.1.2.9 du routeur
  - le routeur et E directement attachés
- la couche liaison envoie le datagramme au 223.1.2.2 (dans une trame liaison) via l'interface 223.1.2.9
- le datagramme arrive au 223.1.2.2!!! (ouf!)

Table de routage du routeur

rés. dest.	routeur	Nhops	interface
223.1.1	-	1	223.1.1.4
223.1.2	-	1	223.1.2.9
223.1.3	-	1	223.1.3.27



# Format du datagramme IP



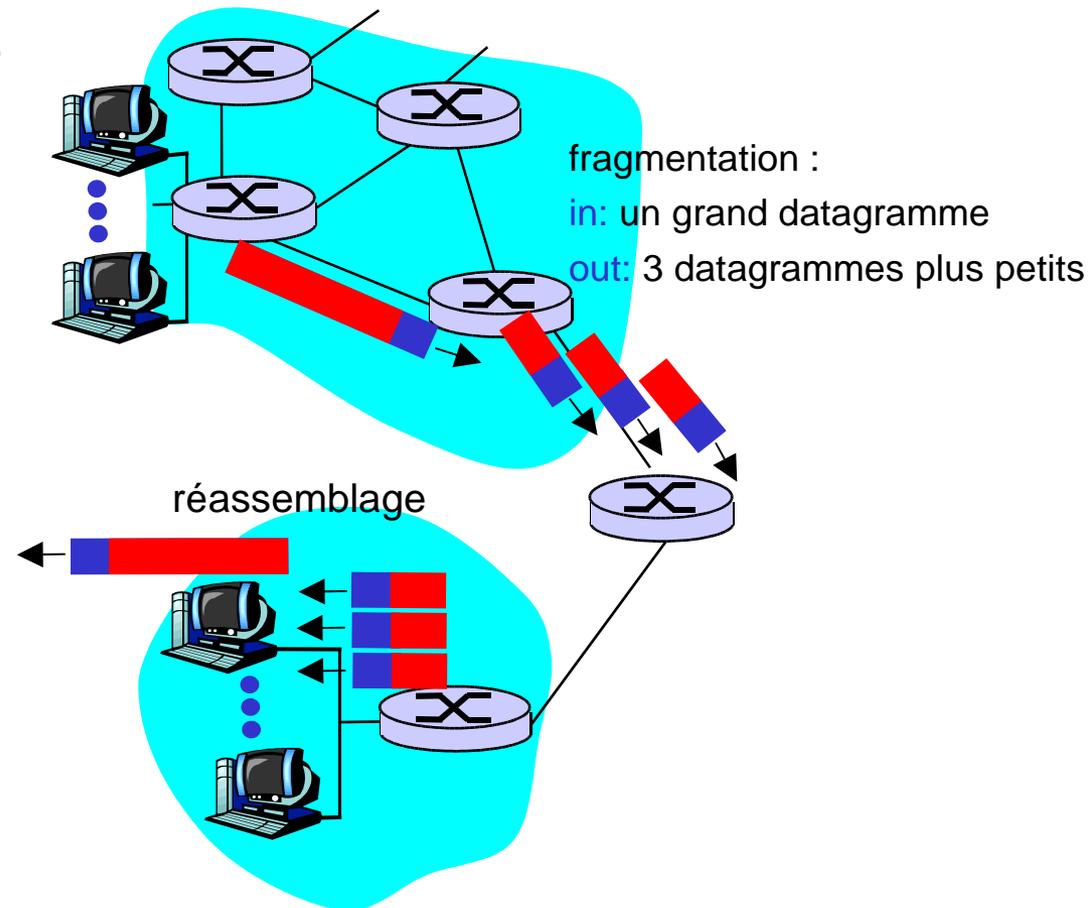
## combien de surplus avec

### TCP ?

- ♦ 20 octets de TCP
- ♦ 20 octets de IP
- ♦ = 40 octets + surplus de la couche app

# Fragmentation et Ré-assemblage IP

- ♦ les liens réseau ont un MTU (taille max. de transfert) : la plus grande trame possible.
  - ♦ types de liens différents, MTUs différents
- ♦ les grands datagrammes IP divisés (“fragmentés”) dans le réseau
  - ♦ un datagramme donne lieu à plusieurs datagrammes
  - ♦ “réassemblés” uniquement au niveau de la dest. finale
  - ♦ bits de l'entête IP utilisés pour identifier, ordonner les fragments



# Fragmentation et Réassemblage IP

## Exemple

- ♦ datagramme de 4000 octets
- ♦ MTU = 1500 octets

	long. =4000	ID =x	flag frag. =0	offset =0	
--	----------------	----------	------------------	--------------	--

Un grand datagramme divisé en plusieurs petits datagrammes

	long. =1500	ID =x	flag frag. =1	offset =0	
--	----------------	----------	------------------	--------------	--

	long. =1500	ID =x	flag frag. =1	offset =1480	
--	----------------	----------	------------------	-----------------	--

	long. =1040	ID =x	flag frag. =0	offset =2960	
--	----------------	----------	------------------	-----------------	--

# ICMP : Internet Control Message Protocol

- ♦ utilisé par les postes, routeurs, gateways pour communiquer les informations du niveau réseau
  - ♦ signaler erreurs : poste, réseau, port, protocole inaccessibles
  - ♦ répercuter demande/réponse (utilisé par **ping**)
- ♦ protocole de niveau réseau au « dessus » de IP:
  - ♦ messages ICMP portés par les datagrammes IP
- ♦ **message ICMP** : type, code, plus premiers 8 octets du datagramme IP responsable de l'erreur

<u>Type</u>	<u>Code</u>	<u>description</u>
0	0	réponse ( <b>ping</b> )
3	0	réseau dest. inaccessible
3	1	poste dest. inaccessible
3	2	protocole dest. inaccessible
3	3	port dest. Inaccessible
3	6	réseau dest. inconnu
3	7	poste dest. inconnu
4	0	source étouffé (contrôle de congestion – non utilisé)
8	0	demande ( <b>ping</b> )
9	0	publication de route
10	0	découverte de routeur
11	0	expiration du TTL
12	0	mauvaise entête IP

# DHCP : Dynamic Host Configuration Protocol

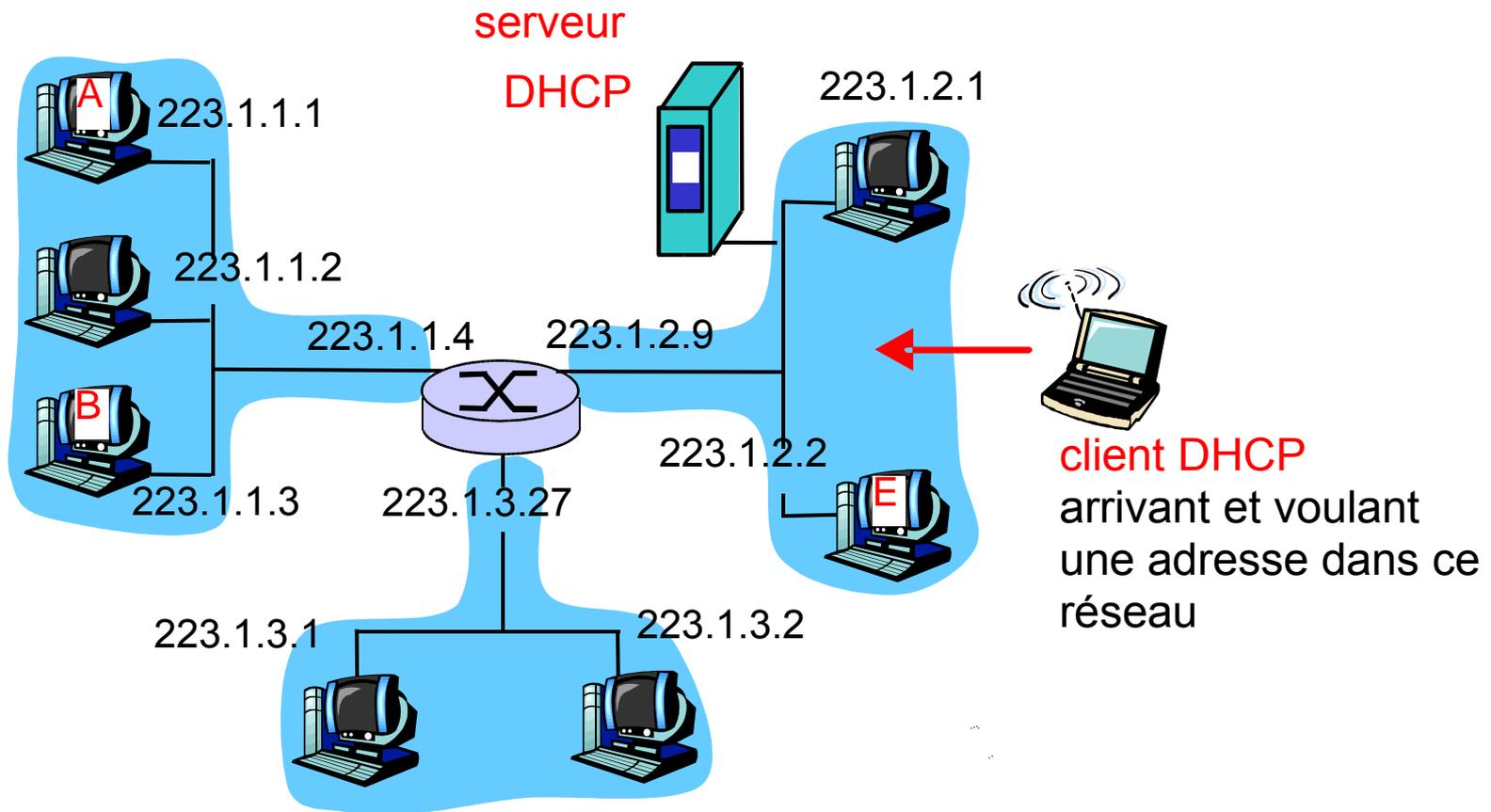
**But** : permet à un poste d'obtenir *dynamiquement* son adresse IP (quand il rejoint le réseau) à partir d'un serveur

- Peut renouveler le bail d'une adresse en cours d'utilisation
- Permet la réutilisation des adresses (prendre l'adresse uniquement quand connecté et "on")
- Support pour utilisateurs mobiles qui veulent rejoindre le réseau (voir plus loin)

DHCP – vue d'ensemble :

- le poste diffuse un msg "DHCP discover"
- le serveur DHCP répond avec le msg "DHCP offer"
- le poste demande un adresse IP : msg "DHCP request"
- le serveur DHCP envoi une adresse : msg "DHCP ack"

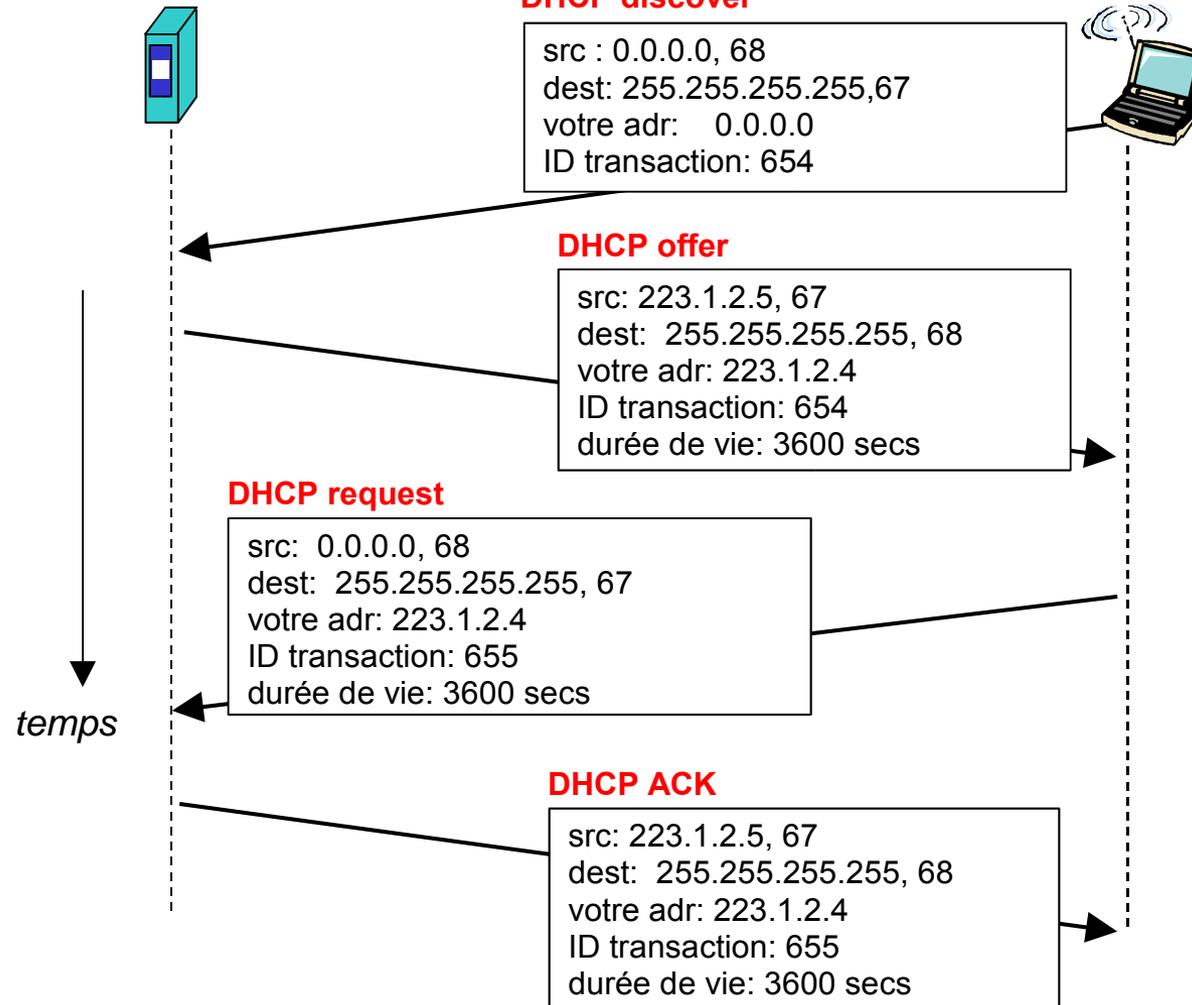
# DHCP : scénario client-serveur



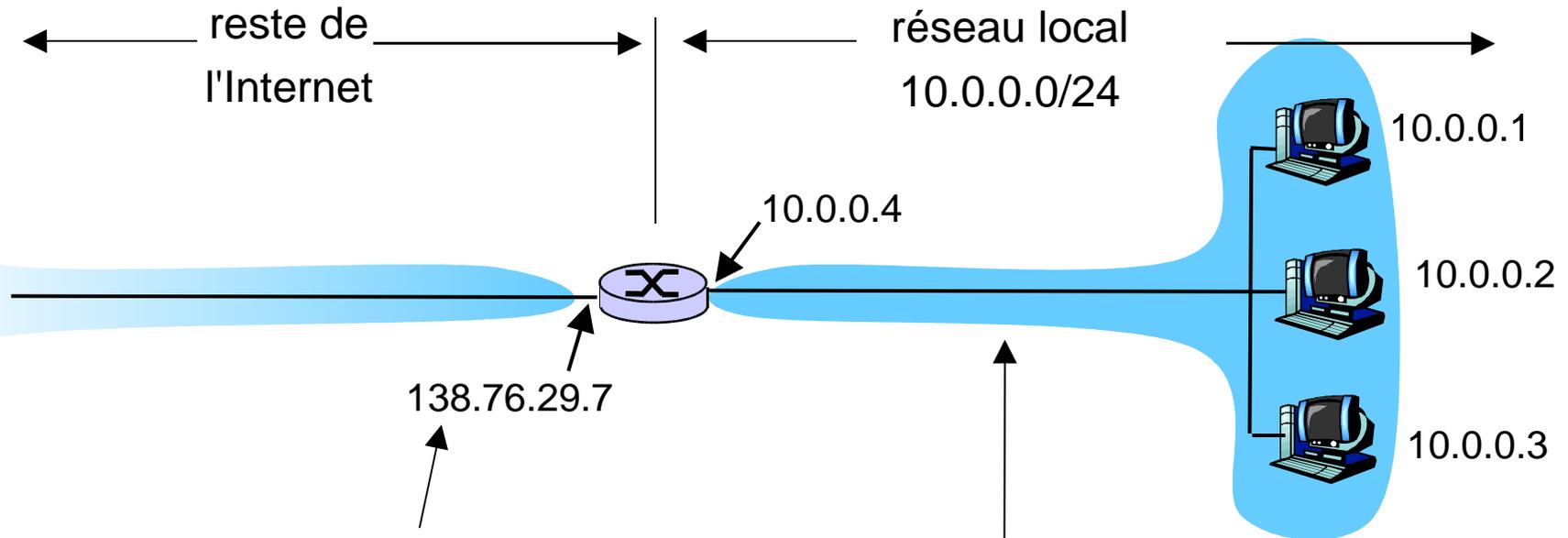
# DHCP : scénario client-serveur

serveur DHCP : 223.1.2.5

nouveau client



# NAT : Network Address Translation



- *Tous* les datagrammes **quittant** le réseau local ont la **même** et unique adresse IP source NAT: 138.76.29.7
- numéros de ports sources différents

- Datagrammes avec la source ou la destination dans ce réseau ayant l'adresse 10.0.0.0/24 pour la source
- destination (comme d'habitude)

# NAT : Network Address Translation

- ♦ **Motivation** : le réseau local utilise juste un adresse IP quand il communique avec l'extérieur :
  - ♦ pas besoin de demander au FAI tout un intervalle d'adresses officielles : une adresse IP est utilisée pour tous les périphériques
  - ♦ on peut changer les adresses des périphériques dans le réseau local sans en avertir le reste du monde
  - ♦ on peut changer de FAI sans bouleverser l'adressage au sein du réseau local
  - ♦ les périphériques du réseau local non explicitement adressables, «invisible» de l'extérieur (un plus de sécurité).

# NAT : Network Address Translation

**Implémentation** : le routeur NAT doit :

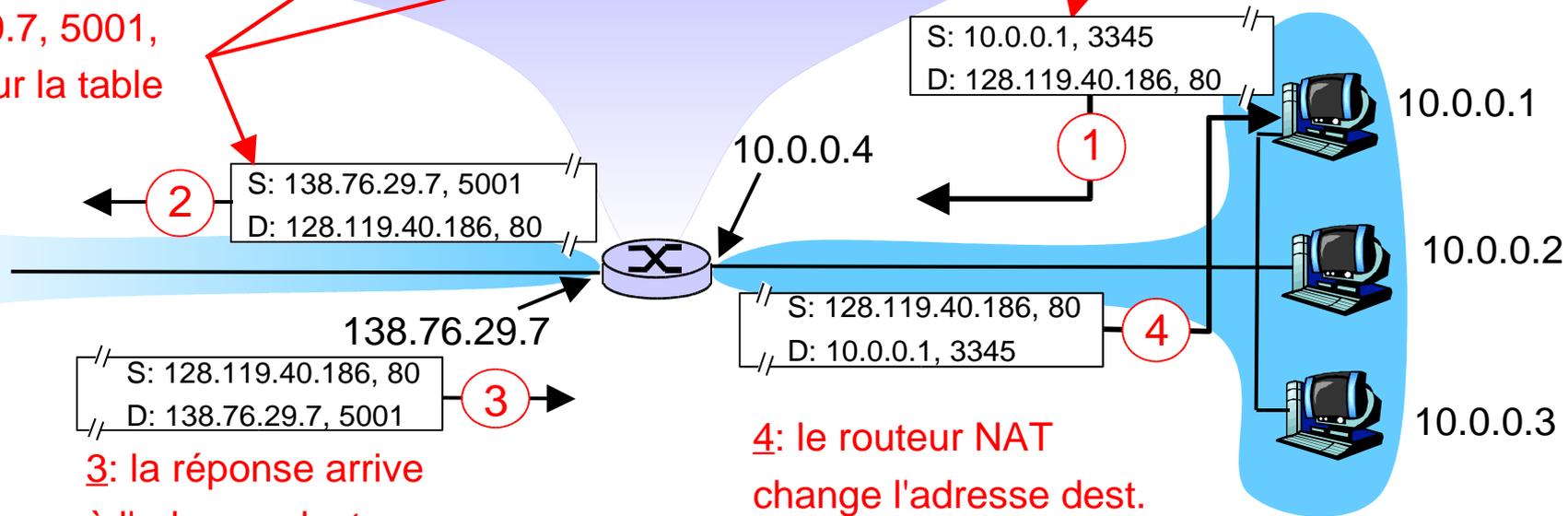
- ♦ *datagrammes sortants : remplacer* (adresse IP, num. port) de tout datagramme sortant par (adresse IP NAT, nouv. num. de port)  
... les clients/serveurs distants répondront en utilisant (adresse IP NAT, le nouv. num. de port) comme adresse de destination
- ♦ *se rappeler (dans la table de traduction NAT)* toutes les pairs de traduction (adresse IP source, num. port) vers (adresse IP NAT, nouv. num. port)
- ♦ *datagrammes entrants : remplacer* (adresse NAT IP, nouv. num. port) dans les champs dest. de tous les datagrammes entrants, par (adresse IP source, num. port) correspondant (couples stockés dans la table NAT)

# NAT : Network Address Translation

table de traduction NAT	
adr côté WAN	adr côté LAN
138.76.29.7, 5001	10.0.0.1, 3345
.....	.....

2: le routeur NAT change l'adresse du datagramme source de 10.0.0.1, 3345 à 138.76.29.7, 5001, mets à jour la table

1: host 10.0.0.1 envoie le datagramme à 128.119.40.186, 80



3: la réponse arrive à l'adresse dest. : 138.76.29.7, 5001

4: le routeur NAT change l'adresse dest. du datagramme de 138.76.29.7, 5001 à 10.0.0.1, 3345

# NAT : Network Address Translation

- ♦ champ numéro de port 16-bit :
  - ♦  $\simeq$  60 000 connexions simultanées avec une seule adresse "officielle" !
- ♦ NAT controversé :
  - ♦ les routeurs doivent travailler uniquement au niveau de la couche 3
  - ♦ viole l'argument end-to-end
    - ♦ la possibilité NAT doit être prise en compte par les développeurs des applis. ex : applications P2P
  - ♦ la pénurie des adresses doit être résolue grâce à l'IPv6

# Chapitre 4 : feuille de route

## 4.1 Introduction

## 4.2 Principes du routage

## 4.3 Routage hiérarchique

## 4.4 IP (Internet Protocol)

## 4.5 Routage dans l'Internet

- routage intra-AS : RIP et OSPF
- routage inter-AS : BGP

## 4.6 IPv6

## 4.7 Routage multicast

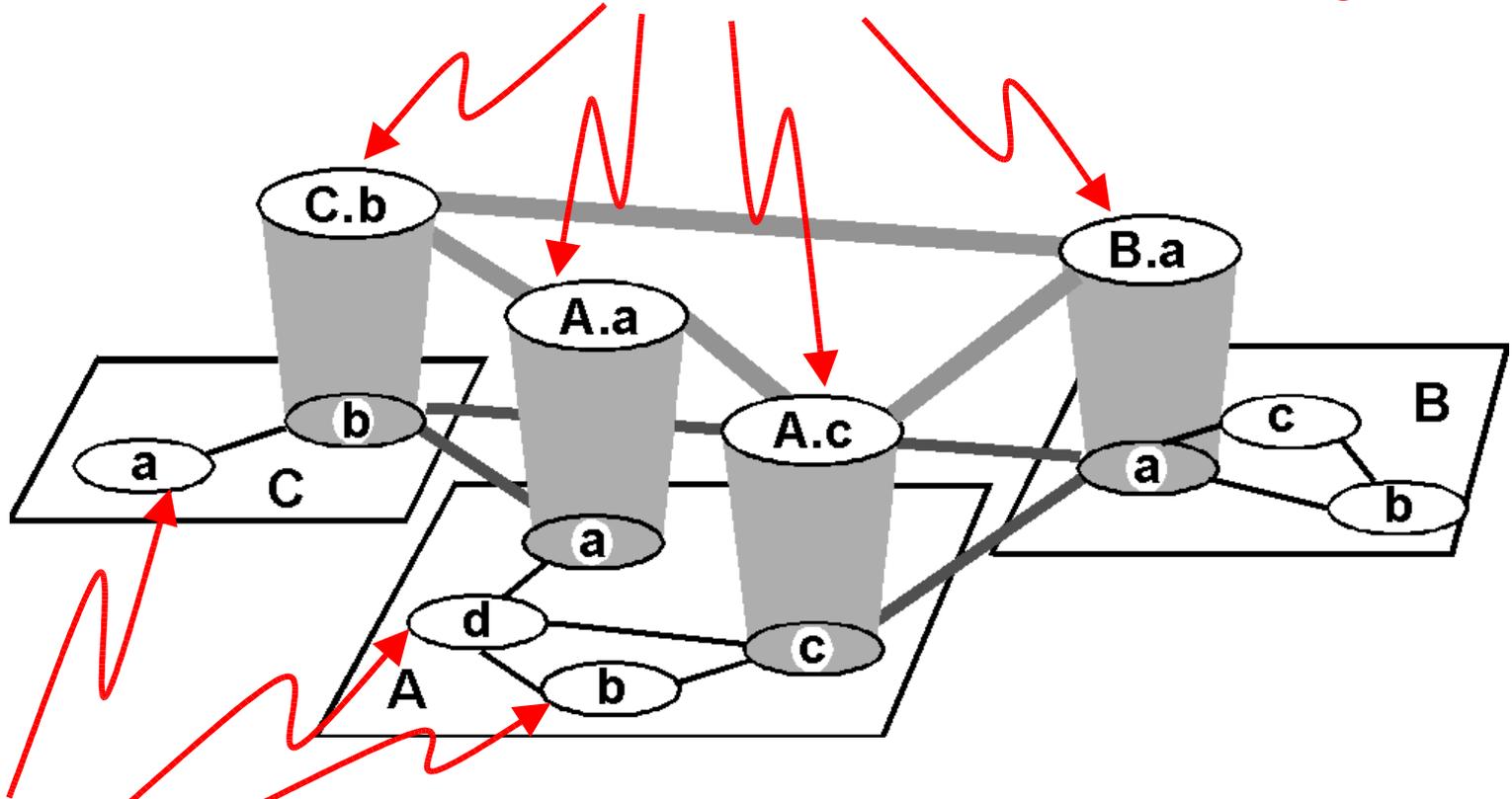
## 4.8 Mobilité

# Routage dans l'Internet

- ♦ L'Internet Global consiste en des **Systemes Autonomes (AS)** interconnectés les uns aux autres :
  - ♦ **AS minuscule** : petite société : une connexion vers les autres AS
  - ♦ **AS grande taille** : grande société (pas de transit): plusieurs connexions aux autres AS
  - ♦ **AS transit** : fournisseur d'accès, fédération de plusieurs AS
- ♦ Deux niveaux de routage :
  - ♦ **Intra-AS** : l'administrateur responsable du choix de l'algorithme de routage à l'intérieur du réseau
  - ♦ **Inter-AS** : standard unique pour le routage inter-AS : BGP

# Hiérarchie des AS dans Internet

routeurs de bordure des Inter-AS (exterior gateway)



routeurs Intra-AS (interior gateway)

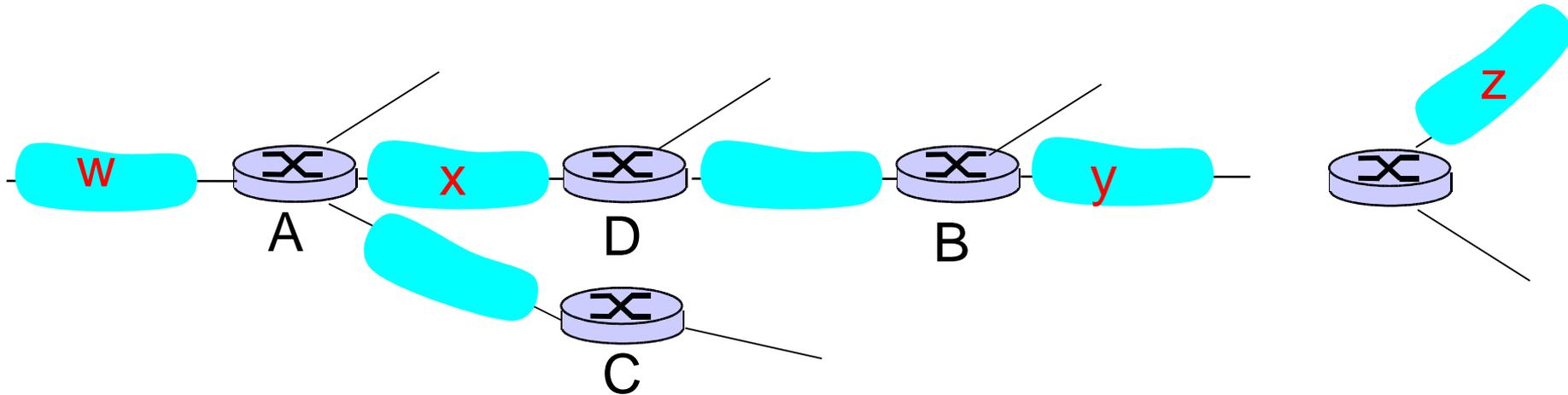
# Routage Intra-AS

- ◆ Connus aussi sous le nom de **Interior Gateway Protocols (IGP)**
- ◆ Les protocoles de routage Intra-AS les plus connus :
  - ◆ RIP: Routing Information Protocol
  - ◆ OSPF: Open Shortest Path First
  - ◆ IGRP: Interior Gateway Routing Protocol (protocole propriétaire CISCO)

# RIP ( Routing Information Protocol)

- ♦ Algorithme Distance Vector
- ♦ Inclus dans la distribution BSD-UNIX en 1982
- ♦ Distance métrique : nombre de sauts (max = 15 sauts)
- ♦ Distance Vectors : échangés parmi les voisins toutes les 30 s via *Response Message* (appelé aussi **publication**)
- ♦ Chaque publication : liste jusqu'à 25 réseaux destination dans l'AS

# RIP : Exemple



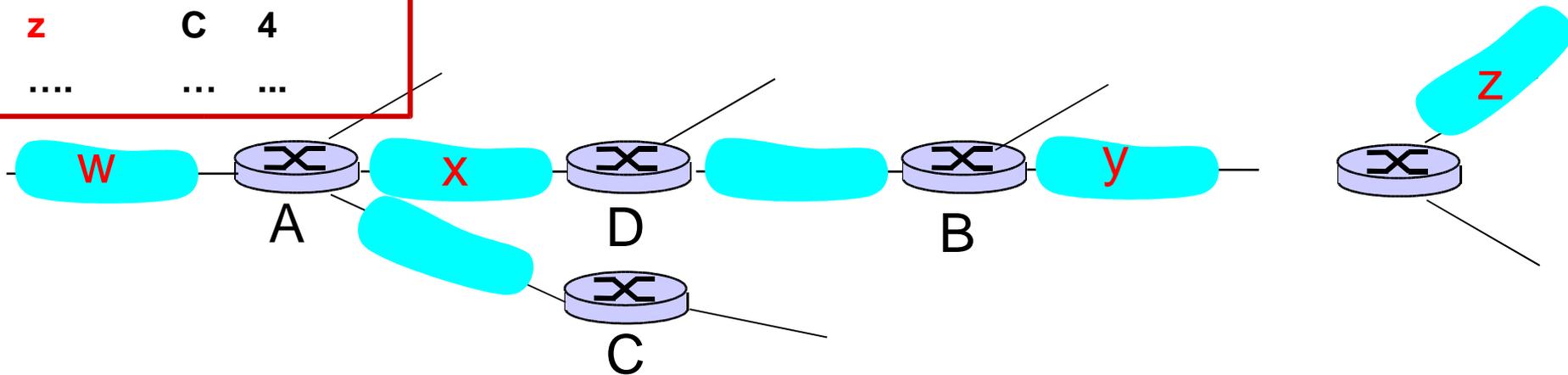
Réseau destination	Prochain routeur	Nbr. de saut à dest.
<b>w</b>	<b>A</b>	<b>2</b>
<b>y</b>	<b>B</b>	<b>2</b>
<b>z</b>	<b>B</b>	<b>7</b>
<b>x</b>	--	<b>1</b>
....	....	....

*Table de routage dans D*

# RIP : Exemple

Dest	PR	sauts
w	-	-
x	-	-
z	C	4
....	...	...

publication  
de A vers D



Réseau destination	Prochain Routeur	Nbr. de saut à dest.
w	A	2
y	B	2
z	<del>B</del> A	<del>7</del> 5
x	--	1
....	....	....

Table de routage dans D

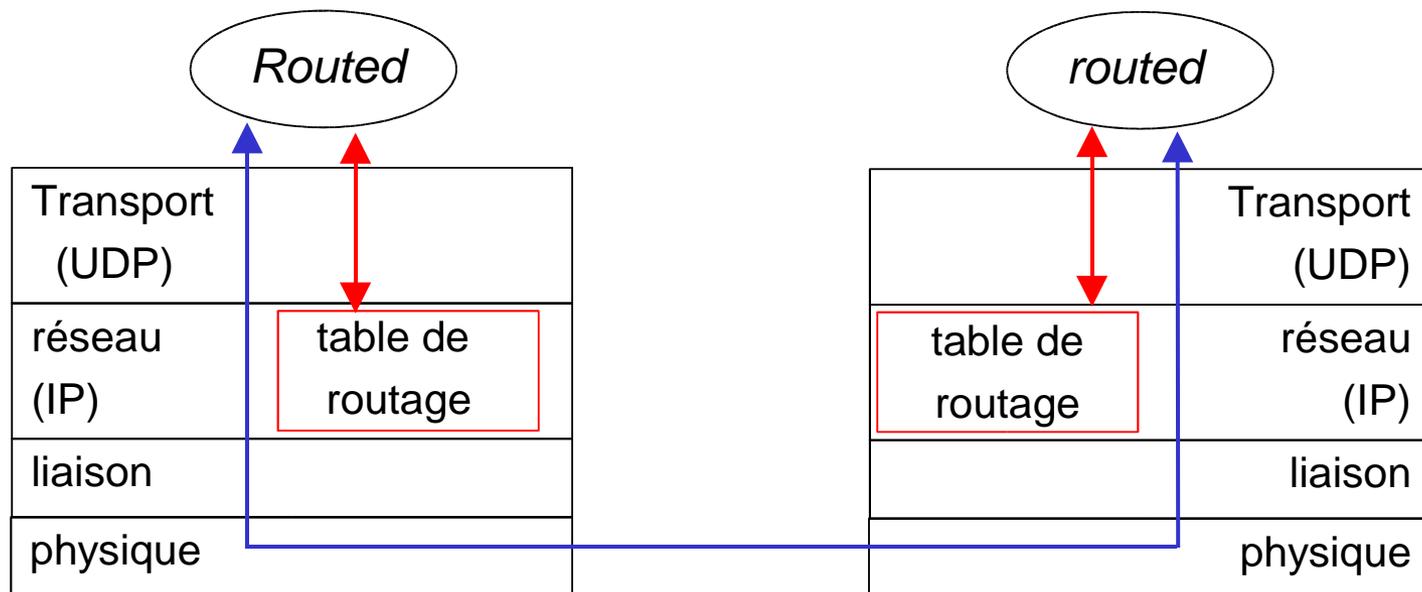
# RIP : Échec de liaison et récupération

Si pas de publication entendue après 180 sec -->  
voisin/lien déclaré mort

- ♦ routes via le voisin invalidées
- ♦ nouvelles publications envoyées aux voisins
- ♦ les voisins à leur tour envoient les nouvelles publications (si les tables changent)
- ♦ l'info «échec de liaison» se propage à tout le réseau
- ♦ distance infinie = 16 sauts

# Table RIP

- ♦ tables de routage RIP gérées par un processus (*daemon*) appelé route-d de la **couche application**
- ♦ publications envoyées dans des paquets UDP, répétés périodiquement



# Table RIP : exemple

Routeur : *giroflée.eurocom.fr*

Destination	Gateway	Flags	Ref	Use	Interface
127.0.0.1	127.0.0.1	UH	0	26492	lo0
192.168.2.	192.168.2.5	U	2	13	fa0
193.55.114.	193.55.114.6	U	3	58503	le0
192.168.3.	192.168.3.5	U	2	25	qaa0
224.0.0.0	193.55.114.6	U	3	0	le0
default	193.55.114.129	UG	0	143454	

- Trois réseaux de classe C attachés (LANs)
- Le routeur ne connaît que les routes vers les LANs attachés
- Un routeur par défaut est utilisé pour aller « ailleurs »
- adresse multicast : 224.0.0.0
- Interface Loopback (l'auto référence (débogage))

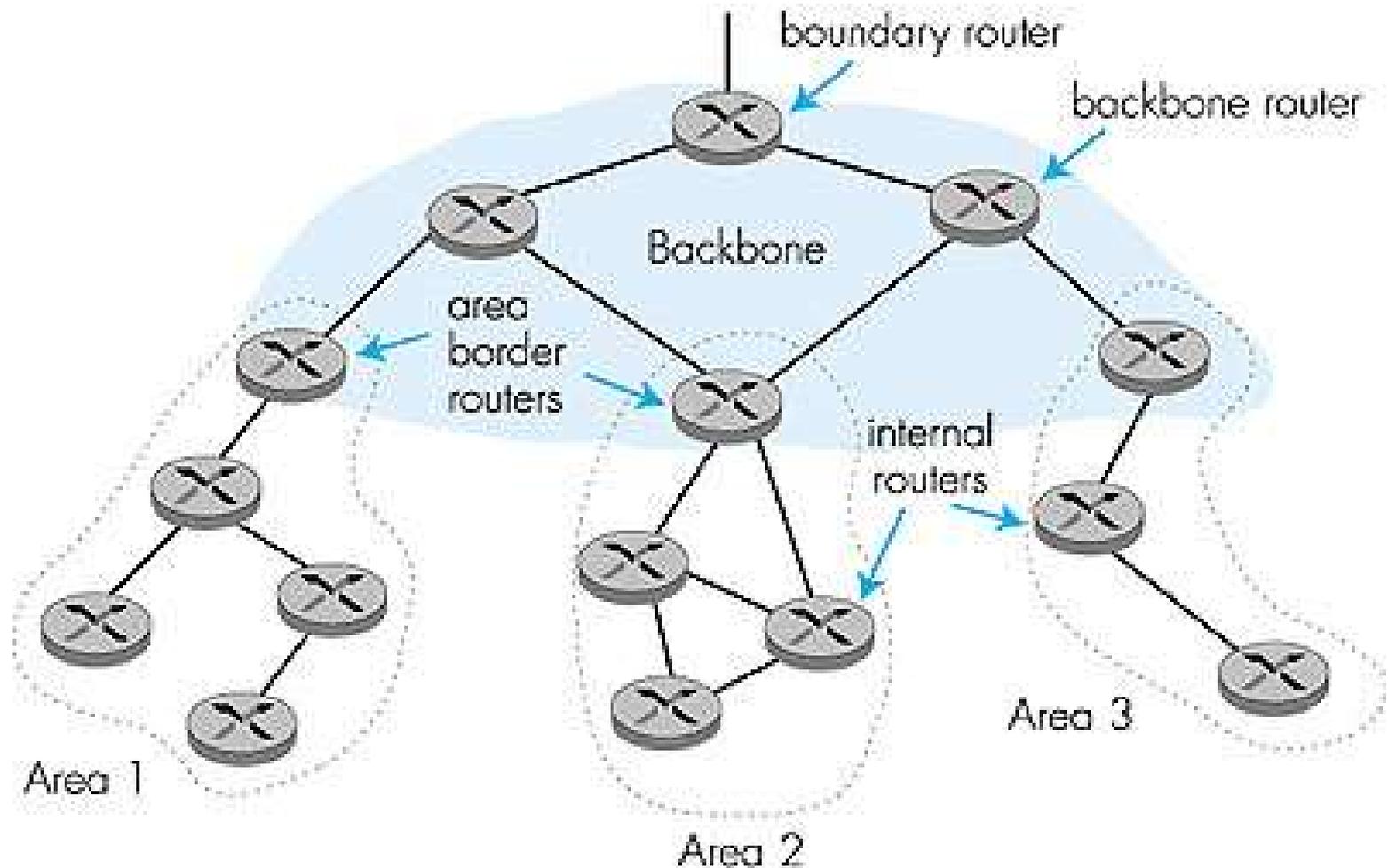
# OSPF (Open Shortest Path First)

- ◆ “open” : dans le domaine public
- ◆ Utilise l'algorithme Link State
  - ◆ Topologie du réseau à chaque noeud
  - ◆ Calcul de la table de routage avec l'algorithme de Dijkstra
- ◆ La publication OSPF porte une entrée par routeur voisin
- ◆ Publications envoyées à **tout** l'AS (par inondation)
  - ◆ Portées par des messages OSPF directement sur IP (au lieu de TCP ou UDP)

# Les *plus* de OSPF par rapport à RIP

- ♦ **Sécurité** : tous les messages OSPF authentifiés (éviter des intrusions)
- ♦ Permet une **multitude** de **chemins** de même coût (un seul chemin dans RIP)
- ♦ Pour chaque lien, métrique multi-coût pour différent **TOS** (ex. coût du lien satellitaire “bas” pour le service “meilleur effort”; “haut” pour le temps réel)
- ♦ Support intégré pour l'unicast et le **multicast** :
  - ♦ OSPF multicast (MOSPF) utilise la même base de données topologique que OSPF
- ♦ OSPF **hiérarchique** dans les grands domaines

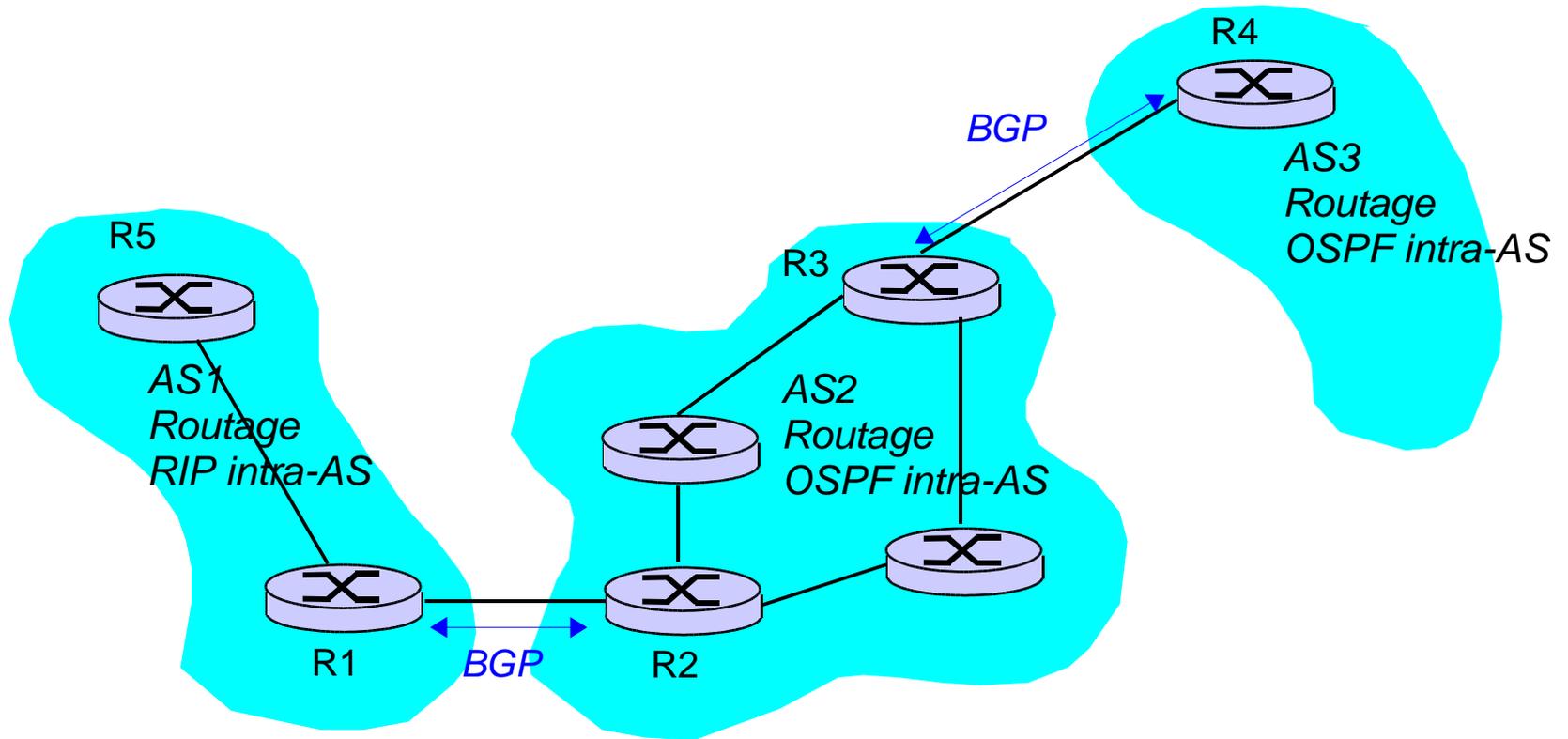
# OSPF hiérarchique



# OSPF hiérarchique

- ♦ **Deux niveaux de hiérarchie** : zone locale, backbone.
  - ♦ publications *link-state* seulement dans la zone
  - ♦ chaque noeud a la topologie détaillée de la zone et seulement la direction connue (le plus court chemin) aux réseaux dans les autres zones
- ♦ **Routeurs en bordure de zone** : “résume” les distances aux réseaux dans sa propre zone, publie vers les autres routeurs en bordure de zone
- ♦ **Routeurs *backbone*** : exécute le routage OSPF limité au backbone
- ♦ **Routeurs de frontière** : connexion vers les autres AS

# Routage Internet Inter-AS : BGP



# Routage Internet Inter-AS : BGP

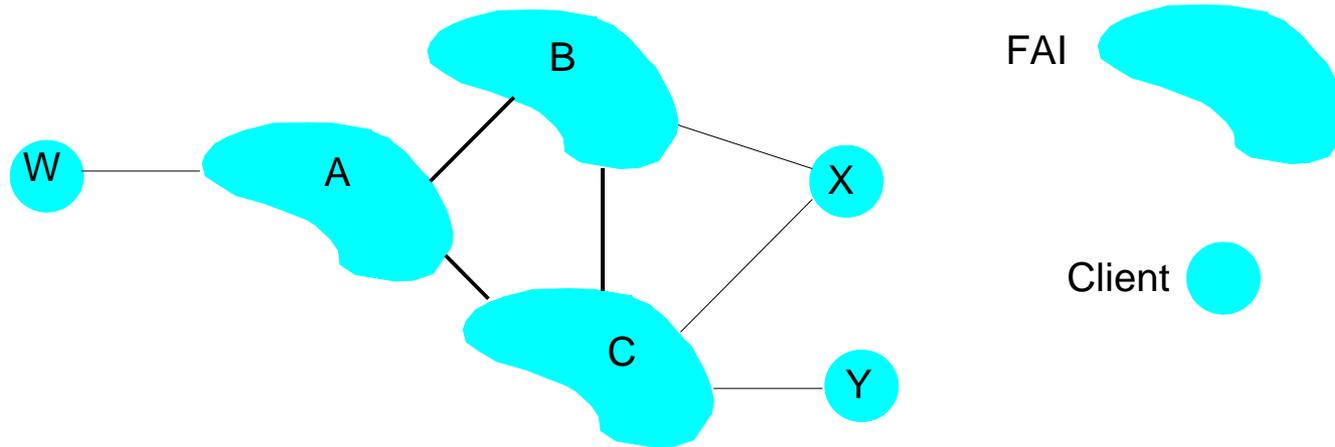
- ♦ **BGP (Border Gateway Protocol):** *Le standard de fait*
- ♦ protocole **Path Vector** :
  - ♦ similaire au protocole *Distance Vector*
  - ♦ chaque *Border Gateway* diffuse au voisins le *chemin entier* (i.e., séquence de AS) vers la destination
  - ♦ BGP route aux réseaux (AS), pas aux postes individuellement
  - ♦ ex. : Gateway X peut envoyer son chemin à la dest. Z :  $\text{chemin}(X,Z) = X, Y1, Y2, Y3, \dots, Z$

# Routage Internet Inter-AS : BGP

*Supposons* : gateway X envoi son chemin au voisin gateway W

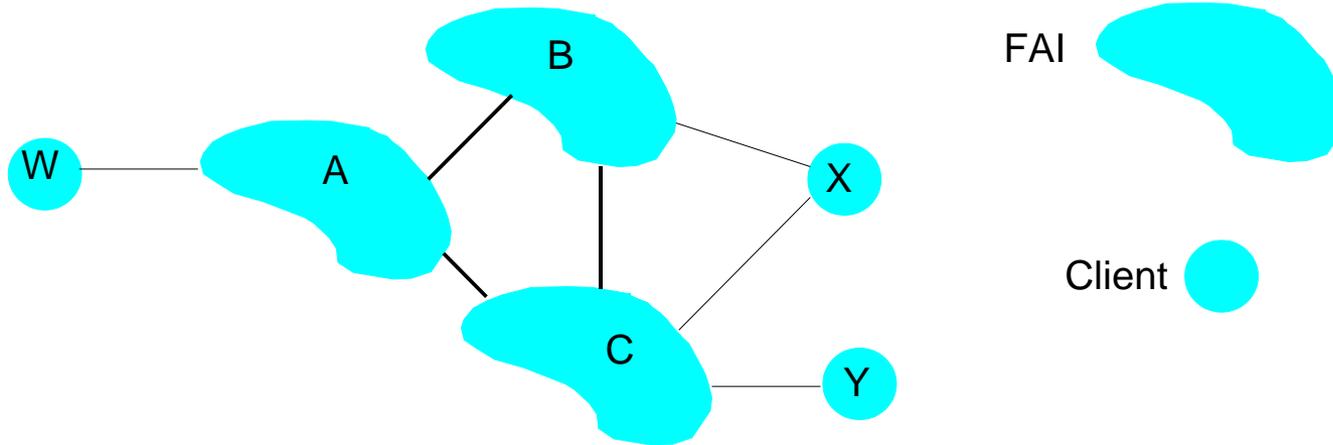
- ♦ W peut ou pas sélectionner le chemin offert par X
  - ♦ coût, règle (ne pas router via des As concurrents), éviter des boucles.
- ♦ Si W sélectionne le chemin publié par X, alors :
$$\text{chemin}(W,Z) = w, \text{ chemin}(X,Z)$$
- ♦ Note : X peut contrôler le trafic entrant en contrôlant les publications (aux voisins) des routes en sa possession:
  - ♦ ex. : ne veut pas router le trafic à Z -> n'informe Z d'aucune route

# BGP : contrôler qui route vers vous



- ♦ A,B,C sont des **réseaux de fournisseurs**
- ♦ X,W,Y sont des clients (de ces fournisseurs)
- ♦ X attaché à deux réseaux (**dual-homed**)
  - ♦ X ne veut pas router depuis B via X vers C
  - ♦ .. donc X n'informe pas B de sa route vers C

# BGP : contrôler qui route vers vous



- ♦ A annonce à B le chemin AW
- ♦ B annonce à W le chemin BAW
- ♦ B doit-il annoncer à C le chemin BAW ?
  - ♦ Sans issue ! B ne reçoit pas de retour pour le routage CBAW puisque ni W ni C ne sont des clients de B
  - ♦ B veut forcer C à router vers w via A
  - ♦ B veut router *uniquement* vers/depuis ses clients !

# Fonctionnement de BGP

## Q : Que fait un routeur BGP ?

- ◆ Reçoit et filtre les annonces de routes depuis les voisins qui lui sont directement attachés
- ◆ Sélectionne une route
  - ◆ pour router à une destination X, quel chemin (parmi ceux annoncés) faut-il prendre ?
- ◆ Envoi les annonces de routes aux voisins

# Messages de BGP

- ♦ Les messages BGP échangés en utilisant TCP.
- ♦ Messages BGP :
  - ♦ **OPEN**: ouvre une connexion TCP au correspondant et authentifie l'émetteur
  - ♦ **UPDATE**: annonce un nouveau chemin (ou retire un ancien)
  - ♦ **KEEPALIVE**: garde la connexion en l'absence des UPDATES; acquitte aussi la requête OPEN
  - ♦ **NOTIFICATION**: rapporte les erreurs vues dans les messages précédents; utilisé aussi pour fermer la connexion

# Intra- et Inter-AS: Pourquoi des routages différents ?

## Politique :

- ♦ Inter-AS: l'admin veut contrôler comment le trafic est routé, comment router à travers son réseau
- ♦ Intra-AS: un seul admin, pas besoin de décision politique

## Taille :

- ♦ le routage hiérarchique réduit la taille des tables, réduit donc le trafic de mise-à-jour

## Performance :

- ♦ Intra-AS: peut se concentrer sur la performance
- ♦ Inter-AS: la politique peut dominer la performance

# Chapitre 4 : feuille de route

4.1 Introduction

4.2 Principes du routage

4.3 Routage hiérarchique

4.4 IP (Internet Protocol)

4.5 Routage dans l'Internet

4.6 IPv6

# IPv6

- ♦ **Motivation initiale** : l'espace d'adressage 32-bit *complètement alloué* vers 2008.
- ♦ Motivation Additionnelle :
  - ♦ le format des en-têtes aide dans la rapidité du traitement/transfert
  - ♦ changer l'en-tête pour faciliter la QoS (Quality of Service)
  - ♦ nouvelle adresse “anycast” : route vers le “meilleur” parmi plusieurs serveurs doublés
- ♦ **Format du datagramme IPv6** :
  - ♦ en-tête à longueur fixe : 40 octets
  - ♦ pas de fragmentation fournie

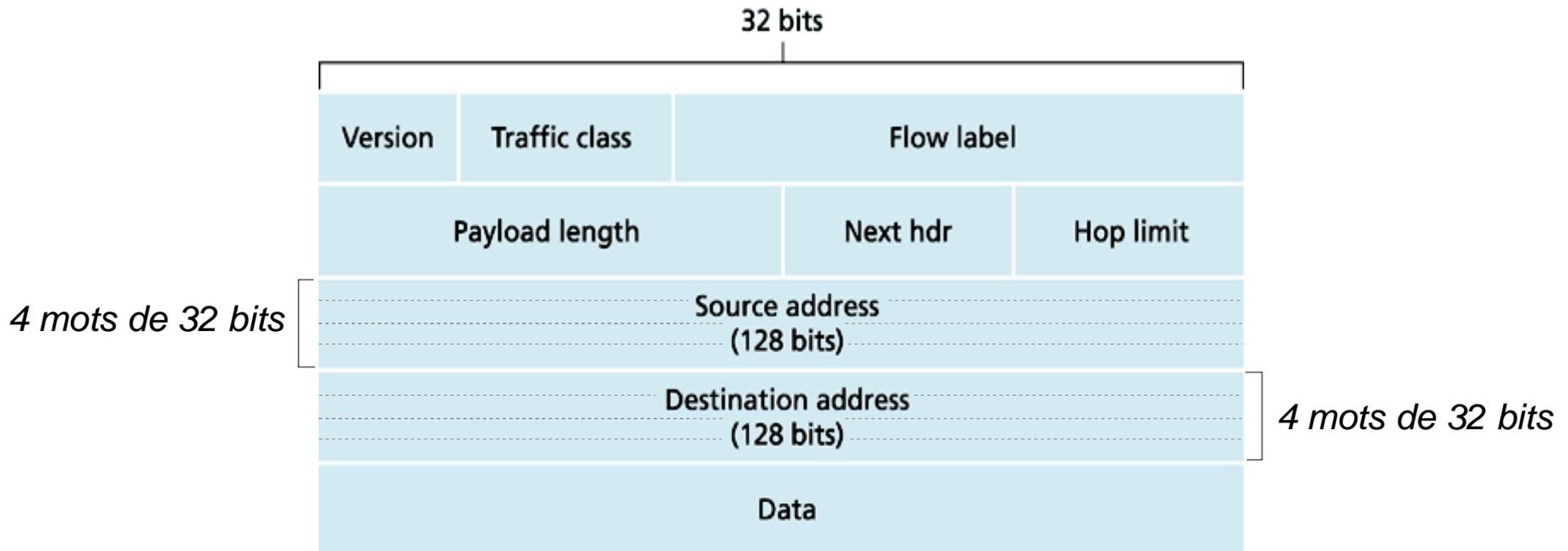
# IPv6 : Quelques chiffres

- ♦ Le nombre d'adresses théoriques dans IPv6 (codées sur 128 bits) est :
  - ♦ 340 milliards de milliards de milliards de milliards
  - ♦ Plus précisément :  
**340 282 366 920 938 463 463 374 607 431 701 211 156**
- ♦ Pour être plus parlant :

Si on recouvrait la surface de la terre d'une couche de sable de 50 km d'épaisseur (jusqu'en haut de la stratosphère), et que l'on attribue une adresse IPv6 à chaque grain de sable, on n'utiliserait qu'environ deux cent milliardième des adresses disponibles.

# En-tête IPv6

- Traffic class** identifie les datagrammes prioritaires parmi le flot
- Flow Label** identifie les datagrammes dans le même "flot" ("flot" pas bien défini).
- Payload length** taille des données (champ Data)
- Next header** identifie le protocole de la couche supérieur pour les données
- Hop limit** nombre de routeurs traversé (au delà duquel le paquet est jeté)



# Les différentes entêtes d'adresses IPv6

- ◆ Les premiers bits des adresses permettent de savoir le type de paquet dont il s'agit :
  - ◆ **96 zéros** suivi d'une adresse sur 32 bits : c'est une adresse IPv4 (compatibilité)
  - ◆ **001** : paquet *global unicast* (envoyé à une seule destination sur l'Internet)
  - ◆ **1111 1110 10** : paquet *link local unicast*, envoyé à une destination directement reliée (un périphérique à côté par exemple)
  - ◆ **1111 1110 11** : paquet *site local unicast*, envoyé à une destination située sur le même site (même réseau)
  - ◆ **1111 1111** : paquet *multicast* envoyé simultanément à plusieurs destinations (par exemple dans le cas de diffusion vidéo)
- ◆ Les autres préfixes non encore attribués sont réservés pour le futur. Cela représente 7/8 de la totalité des adresses

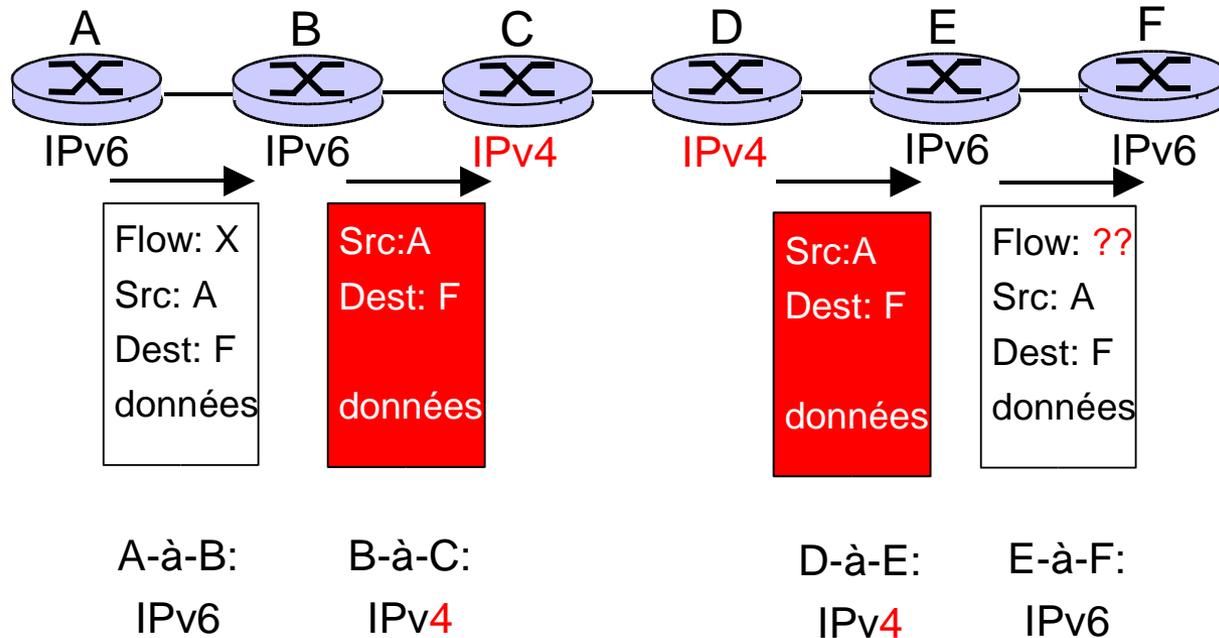
## Autres changements par rapport à l'IPv4

- ♦ **Checksum:** retiré complètement afin de réduire le temps de traitement à chaque saut
- ♦ **Options:** permis, mais en dehors de l'en-tête (indiqué par le champ “Next Header”)
- ♦ **ICMPv6:** nouvelle version de ICMP
  - ♦ types de messages additionnels, ex. “Paquet Trop Grand” (*Packet Too Big*)
  - ♦ fonctions de gestion des groupes multicast

# Transition de l'IPv4 à l'IPv6

- ♦ Tous les routeurs ne peuvent pas être mis à jour simultanément
  - ♦ pas de “flag days”
  - ♦ Comment le réseau peut opérer dans un environnement mixte de routeurs IPv4 et IPv6 ?
- ♦ Deux approches proposées :
  - ♦ *Double Piles* : certains routeurs avec une double pile (v6, v4) peuvent faire la “traduction” entre formats
  - ♦ *Tunneling* : les datagrammes IPv6 transportés dans des datagrammes IPv4 parmi les routeurs Ipv4

# Approche Double Pile

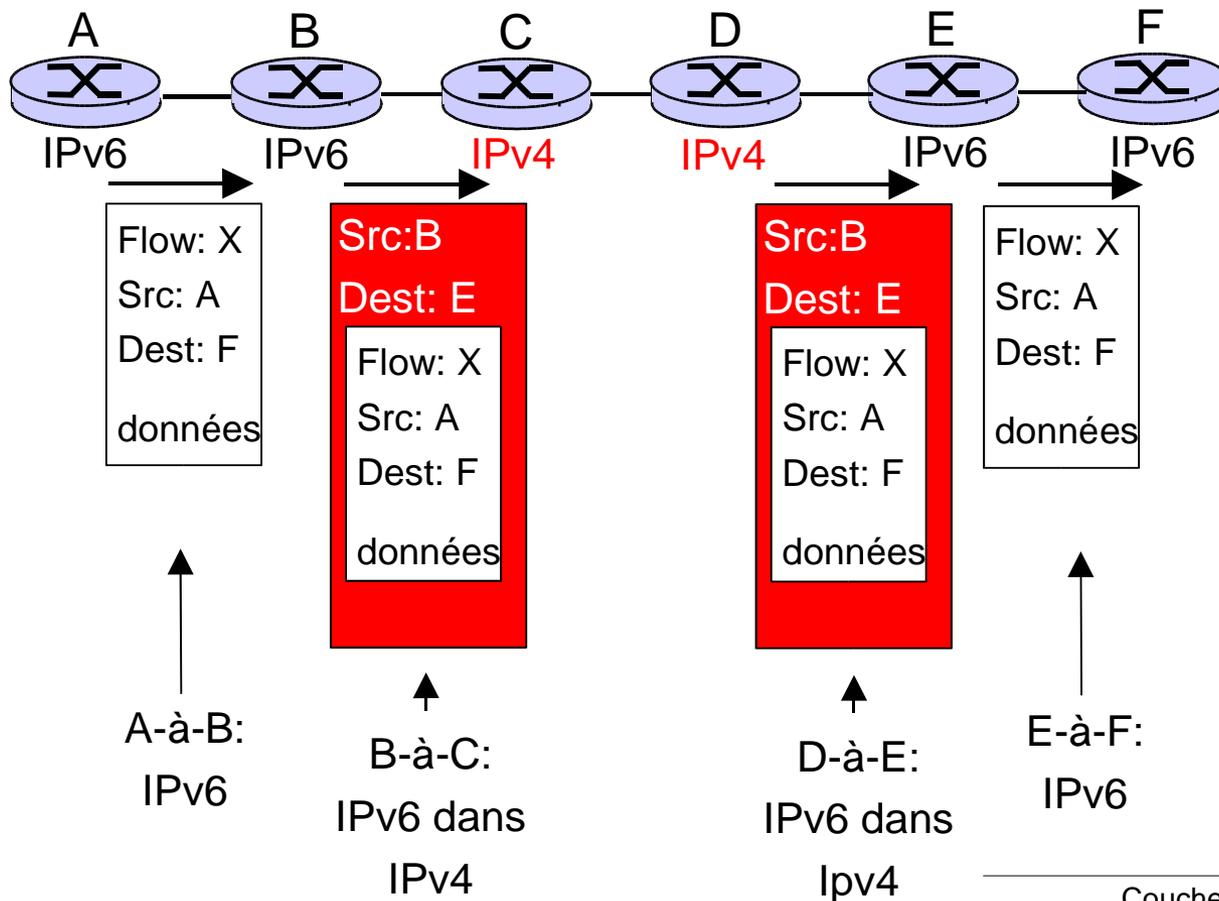


# Tunneling

Vue logique:



Vue physique:



# Fin

## Ce qui a été couvert :

- ♦ services de la couche réseau
- ♦ principes de routage : *Link State* et *Distance Vector*
- ♦ routage hiérarchique
- ♦ IP (IPv4)
- ♦ protocoles de routage Internet : RIP, OSPF, BGP
- ♦ IPv6

La suite ?

La couche :

Liaison de Données