– PANTHÉON - SORBONNE –
UNIVERSITÉ PARIS 1

INF2 – RÉSEAUX INFORMATIQUES

Couche Réseau

- Routage

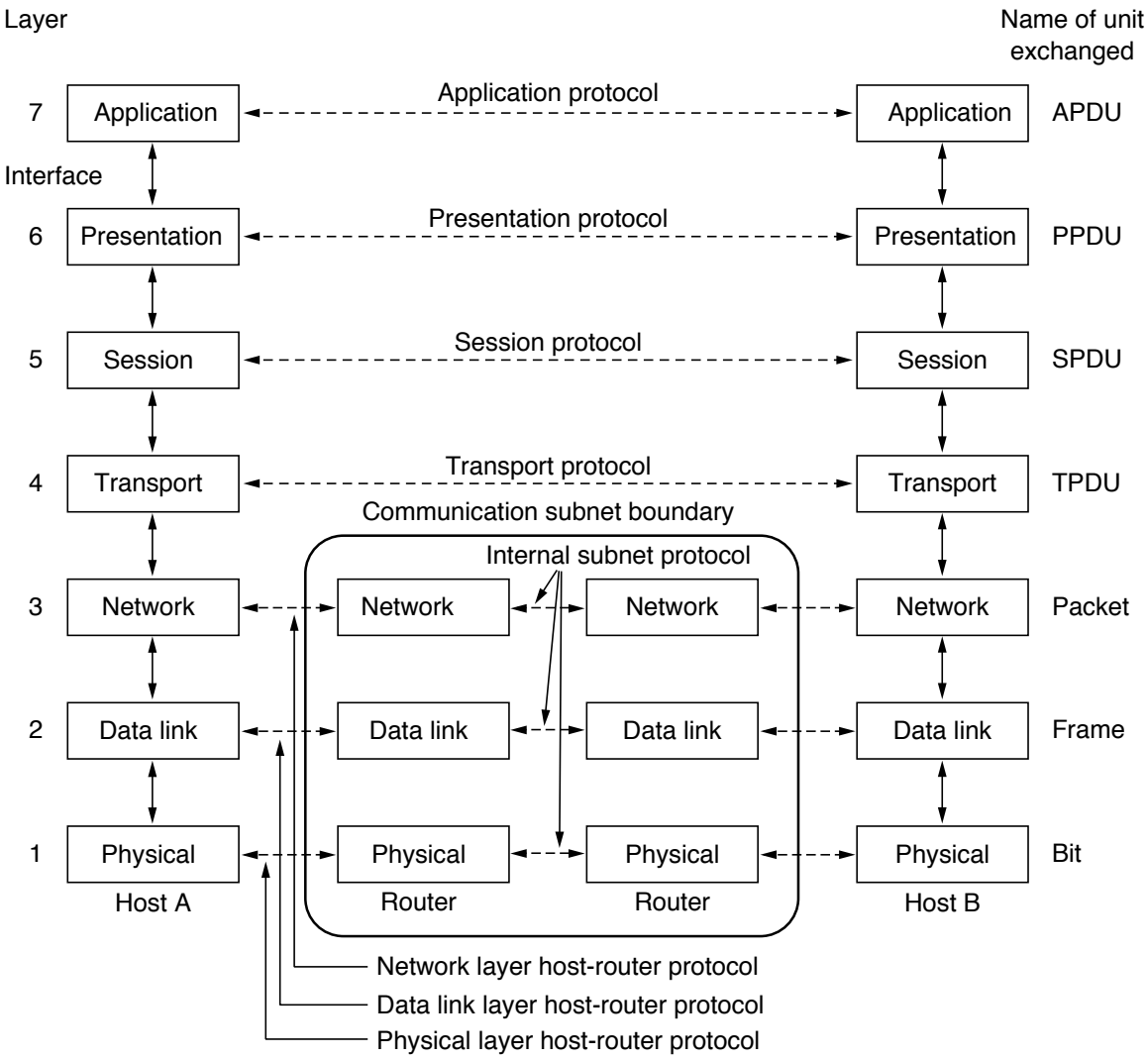
Plan Couche Réseau

- Présentation
- Routage
- Rappel sur l'adressage IP

La Couche Réseau

- Rôles
  - Interface du sous-réseau de communication
  - Détermination du chemin des paquets
    - source et destination dans le même sous-réseau
    - source et destination dans différents sous-réseaux
  - Contrôle de congestion
  - Résolution des problèmes d'interconnexion
  - Résolution des problèmes d'hétérogénéité

Services pour la Couche Transport



Implémentation des Services

- La couche Réseau est la première couche indépendante du matériel physique
  - Les services fournis doivent être indépendants
    - De la technologie d'acheminement (routeurs)
    - Du nombre et du type d'équipements réseau
  - La topologie doit être transparente du point de vue de la couche Transport
  - L'adressage doit être uniforme à travers la totalité du réseau
- Deux modes de service
  - Connectionless – services sans connexion
  - Connection-oriented – services avec connexion

Mode sans Connexion

- Communauté Internet
  - +40 ans d'expérience des réseaux informatiques
  - Le réseau fait le minimum :
    - Service sous-jacent non-fiable
    - L'hôte gère la fiabilité (détection et correction des erreurs) et le contrôle de flux
- Mode sans connexion
  - primitives : *SEND_PACKET* et *RECEIVE_PACKET*
  - pas de fonctionnalités qui se trouvent dans les hôtes (réordonnancement, contrôle de flux...)

Mode Orienté Connexion

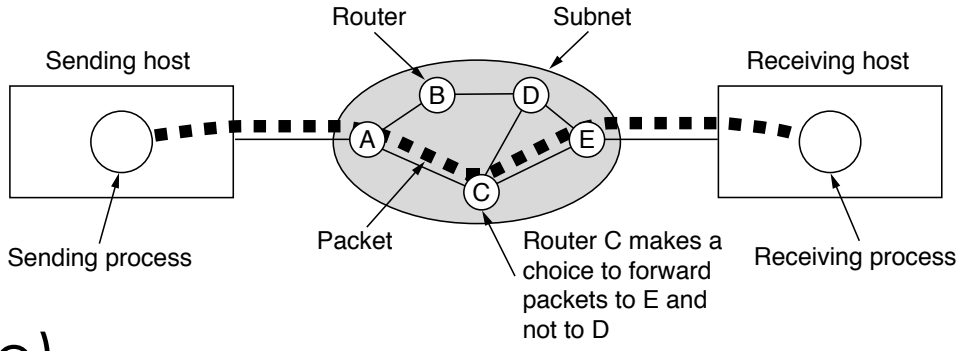
- Communauté Téléphonique
  - 100 ans d'expérience de réseaux non-informatiques
  - Le réseau fait le maximum :
    - Service sous-jacent fiable
- Mode Orienté Connexion
  - Création d'une connexion préalable, maintenue pendant toute la communication et explicitement terminé
  - Négociation des paramètres de qualité de service à l'établissement
  - Communication bidirectionnelle et ordonnée
  - Contrôle de congestion systématique

Deux Techniques de Support

- Technique basée sur les Datagrammes
  - permet de construire un service en mode non-connecté ou orienté connexion
  - généralement associé au mode sans connexion
- Technique basée sur les Circuits Virtuels
  - permet de construire un service en mode orienté connexion ou non
  - généralement associé au mode orienté connexion

Service par Datagrammes

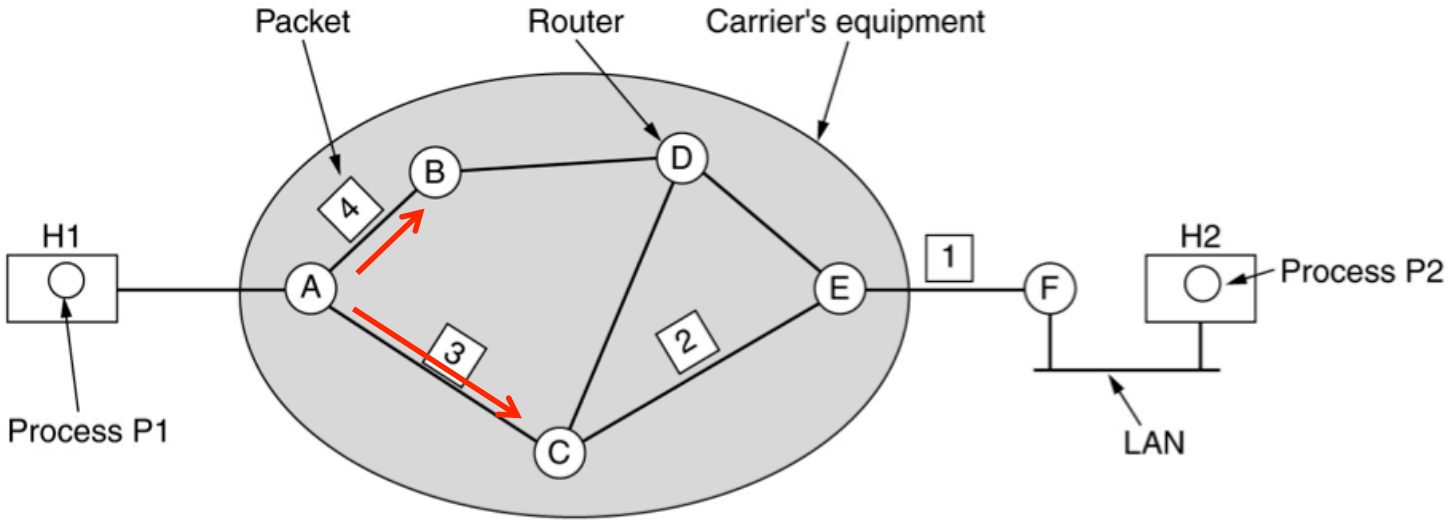
- Analogie avec le service postal
- Buts :
  - pas de mémorisation du trajet d'un type de paquet
  - possibilité de routes diverses
  - robustesse
  - efficacité (multiplexage)
  - tarification au débit



Service par Datagrammes

- Principes du routage
  - Chaque paquet est routé individuellement
  - Prérequis : chaque paquet doit transporter ses adresses
- Avantages
  - Implémentation simplifiée (vision locale)
  - Tolérance aux pannes
- Inconvénients
  - Garanties limitées sur le service de livraison

Services sans Connexion



A

Dest.	Line
A	–
B	B
C	C
D	B
E	B
F	B
F	C

C's table

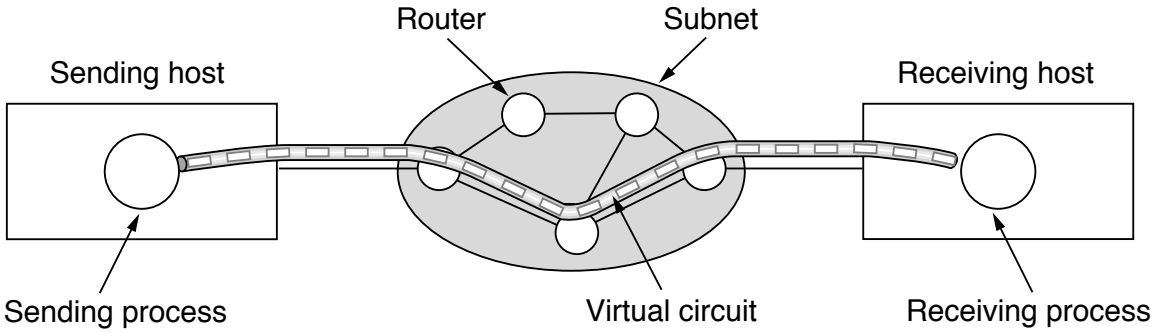
A	A
B	A
C	–
D	D
E	E
F	E

E's table

A	C
B	D
C	C
D	D
E	–
F	F

Services avec Circuit Virtuel

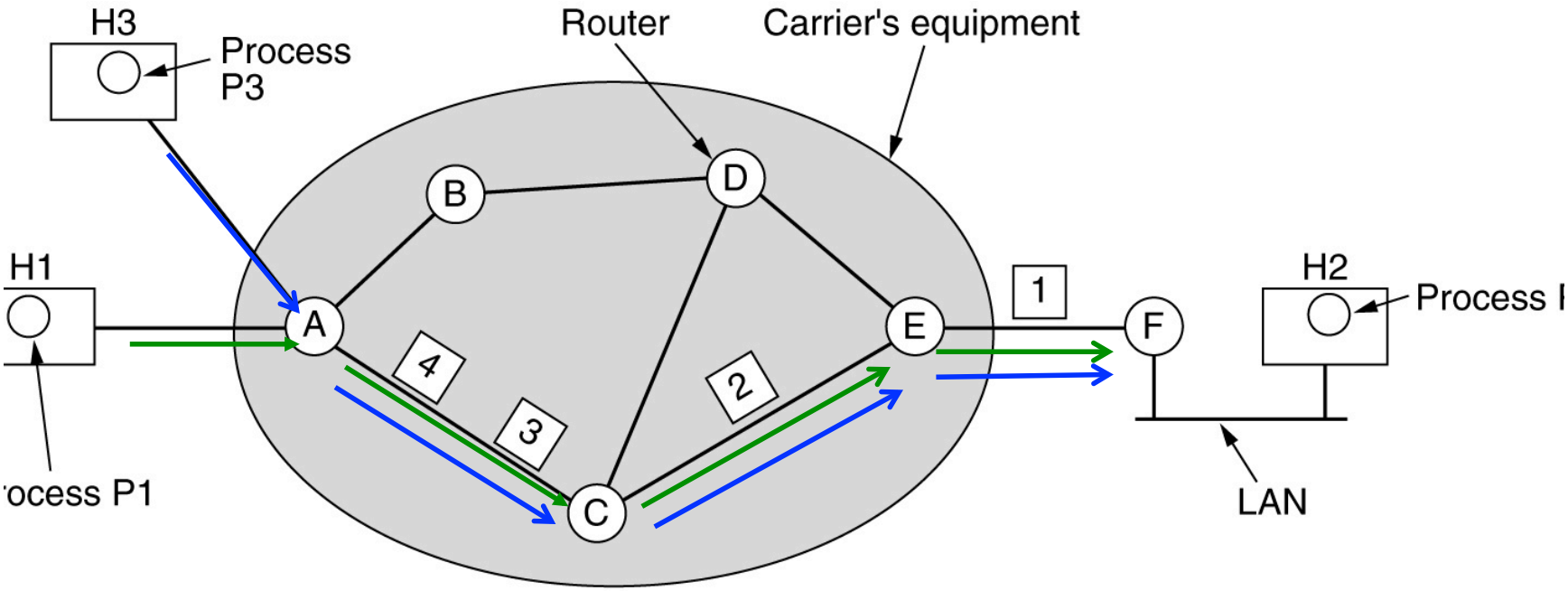
- Analogie avec les circuit physiques du téléphone
- Buts :
  - éviter de recalculer la route pour chaque paquet
  - traitement efficace des paquets (temps réel possible)
  - tarification à la durée



Services avec Circuit Virtuel

- Mécanisme
  - Établissement d'une connexion "bout-à-bout"
- Principe
  - Éviter de prendre une décision à chaque paquet envoyé/retransmis
- Avantages
  - Garanties sur le chemin choisi
- Inconvénients
  - Coût d'établissement d'une connexion
  - Reconnexion en cas de panne

Services avec Circuit Virtuel



A's table

H1	1	C	1
H3	1	C	2

In Out

C's table

A	1	E	1
A	2	E	2

E's table

C	1	F	1
C	2	F	2

Comparaison

Aspect	Datagrammes	Circuit Virtuel
Initialisation	Non	Requise
Adressage	Chaque paquet contient les adresses source et destination	Les paquets contiennent l'identifiant de la connexion virtuelle
Routage	Chaque paquet est routé indépendamment	La route est choisie lors de l'établissement du CV
Panne d'un routeur	Les paquets sont routés vers des routeurs alternatifs	Tous les CV sont supprimés
Qualité de Service	Difficile à garantir	Allocation par avance lors de l'établissement du CV
Contrôle de congestion	Difficile	Simple

Plan Couche Réseau

- Routage
  - **Principes de base**
  - Inondation
  - Routage par vecteur de distance
  - Routage par état des liaisons
  - Routage hiérarchique
  - Routage multicast
  - Routage dans les réseaux mobile, ad-hoc et P2P

Principes du Routage

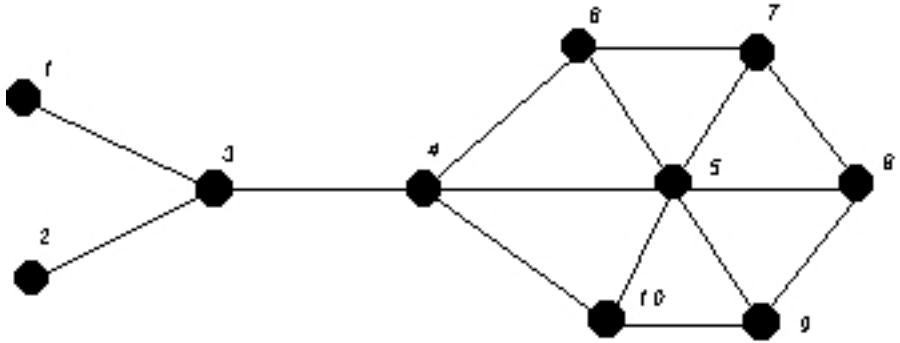
- Acheminement des paquets de la source à la destination quelque soit le nombre de saut
- Fonction principale de la Couche Réseau
- Utilisation d'un algorithme de routage
  - choix de l'interface de sortie
    - pour chaque paquet pour les datagrammes
    - pour le 1er paquet d'un circuits virtuels (*session routing*)

Protocole de Routage ou Protocole Routé ?

- Protocoles routés
  - Tout protocole qui n'est pas capable (tout seul) d'atteindre une destination distante
    - Ex : TCP, HTTP, IP
- Protocoles de routage
  - Protocoles auxiliaires qui permettent le routage des paquets
    - Diffusion de l'information, construction des tables de routage, meilleur chemin
    - Ex : RIP, OSPF, EIGRP, IS-IS, BGP
- Certains protocoles de routage utilisent les services de protocoles routés pour l'établissement de connexions sur le segment
  - Ex : RIP et OSPF utilisent UDP/IP, BGP utilise TCP/IP
- D'autres protocoles de routage utilisent leurs propres services de transport, indépendants des protocoles routés
  - Ex : EIGRP utilise RTP, IS-IS utilise CLNP

Les Routage

- Les 3 commandements
  - Chaque routeur décide indépendamment sur le routage des paquets, grâce aux informations contenues dans sa propre table de routage
  - Le fait qu'un routeur détient une information dans sa table de routage ne veut pas dire que les autres routeurs la détiennent aussi
  - L'information du chemin de routage entre un réseau A et un réseau B (A→B) ne permet pas d'inférer sur le chemin contraire (B→A)

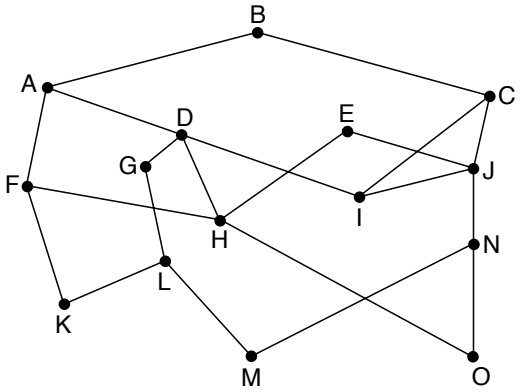


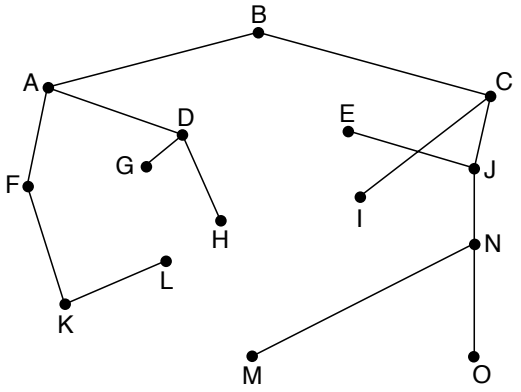
Propriétés des Algorithmes de Routage

- Exactitude (*correctness*)
- Simplicité (*simplicity*)
- Robustesse (*robustness*)
  - doit toujours fonctionner
- Stabilité (*stability*)
  - convergence rapide vers un équilibre
- Équité (*fairness*)
  - même traitement pour tout les paquets
- Optimalité (*optimality*)

Principe d'Optimalité

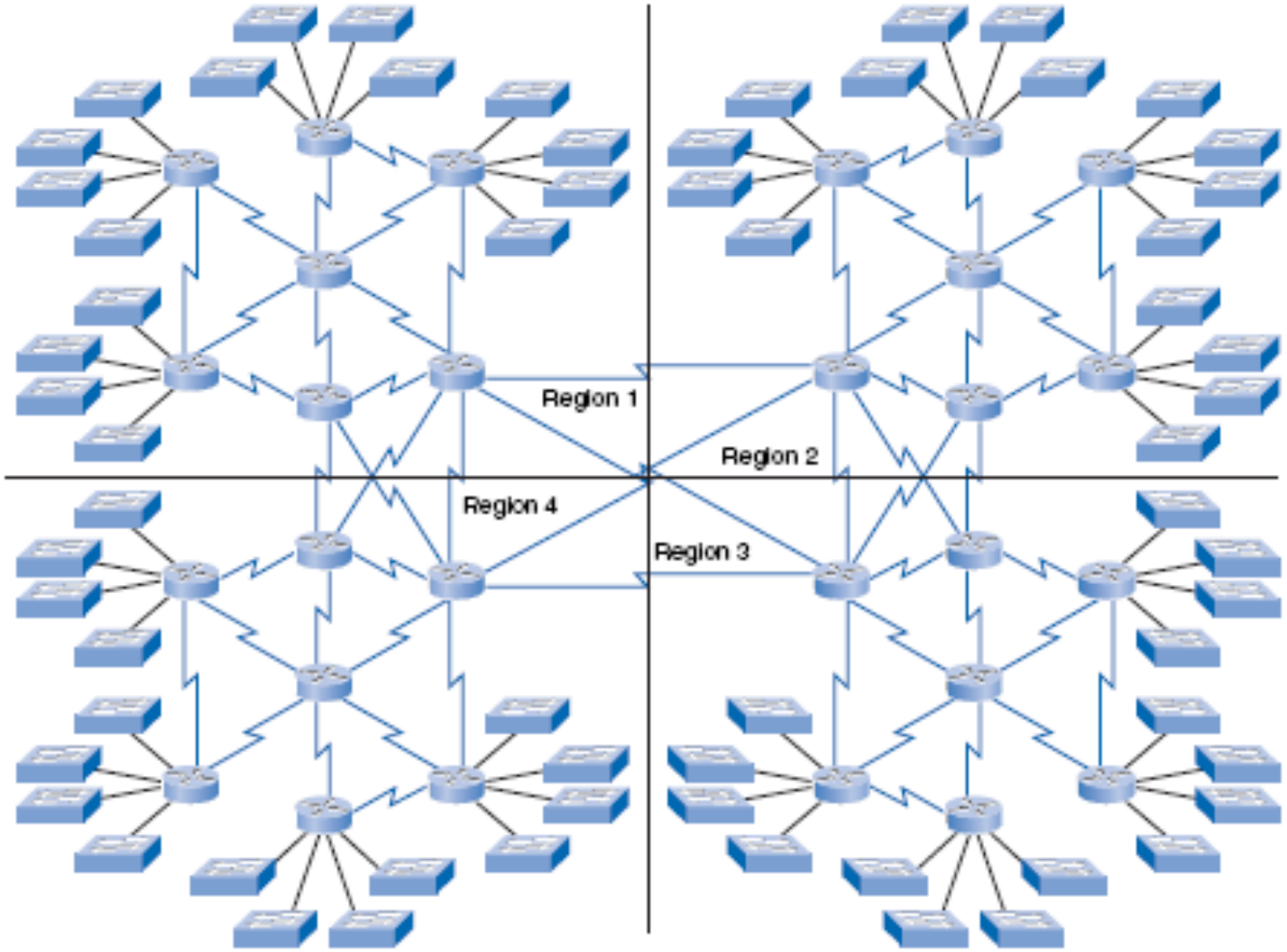
- Si B est sur la route optimale de A à C, alors la route optimale de B à C suit la même route
- Conséquence : les routes optimales vers une destination donnée forme un arbre (sink tree)
  - pas de boucle
  - nombre de sauts fini





Types d'Algorithme de Routage

- Routage statique
  - non adaptatif
  - décisions pré-calculées et chargées initialement dans le routeur
- Routage dynamique
  - Adaptatif
  - décisions de routage évoluant selon les modification de topologie et de trafic (information locale, de routeur adjacent ou globale)



- Imaginez-vous gérer les liens statiques dans un réseau pareil
- Qu'est-ce que se passe si un lien tombe à 3h du matin ?

Plan Couche Réseau

- Routage
  - Principes de base
  - **Inondation**
  - Routage par vecteur de distance
  - Routage par état des liaisons
  - Routage multicast
  - Routage hiérarchique
  - Routage dans les réseaux mobile, ad-hoc et P2P

Algorithme de Flooding

- Algorithme de routage "naïf"
  - Envoi vers toutes les interfaces sauf celle d'arrivée
  - Stop : uniquement se le paquet a déjà été vu
- Simple mais coûteux
  - Le nombre de messages est énorme
- Robuste, finit toujours par choisir le meilleur chemin
  - Toutes les machines sont visitées

Plan Couche Réseau

- Routage
  - Principes de base
  - Inondation
  - **Routage par vecteur de distance**
  - Routage par état des liaisons
  - Routage hiérarchique
  - Routage multicast
  - Routage dans les réseaux mobile, ad-hoc et P2P

Routage par Vecteur de Distance

- Algorithme de routage dynamique
  - Chaque routeur maintient une table (**vecteur**) de distances vers chaque destinataire possible avec l'interface à utiliser
    - Destination
    - Distance
    - Direction (interface réseau)
- Utilise l'algorithme de routage distribué de Bellman-Ford
- Algorithme de routage initial d'ARPANET puis d'Internet (RIP)

Opération d'un protocole à vecteur de distance

- La **destination** est un réseau
- La **distance** est définie selon une métrique
  - Ex : le nombre de sauts (hop count)
- La **direction** indique simplement :
  - L'adresse du prochain routeur ou
  - L'interface de sortie.

Opération d'un protocole à vecteur de distance

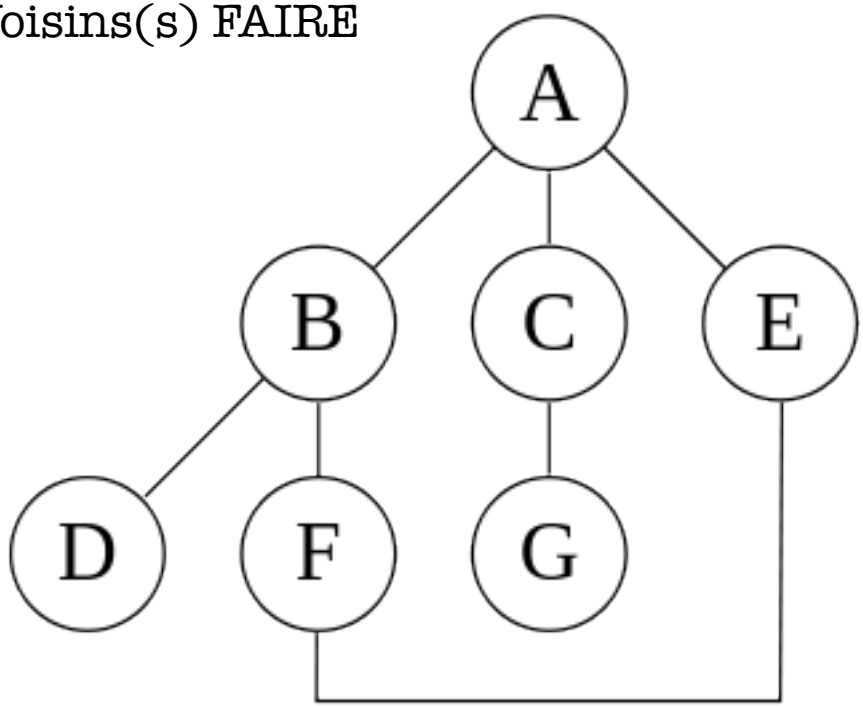
- Le protocole de routage
  - Ne connaît pas la topologie du réseau
  - La seule information qu'il détient est l'information de routage reçue de ses voisins
- Principe similaire à celui des pancartes sur une route

Origines de l'algorithme

- Algorithme de parcours de graphe DFS

```
DFS (graphe G, sommet s)
  Marquer (s);
  POUR CHAQUE élément s_fils de Voisins(s) FAIRE
    SI NonMarqué(s_fils) ALORS
      DFS (G,s_fils);
    FIN-SI
  FIN-POUR
```

- Ce parcours du graphe ne prend pas en compte le coût



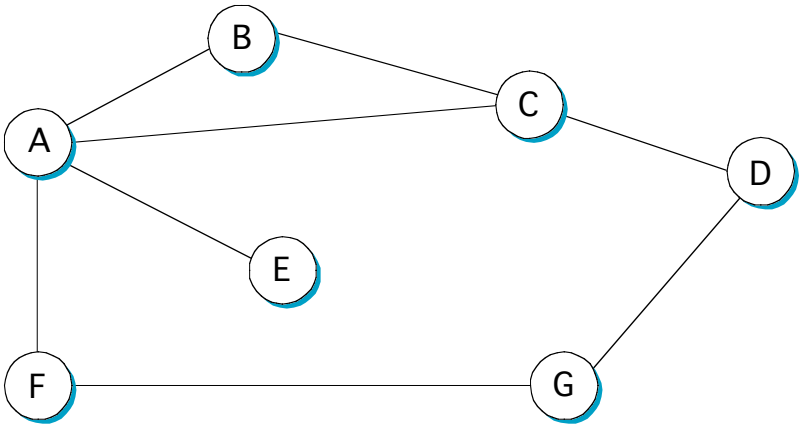
Algorithme de Bellman-Ford

- Le routeur connaît la distance à ses voisins
  - métrique = nb de saut, délais, charge...
- Échanges réguliers de vecteurs avec ses voisins
- Estimation des distances en calculant le minimum des vecteurs des voisins, additionné à la distance à ceux-ci

Pseudo-Algorithme

```
Bellman_Ford(G, s)
   initialisation (G, s)  // les poids de tous les sommets sont mis à +infini
                          // le poids du sommet initial à 0
  pour i=1 jusqu'à Nombre de sommets -1 faire
   |    pour chaque arc (u, v) du graphe faire
   |     |  paux = poids(u) + poids(arc(u, v));
   |     |  si paux < poids(v) alors
   |     |    |  pred(v) = u;
   |     |    |  poids(v) = paux;
  pour chaque arc (u, v) du graphe faire
   |   si poids(u) + poids(arc(u, v)) < poids(v) alors
   |      retourner faux  // il existe un cycle absorbant
  retourner vrai
```

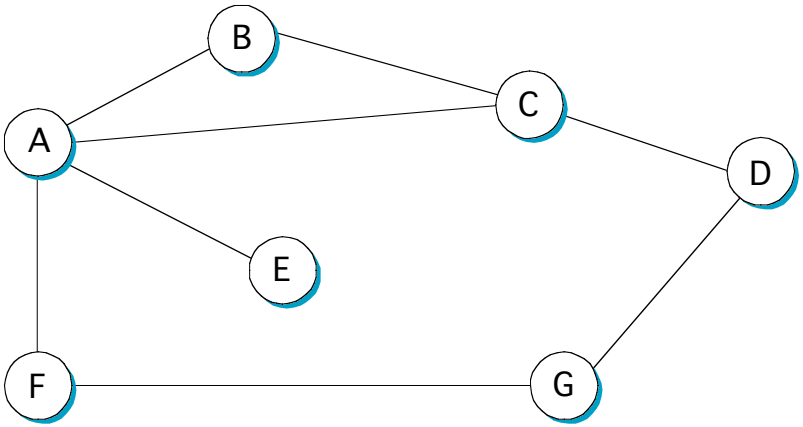
Exemple



- Information sur chaque nœud
- On considère une métrique "saut"

	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

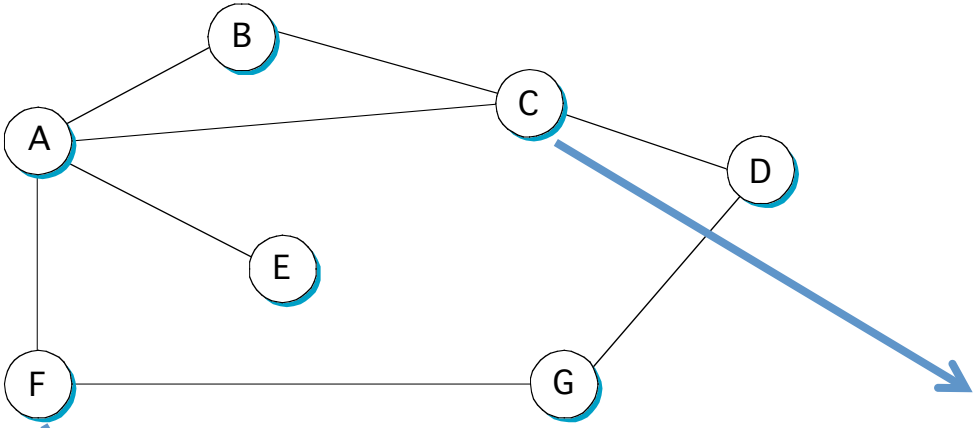
La Table de routage de A



- Avec l'information de départ, la table de routage de A est :

	Coût	Next Hop
B	1	B
C	1	C
E	1	E
F	1	F

La Table de routage de C et F

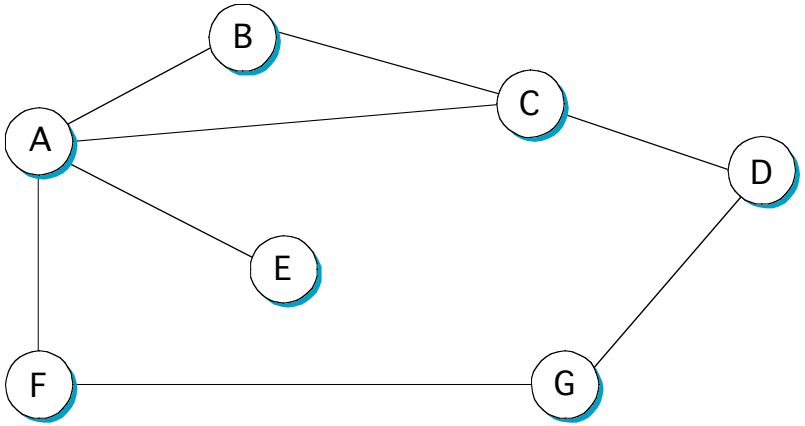


	Coût	Next Hop
A	1	A
G	1	G

	Coût	Next Hop
A	1	A
B	1	B
D	1	D

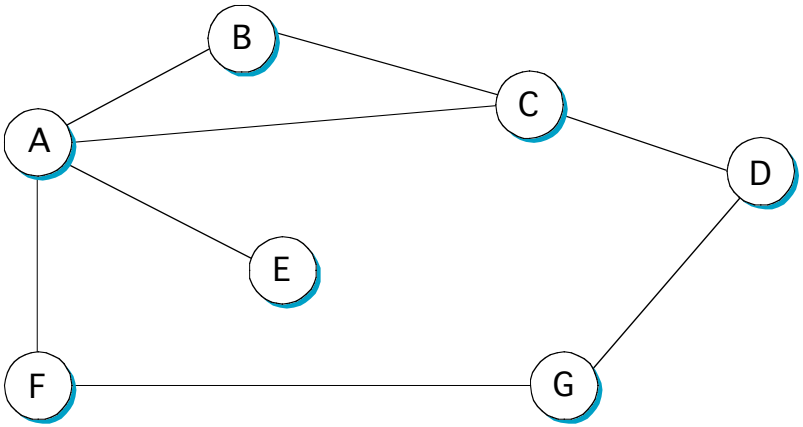
Évolution de la Table de Routage

- Chaque nœud envoie à ses voisins le contenu de sa table de routage
- Les nouvelles entrées sont mises à jour
- F → A : je connais G à une distance 1
- C → A : je connais D à une distance 1



	Coût	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Matrice de Distances Finale



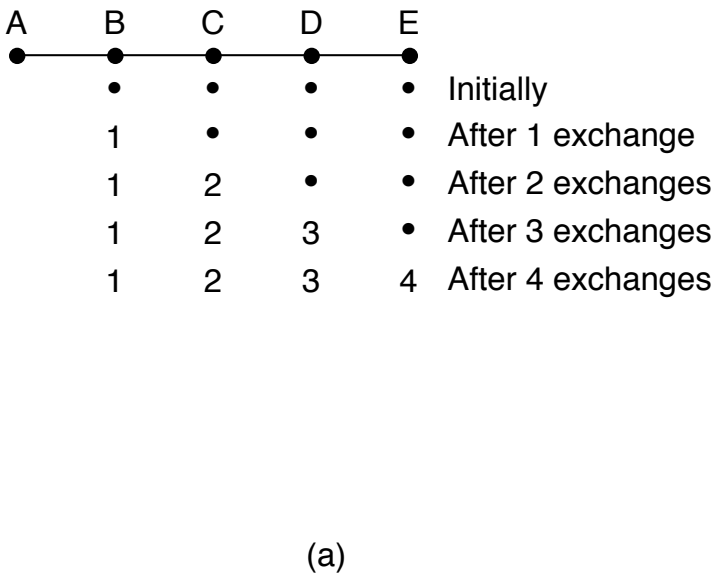
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

Attention :

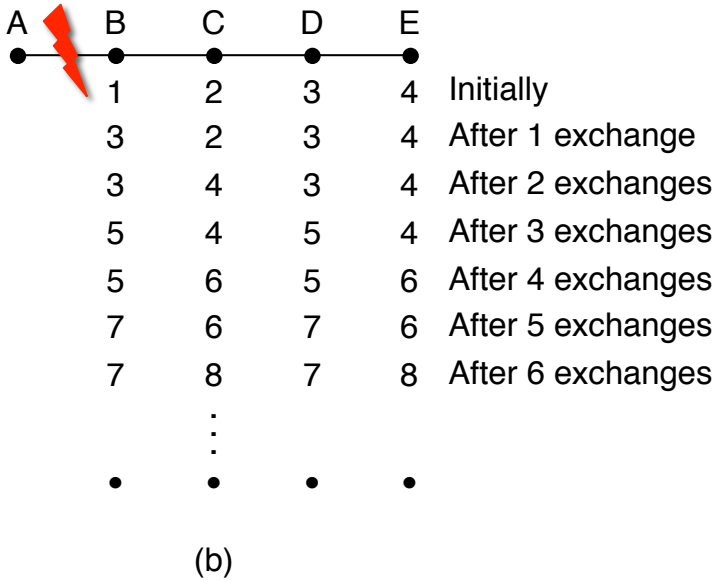
- Plusieurs "vagues" peuvent être nécessaires pour stabiliser la table de routage
- Les envois ne sont pas synchronisés

Comptage à l'Infini

- Convergence lente dans certain cas
  - Problème : réalimentation des informations
- exemple pour le nœud A :
  - (a) démarrage
  - (b) A tombe en panne -> rebond (ping-pong)



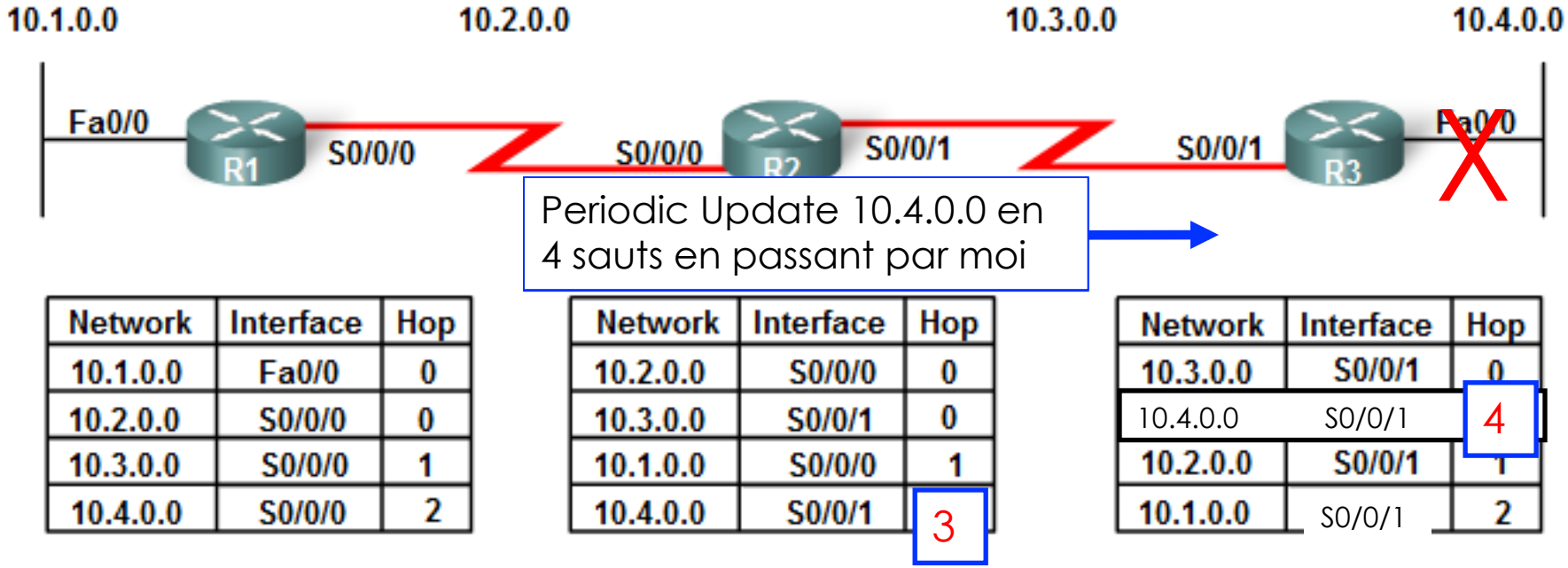
(a)



(b)

Compter à l'infini

Periodic Update 10.4.0.0 en 5 sauts en passant par moi



Network	Interface	Hop
10.1.0.0	Fa0/0	0
10.2.0.0	S0/0/0	0
10.3.0.0	S0/0/0	1
10.4.0.0	S0/0/0	2

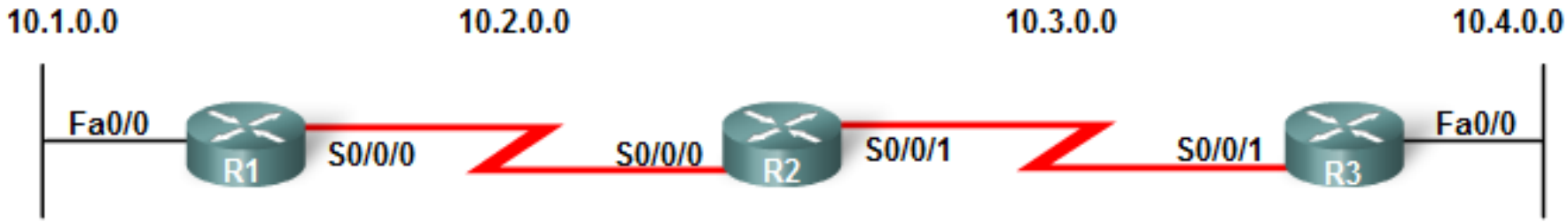
Network	Interface	Hop
10.2.0.0	S0/0/0	0
10.3.0.0	S0/0/1	0
10.1.0.0	S0/0/0	1
10.4.0.0	S0/0/1	3

Network	Interface	Hop
10.3.0.0	S0/0/1	0
10.4.0.0	S0/0/1	4
10.2.0.0	S0/0/1	1
10.1.0.0	S0/0/1	2

- Compter à l'infini est une manière de définir une limite maximale pour le nombre de sauts, évitant une boucle éternelle
  - Chaque protocole a une valeur différente
- RIP définit 16 comme "infini"
- Une route qui a compté jusqu'à l'infini est marquée comme inatteignable

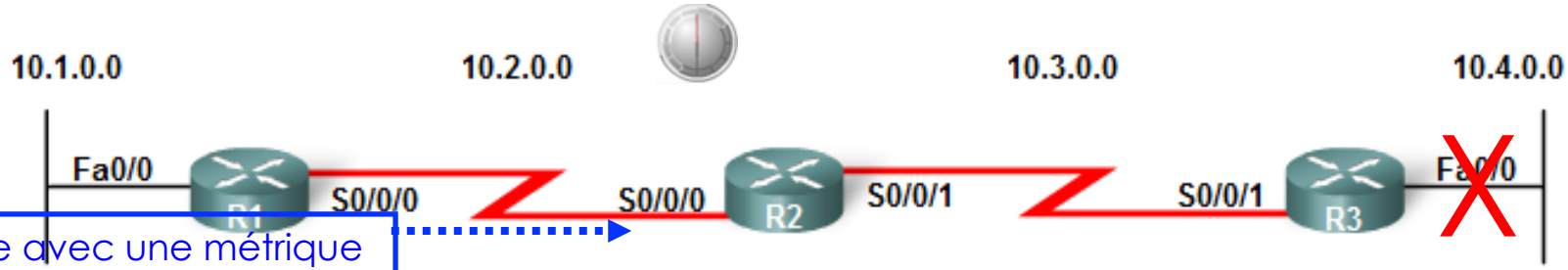
Horizon Partagé

- Technique utilisée pour éviter le comptage à l'infini
  - évite la réalimentation des informations
- Astuce : suppression des métriques provenant du destinataire dans les vecteurs lui étant destinés



Horizon partagé – split-horizon

- Dans l'exemple précédent on a vu que l'un des problèmes était la "réalimentation" des routes par des nœuds distants
- Pendant qu'on compte "à l'infini", les paquets tournent en rond entre les machines
- Il faut trouver un moyen de bloquer cette réalimentation

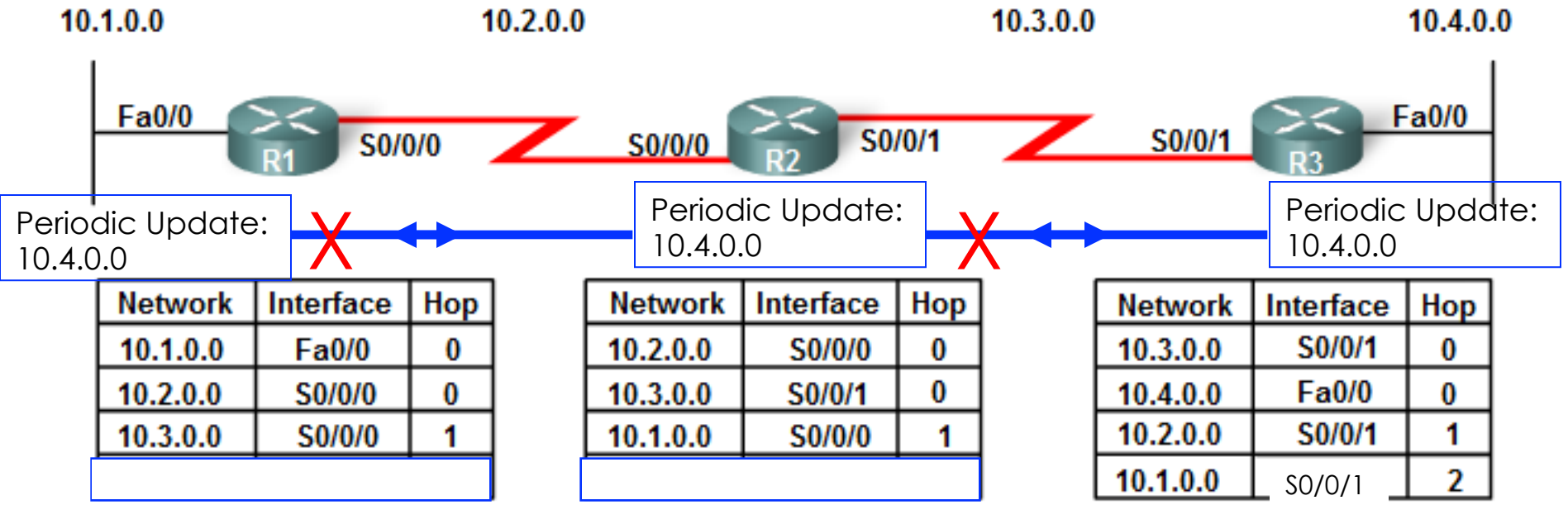


Network	Interface	Hop
10.1.0.0	Fa0/0	0
10.2.0.0	S0/0/0	0
10.3.0.0	S0/0/0	1
10.4.0.0	S0/0/0	2

Network	Interface	Hop
10.2.0.0	S0/0/0	0
10.3.0.0	S0/0/1	0
10.1.0.0	S0/0/0	1
10.4.0.0	S0/0/1	1

Network	Interface	Hop
10.3.0.0	S0/0/1	0
~~10.4.0.0~~	~~Fa0/0~~	~~0~~
10.2.0.0	S0/0/1	1
10.1.0.0	S0/0/1	2

Split Horizon



Network	Interface	Hop
10.1.0.0	Fa0/0	0
10.2.0.0	S0/0/0	0
10.3.0.0	S0/0/0	1

Network	Interface	Hop
10.2.0.0	S0/0/0	0
10.3.0.0	S0/0/1	0
10.1.0.0	S0/0/0	1

Network	Interface	Hop
10.3.0.0	S0/0/1	0
10.4.0.0	Fa0/0	0
10.2.0.0	S0/0/1	1
10.1.0.0	S0/0/1	2

1. R3 annonce le réseau 10.4.0.0 à R2

2. R2 reçoit l'information et met à jour sa table de routage

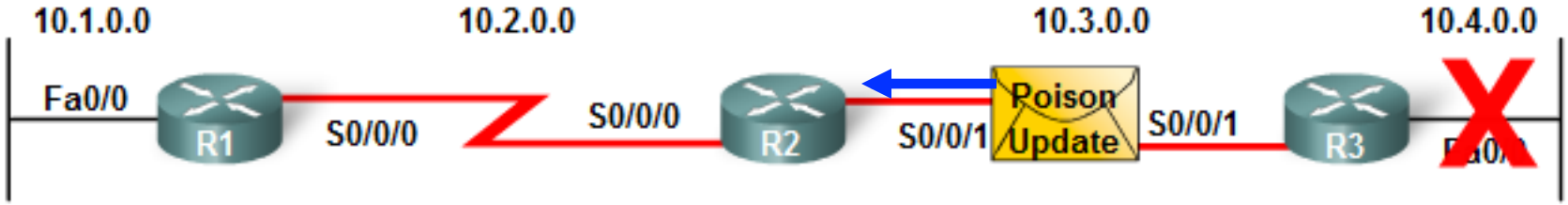
3. R2 annonce 10.4.0.0 à R1 via S0/0/0

- R2 n'annonce pas 10.4.0.0 à R3 via S0/0/1, car cette route est issue de cette interface

4. R1 reçoit l'information et met à jour sa table de routage

5. À cause du split horizon, R1 n'annonce pas 10.4.0.0 sur R2 non plus

Route empoisonnée



Network	Interface	Hop
10.1.0.0	Fa0/0	0
10.2.0.0	S0/0/0	0
10.3.0.0	S0/0/0	1
10.4.0.0	S0/0/0	2

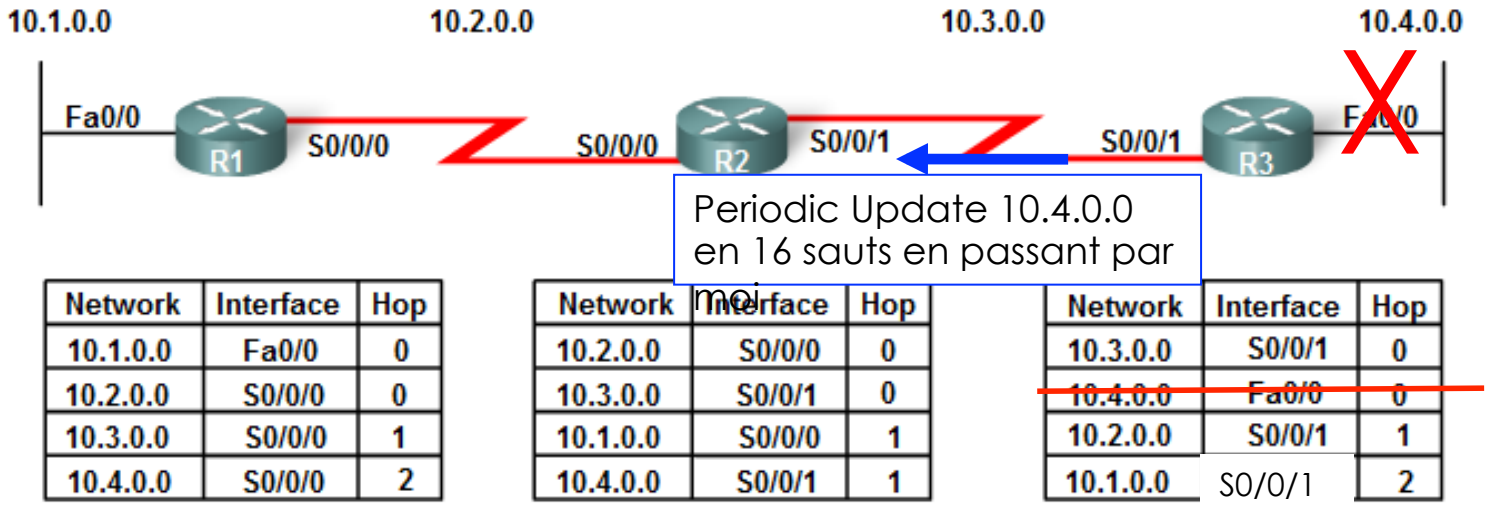
Network	Interface	Hop
10.2.0.0	S0/0/0	0
10.3.0.0	S0/0/1	0
10.1.0.0	S0/0/0	1
10.4.0.0	S0/0/1	1

Network	Interface	Hop
10.3.0.0	S0/0/1	0
10.4.0.0	Fa0/0	16
10.2.0.0	S0/0/1	1
10.1.0.0	S0/0/1	2

- ***C'est une technique pour marquer une route comme inatteignable*** lors d'une mise à jour envoyée aux autres machines.
  - **Inatteignable** = métrique au maximum
  - Les routes empoisonnées accélèrent la convergence
  - Risque : si un update se perd, on peut trouver des boucles de routage dans d'autres parties du réseau

Split-horizon avec retour empoisonné

- Les deux techniques précédentes travaillent dans des directions différentes
  - Split-horizon : empêche le retour "→"
  - Route empoisonné : annonce des routes infinies "←"
- Si les deux techniques ne sont pas bien accordées, des boucles de routage peuvent toujours avoir lieu
  - Update empoisonné perdu, split-horizon en marche → comptage à l'infini
- Une manière d'empêcher ceci est de "mélanger" les deux approches



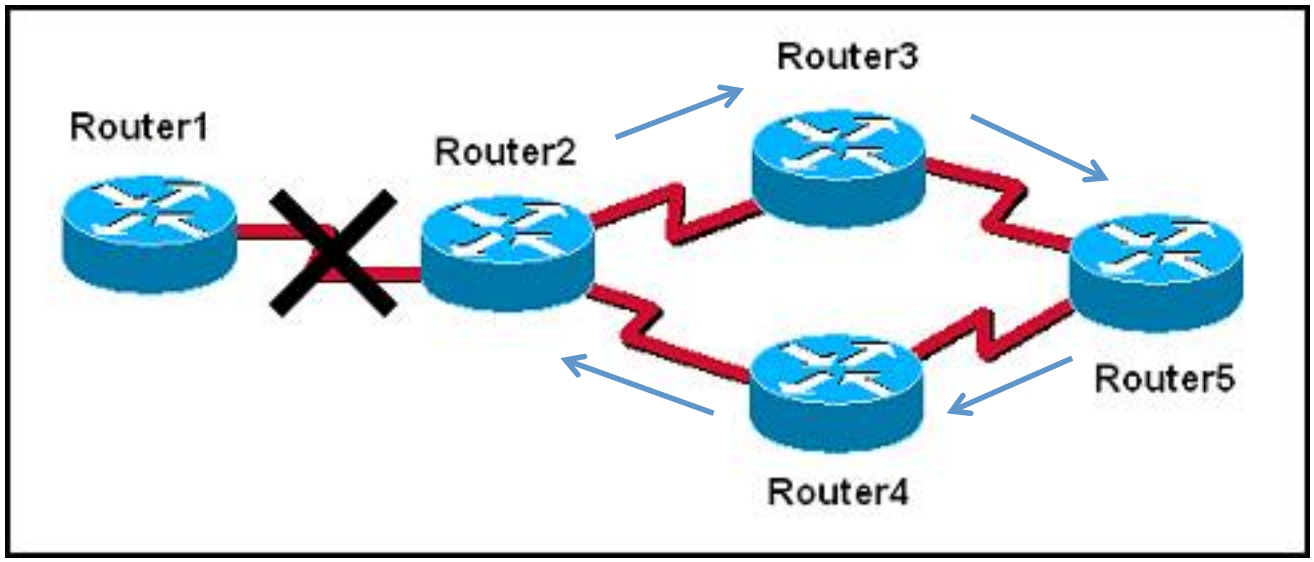
Network	Interface	Hop
10.1.0.0	Fa0/0	0
10.2.0.0	S0/0/0	0
10.3.0.0	S0/0/0	1
10.4.0.0	S0/0/0	2

Network	Interface	Hop
10.2.0.0	S0/0/0	0
10.3.0.0	S0/0/1	0
10.1.0.0	S0/0/0	1
10.4.0.0	S0/0/1	1

Network	Interface	Hop
10.3.0.0	S0/0/1	0
~~10.4.0.0~~	~~Fa0/0~~	~~0~~
10.2.0.0	S0/0/1	1
10.1.0.0	S0/0/1	2

Est-ce fini ?

- Est-ce que le problème des boucles de routage est fini avec l'utilisation de l'horizon partagé avec retour empoisonné ?
- **NON**
- Ces techniques n'empêchent que les boucles entre deux nœuds
- Cet autre scénario n'est pas concerné :

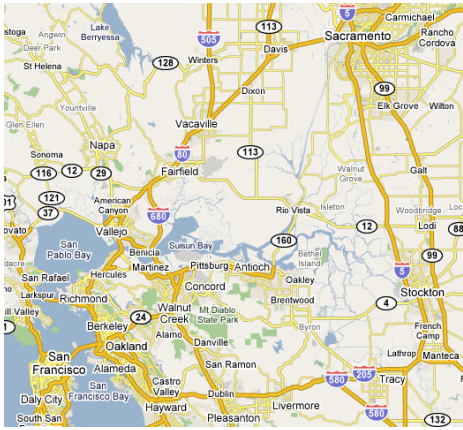


Plan Couche Réseau

- Routage
  - Principes de base
  - Inondation
  - Routage par vecteur de distance
  - **Routage par état des liaisons**
  - Routage hiérarchique
  - Routage multicast
  - Routage dans les réseaux mobile, ad-hoc et P2P

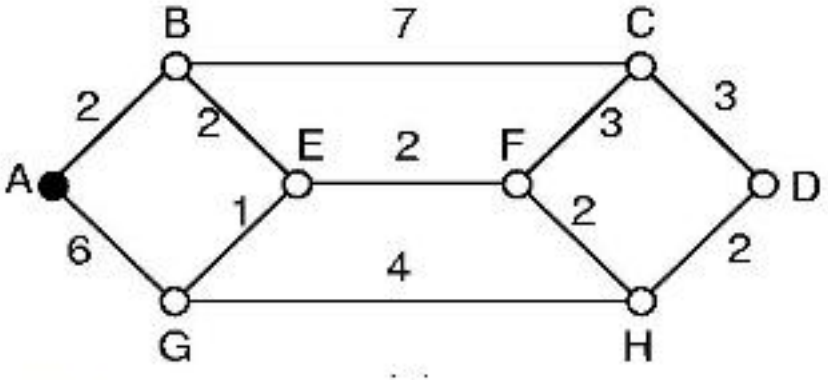
Link State Routing

- Algorithme de routage dynamique
- Fonctionnalités (par chaque routeur) :
  - découverte de ses voisins et de leurs adresses
  - mesure et calcul de coût vers ses voisins
  - construction de paquets avec les informations connues et envoi de celui-ci vers tous les routeurs
  - calcul du plus court chemin vers tous les routeurs
- Représentation complète du réseau dans tous les routeurs
  - calcul du plus court chemin avec SPF de Dijkstra



SPF - Plus Court Chemin

- Représentation à l'aide d'un graphe :
  - routeur = nœud, sommet
  - liaisons = arcs (valeur = métrique)

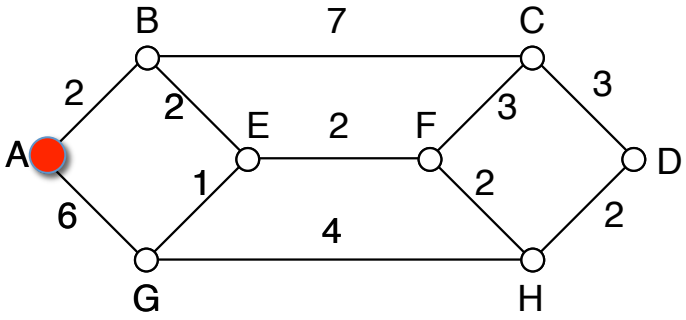


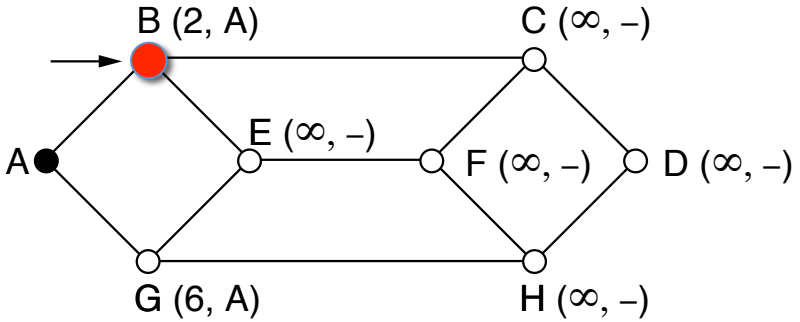
Plus Court Chemin

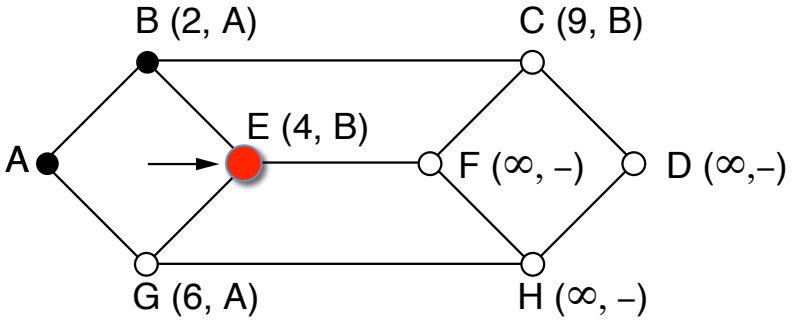
- Implémentation
  - une étiquette pour chaque nœud, contenant la distance minimum ainsi que le nœud précédent
  - initialement le meilleur chemin n'est pas connu, la source (destination) est instaurée comme nœud initial
  - mise a jour progressive des étiquettes des nœuds à partir de la route à la distance minimum

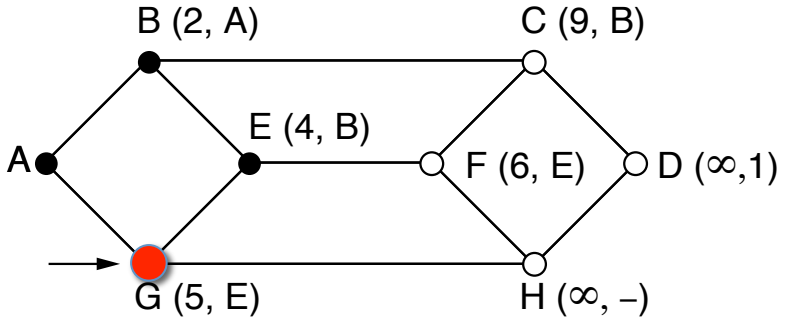
Algorithme SPF de Dijkstra

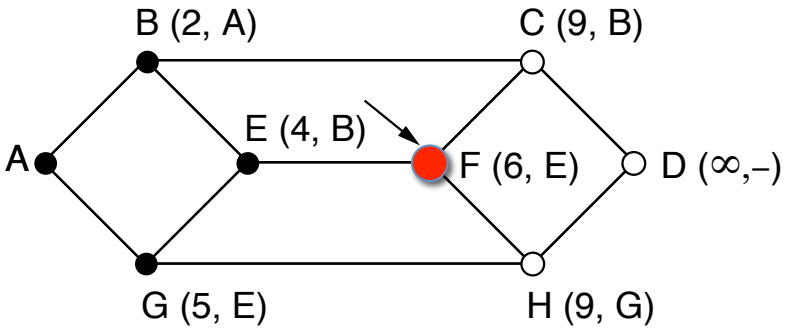
- pour chaque sommet *s* du graphe faire :
  - d[*s*] = inf., pred[*s*] = NIL
- d[*s-init*] = 0, E = vide, R = tous les sommet
- tant que R n'est pas vide, faire :
  - soit *m* le sommet de R tq d[*m*] = min des d[*s*] de R
  - retirer *m* de R et l'inclure dans E
  - pour chaque sommet adjacent de *m* faire :
    - si d[*s*] > d[*m*] + métrique[*s,m*] alors
      - d[*s*] = d[*m*] + métrique[*s,m*]
      - pred[*s*] = *m*

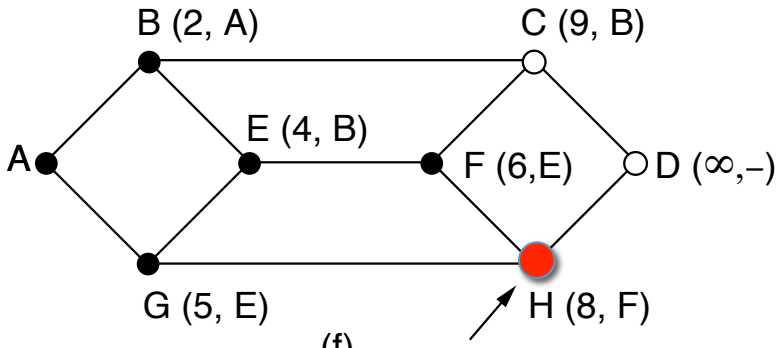

B
7
C
2
2
E
2
F
3
3
A
1
2
D
6
4
2
G
H
(a)


B (2, A)
C (∞, –)
E (∞, –)
A
F (∞, –)
D (∞, –)
G (6, A)
H (∞, –)
(b)


B (2, A)
C (9, B)
E (4, B)
A
F (∞, –)
D (∞,–)
G (6, A)
H (∞, –)
(c)


B (2, A)
C (9, B)
E (4, B)
A
F (6, E)
D (∞,1)
G (5, E)
H (∞, –)
(d)


B (2, A)
C (9, B)
E (4, B)
A
F (6, E)
D (∞,–)
G (5, E)
H (9, G)
(e)

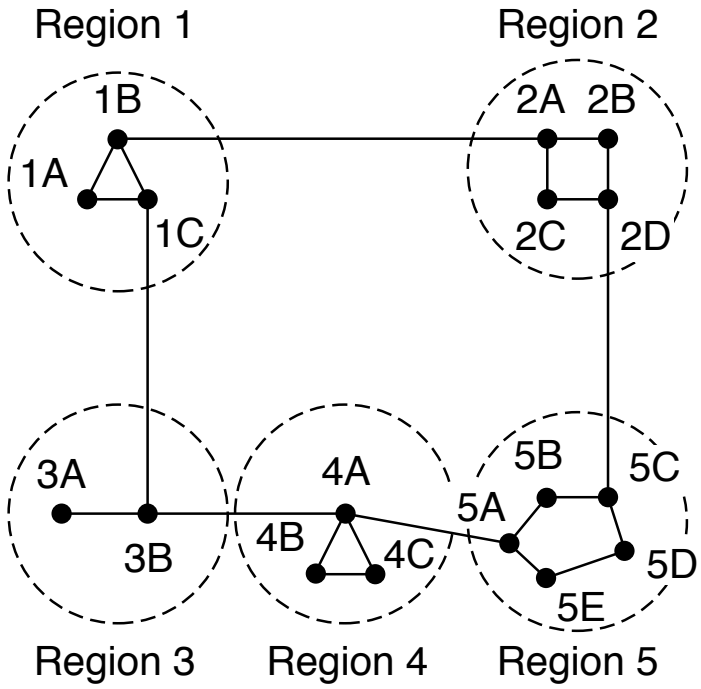

B (2, A)
C (9, B)
E (4, B)
A
F (6,E)
D (∞,–)
G (5, E)
H (8, F)
(f)

Plan Couche Réseau

- Routage
  - Principes de base
  - Inondation
  - Routage par vecteur de distance
  - Routage par état des liaisons
  - **Routage hiérarchique**
  - Routage multicast
  - Routage dans les réseaux mobile, ad-hoc et P2P

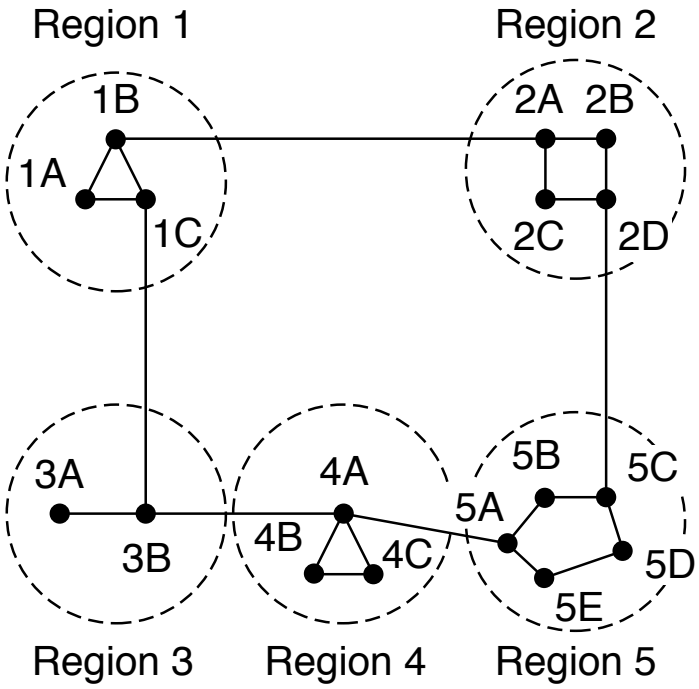
Routage Hiérarchique

- Le routage dynamique est coûteux
  - Mémoire
  - Échange de messages
- La sous-division du réseau en zones permet une vue hiérarchique du réseau
  - Solution actuelle dans Internet



Routage Hiérarchique

- Agrégation des routes d'un domaine
- Simplification des choix
  - Solutions sous-optimales



Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Plan Couche Réseau

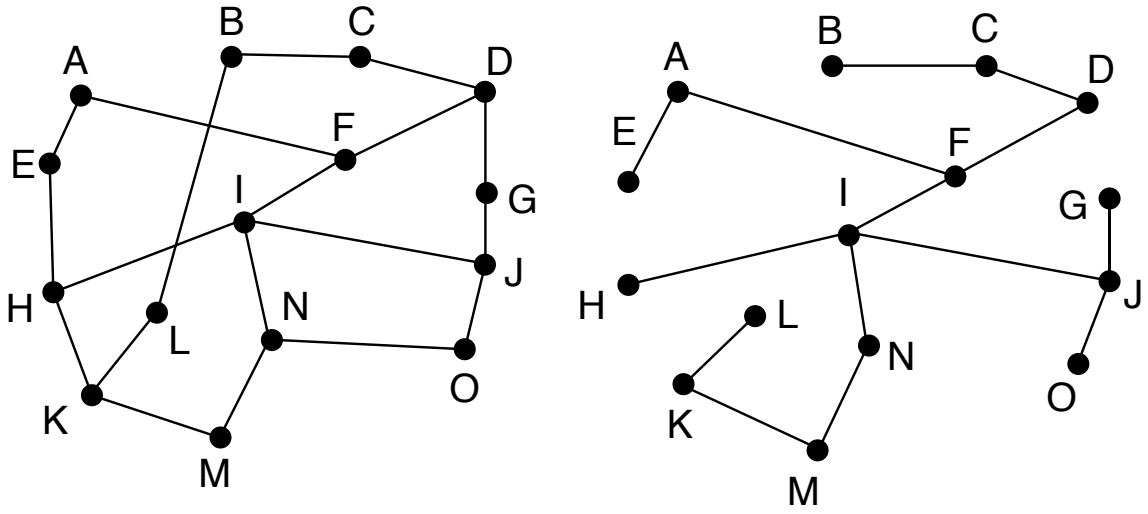
- Routage
  - Principes de base
  - Inondation
  - Routage par vecteur de distance
  - Routage par état des liaisons
  - Routage hiérarchique
  - **Routage multicast**
  - Routage dans les réseaux mobile, ad-hoc et P2P

Routage Broadcast

- Envoi multiple
  - inefficace et nécessite la liste des destinataires
- *Flooding*
  - renvoi systématique des messages
  - trop cher
  - reste la référence par rapport au meilleur chemin
- Routage multi-destination
  - efficace mais avec une liste explicite des destinataires

Arbre Couvrant

- Spanning-tree – un seul chemin (le plus court) vers chaque nœud du réseau
  - Utilisation de l'algorithme SPF
    - Efficace, sans boucles, mais mémorisation obligatoire

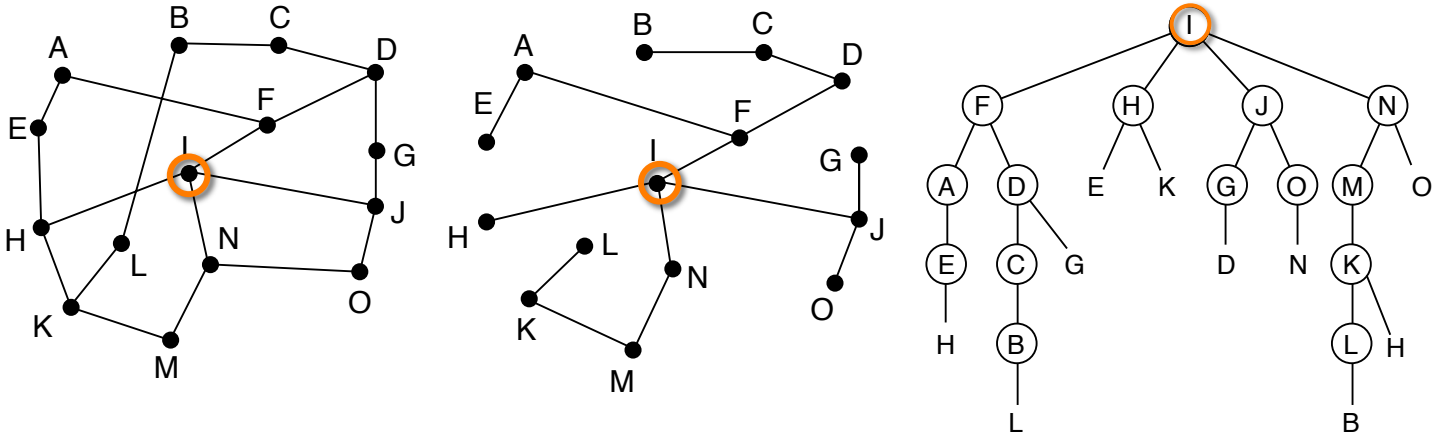


Reverse Path Forwarding

- Alternative "dynamique" au spanning-tree
- Principe : utiliser le routage unicast comme référence (flooding filtré)
- Permet la construction **d'arbres dynamiques** proches de celles du Spanning-tree
  - Arbre couvrant sans mémoire

Reverse Path Forwarding

- Algorithme :
  - lorsqu'un paquet diffusé arrive dans un routeur, vérification qu'il arrive par l'interface utilisée pour joindre la source de celui-ci
    - si oui, alors probablement sur l'arbre couvrant et diffusion vers les interfaces de sorties
    - sinon élimination (paquet dupliqué)

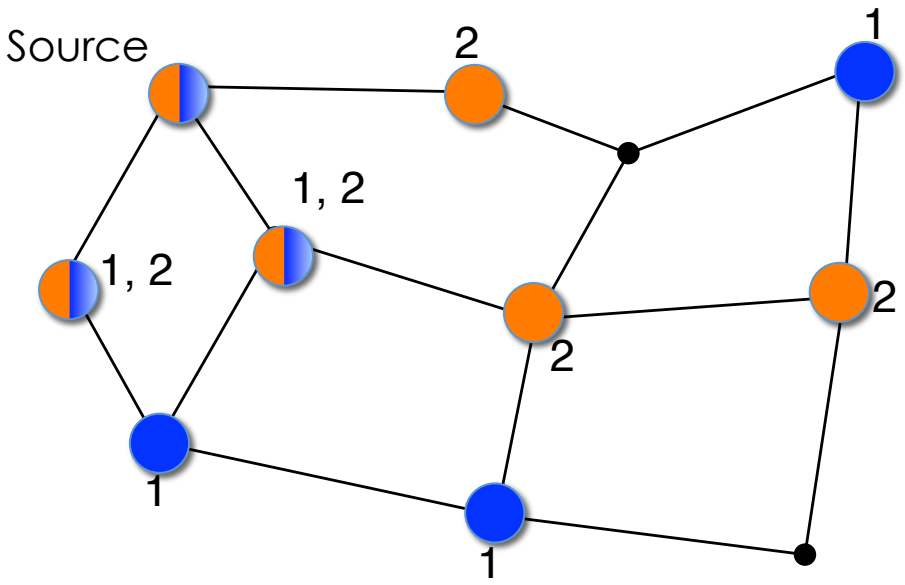


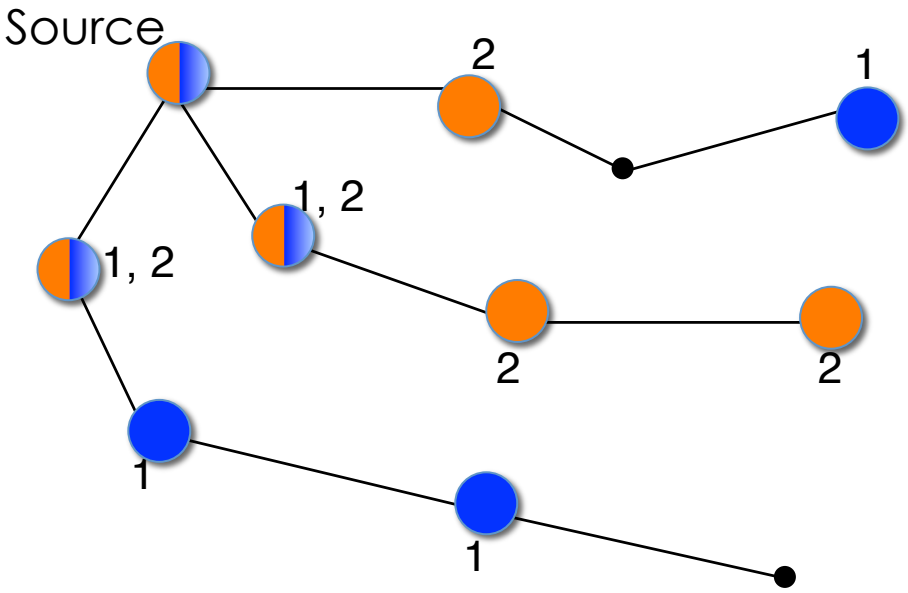
Routage Multicast

- Problème : seul un groupe de machines n'est intéressé par le flux
  - Comment limiter la diffusion ?
  - Comment optimiser les flux pour différents groupes ?
- Outils nécessaires
  - Un système de gestion de groupes
    - Création, suppression, join/leave
    - Des nœuds qui avertissent leurs groupes aux routeurs
    - Des arbres couvrants (ou RPF) avec élagage

Routage Multicast

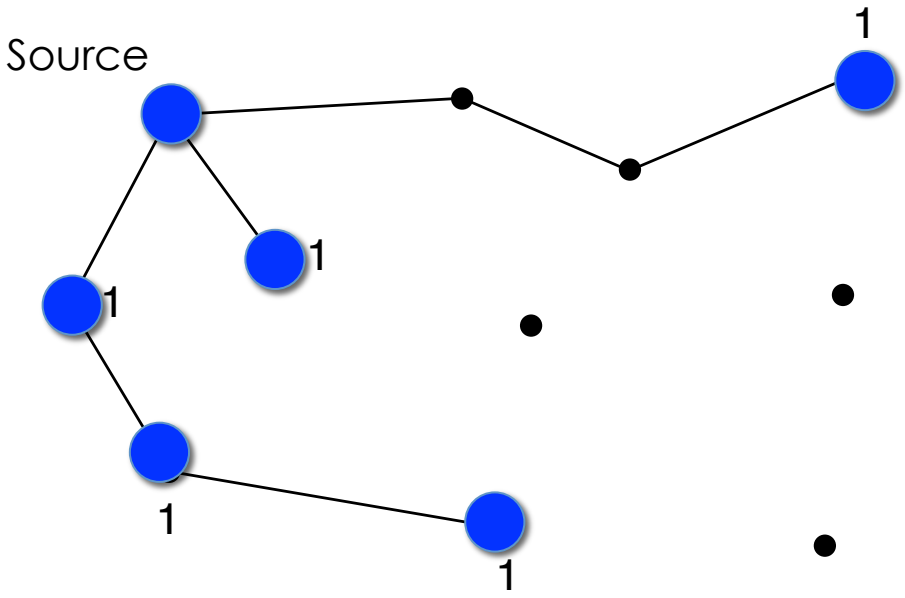
- Exemple : deux groupes
  - Construction des arbres couvrants
    - Algorithme RPF, par exemple

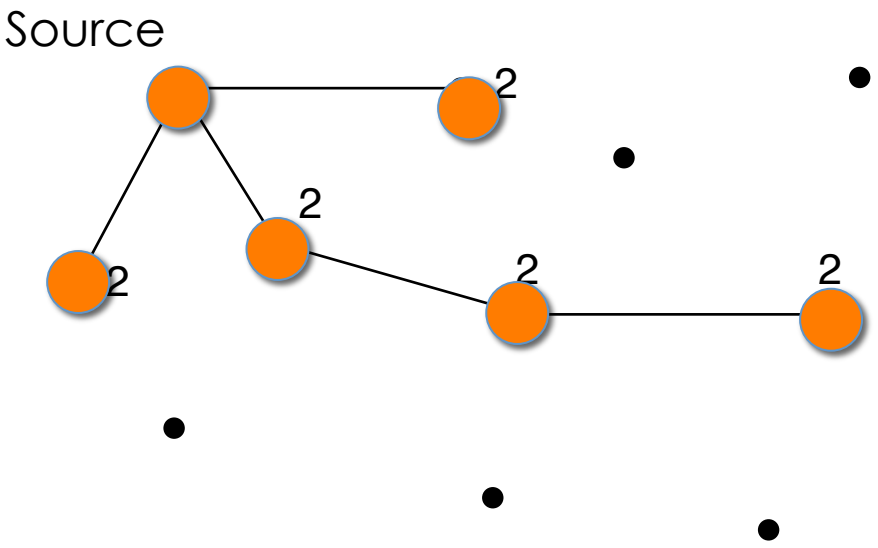




Routage Multicast

- Exemple : deux groupes
  - Élagage des arbres
    - Messages "PRUNE"





Routage Multicast

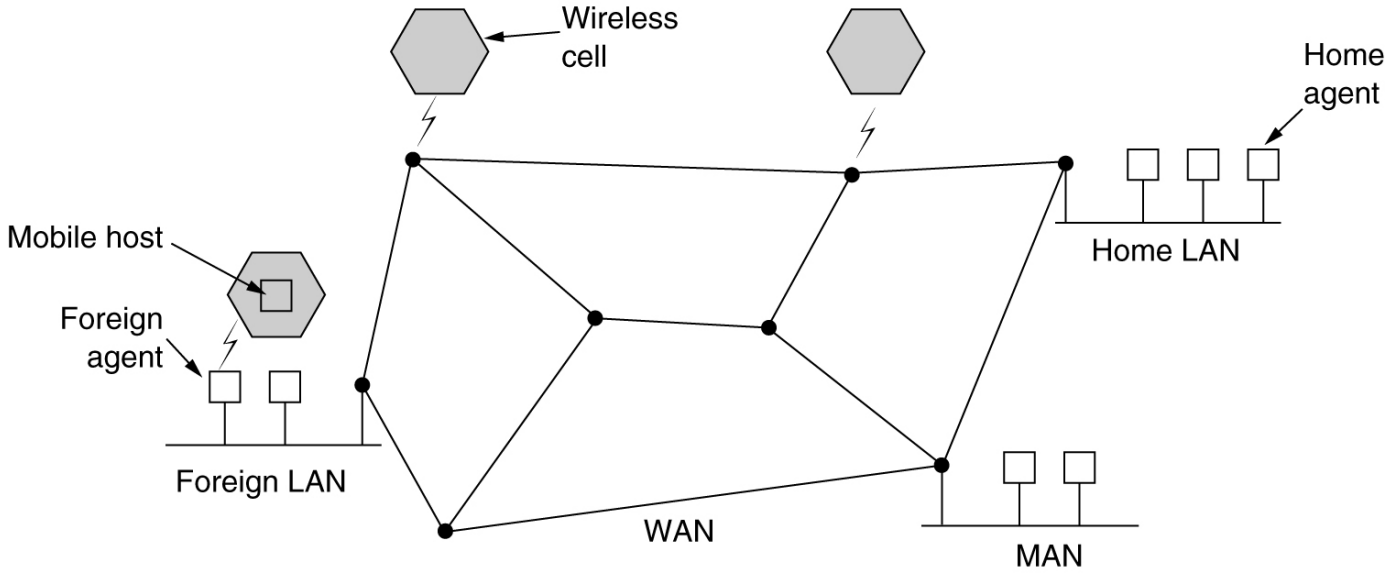
- Inconvénients :
  - Ne s'adapte pas très bien aux grands réseaux
    - Un arbre à mémoriser par groupe et par source !!!
- Alternative
  - *Core-base tree* (aussi connu comme *Rendez-vous Point*)
  - Un nœud "central" à un groupe est choisi ; il devient le point de départ des multicasts
    - Ce n'est pas optimal, mais réduit le nombre d'arbres à stocker

Plan Couche Réseau

- Routage
  - Principes de base
  - Inondation
  - Routage par vecteur de distance
  - Routage par état des liaisons
  - Routage hiérarchique
  - Routage multicast
  - **Routage dans les réseaux mobile, ad-hoc et P2P**

Réseaux mobiles

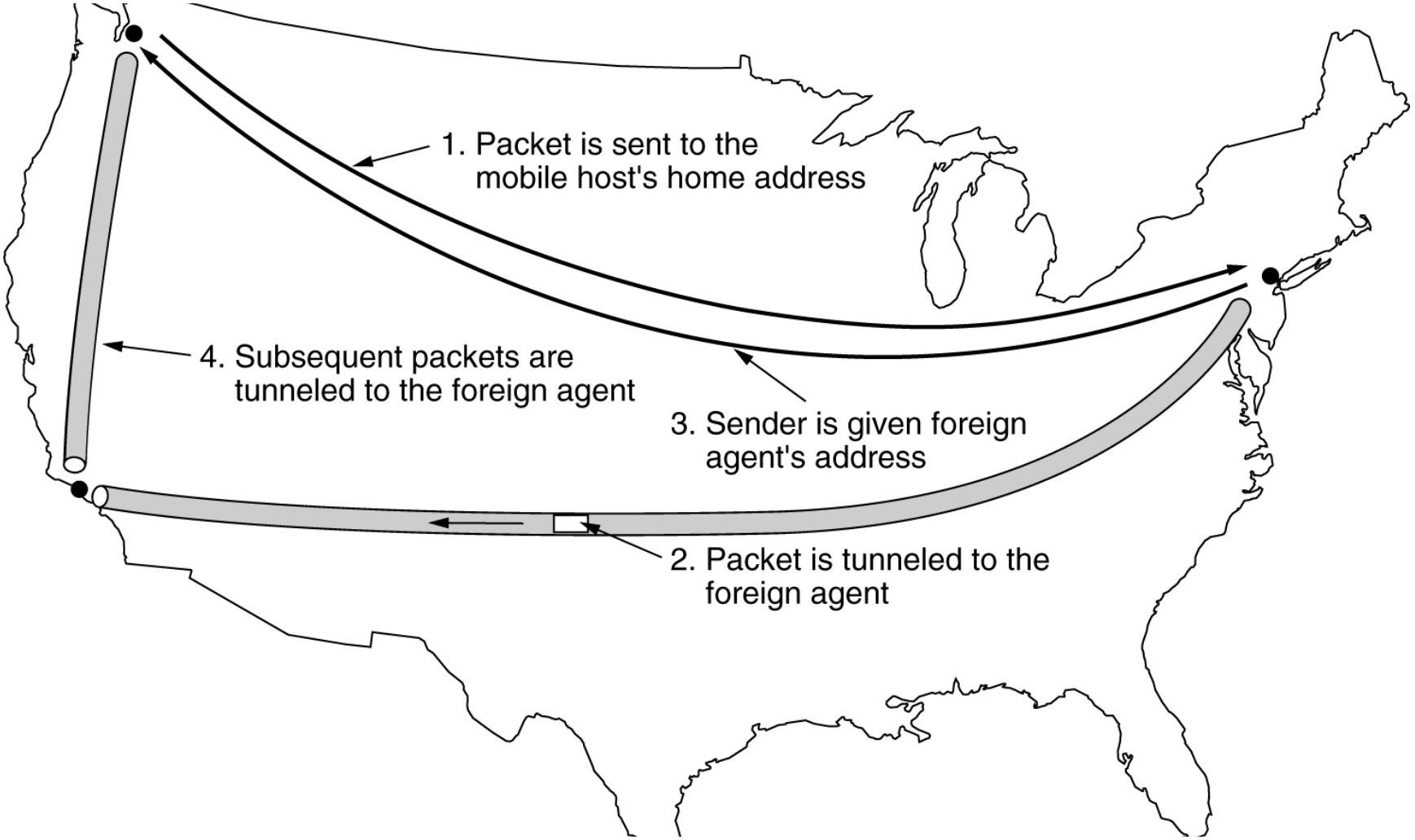
- Difficulté : le client se déplace
  - Il faut garder un "lien" avec son identifiant d'origine
    - Numéro de téléphone
    - Numéro IP (pas de DHCP)



Réseaux mobiles

- Solution : utilisation d'agents relai
  - Agents de domiciliation
    - Service qui garde une trace de la localisation de ses "clients" et redirige le trafic en cas de déplacement
  - Agents extérieurs
    - Service qui gère les clients entrant dans une zone extérieur
      - Met à jour les informations sur l'agent de domiciliation
      - Transmet les données au client mobile

Réseaux mobiles



Réseaux ad-hoc

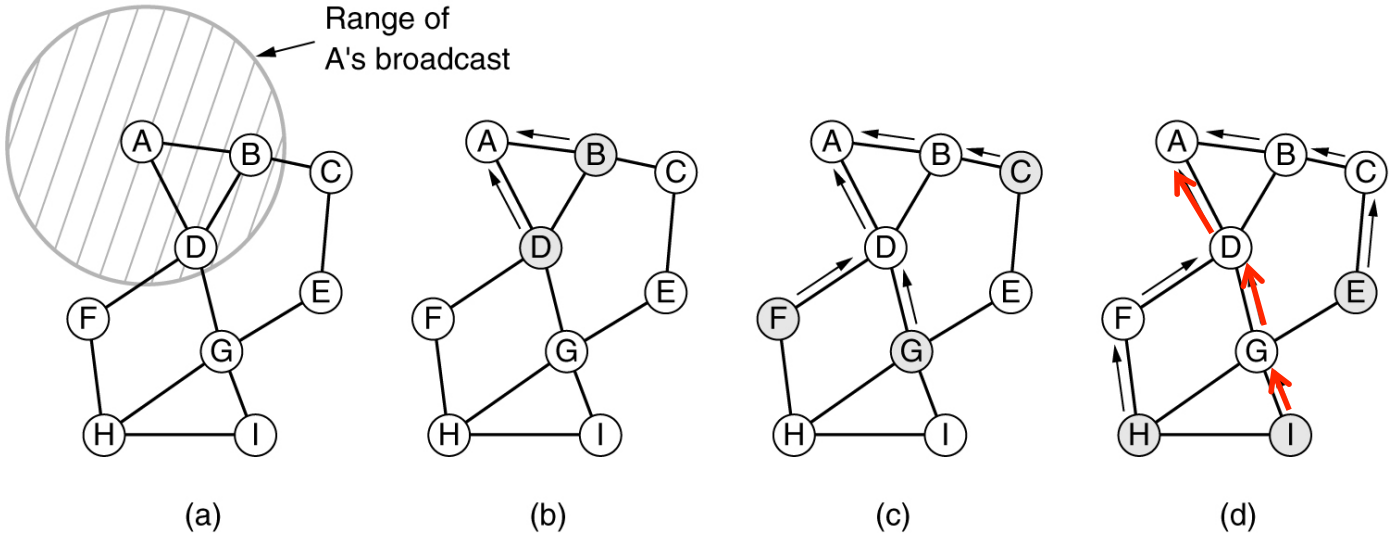
- Ici, tout le réseau est "mobile"
  - Aucun agent de référence (routeur) auquel se fixer
  - Les machines n'ont qu'une visibilité réduite de leur voisinage (portée de diffusion)
  - Les dispositifs ad-hoc peuvent avoir des limitations de baterie
- Le routage dans les réseaux ad-hoc se fait à la demande, mais avec des algorithmes similaires à ceux déjà vus

Réseaux ad-hoc : protocole AODV

- AODV – Ad-hoc On-demand Distance Vector
  - Protocole inspiré de Bellman-Ford (et DFS)
- Caractéristiques
  - Route vers un nœud distant grâce à une recherche de type vecteur de distance
    - "destination, passerelle, distance"
  - Surveillance du voisinage grâce à des "hello"
  - "Retour empoisonné" pour les routes qui disparaissent

Protocole AODV

- Inondation du réseau par vagues
  - Message de type "route request"/"route reply"
  - Utilisation de "timestamps" pour éviter les doublons

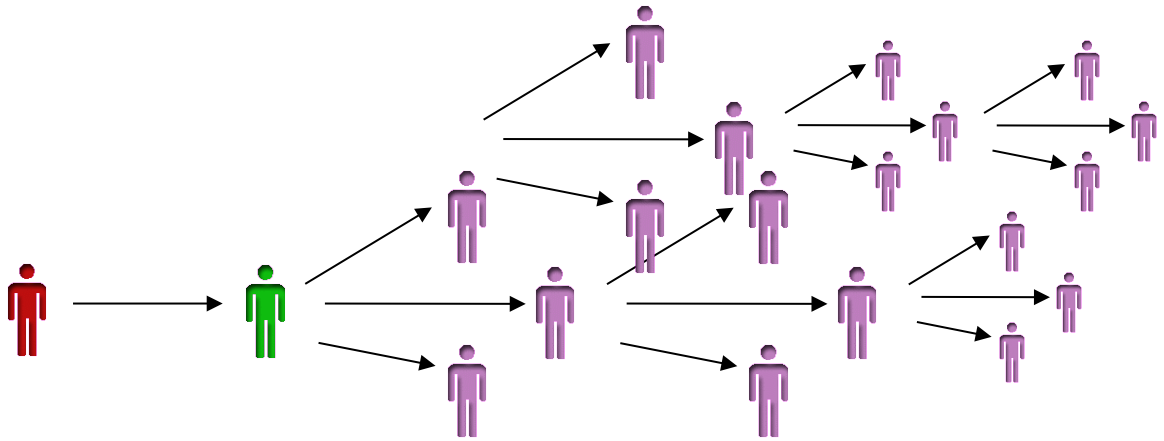


Routage dans les réseaux P2P

- Les réseaux P2P peuvent être "purs" ou "hybrides"
  - Hybrides : un élément central (serveur) connaît les adresses des différents éléments
  - Pur : aucun élément centralisé n'existe
    - Non-structuré
    - Structuré

Routage P2P

- P2P non-structuré
  - La recherche se fait par inondation
  - Ex : gnutella



	T=1	T=2	T=3	T=4	T=5	T=6	T=7
N=2	2	4	6	8	10	12	14
N=3	3	9	31	45	93	189	381
N=4	4	16	52	160	484	1456	4372
N=5	5	25	105	425	1705	6825	27305
N=6	6	36	186	936	4686	23436	117186
N=7	7	49	301	1813	10885	65317	391909
N=8	8	64	456	3200	22408	156864	**1098056**

Systèmes Structurés

- Principe :
  - Totalement distribué mais suffisamment structuré pour éviter le flooding
  - les index sont décentralisés (DHT)

DHT (Distributed Hash Table)

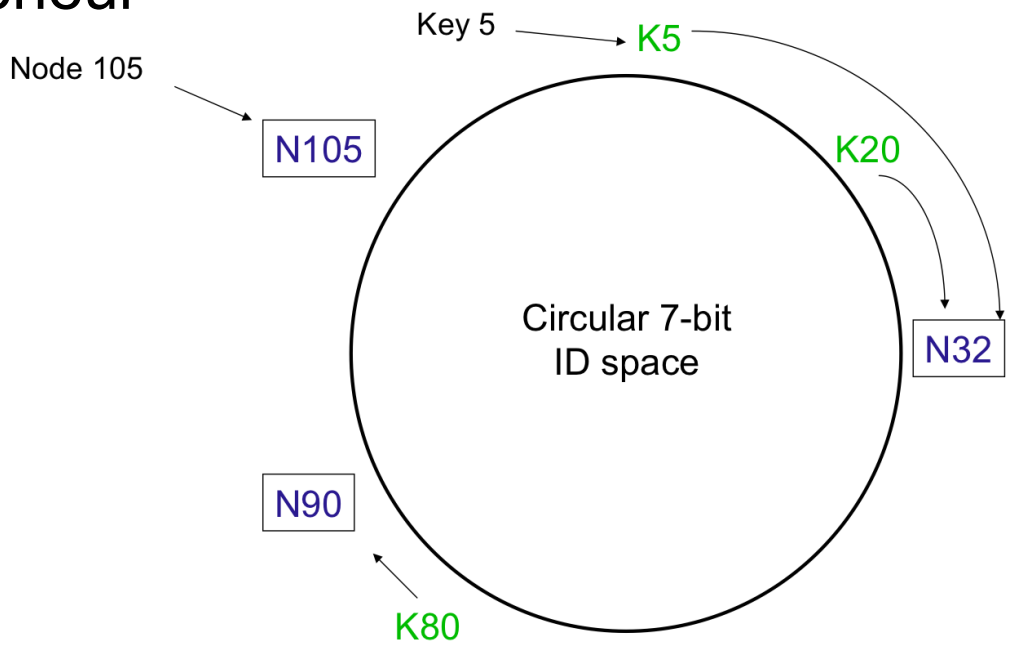
- une fonction de hachage permet de contacter le peer qui a la plus grande probabilité de connaître le index pour les informations demandées
- Exemples
  - CAN (Content Addressable Network)
  - Chord
  - Pastry

Hachage des Index Décentralisé (DHT)

- Le système Chord
  - Une fonction de hachage est appliquée sur les IPs (**identifiant**) et le nom des ressources (**clé**), résultant en une chaîne de m bits
    - Habituellement m=160 → 2^{160} identificateurs
  - Les pairs **(nom)** sont stockés de façon distribué
  - Une fonction ***successeur(k)*** permet de trouver le premier nœuds réel ≥ k module 2^m

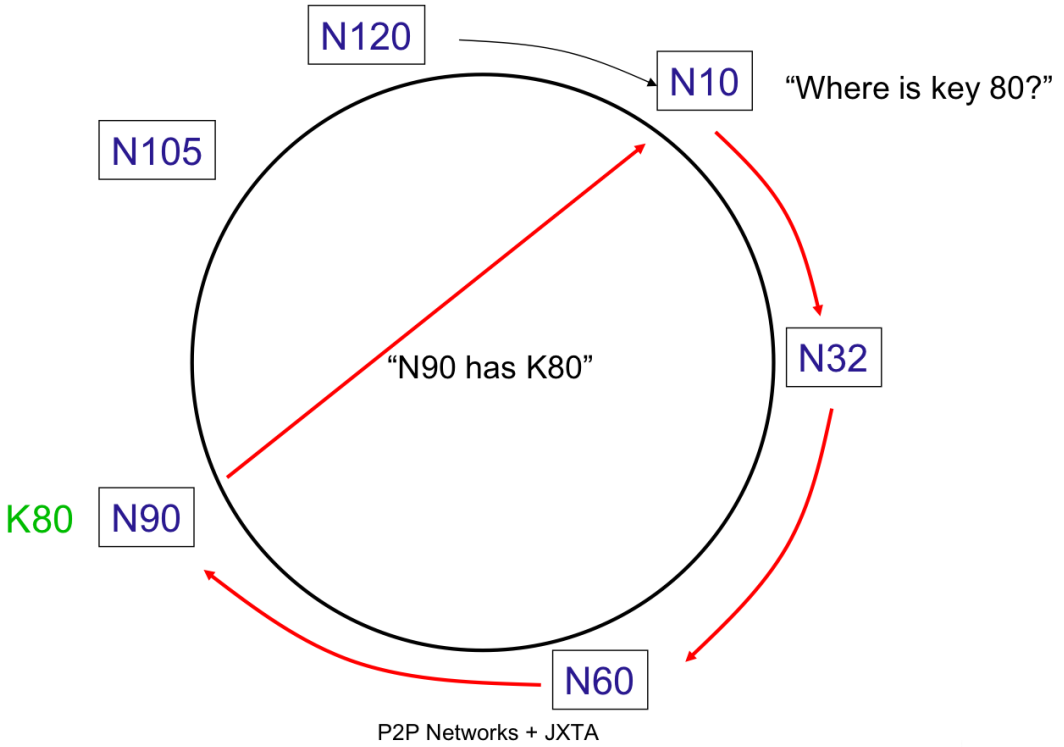
Hachage des Index Décentralisé (DHT)

- Chord
  - les nœuds et les clés partagent un espace d'identification
  - les clés sont stockés dans les nœuds avec l'ID immédiatement supérieur



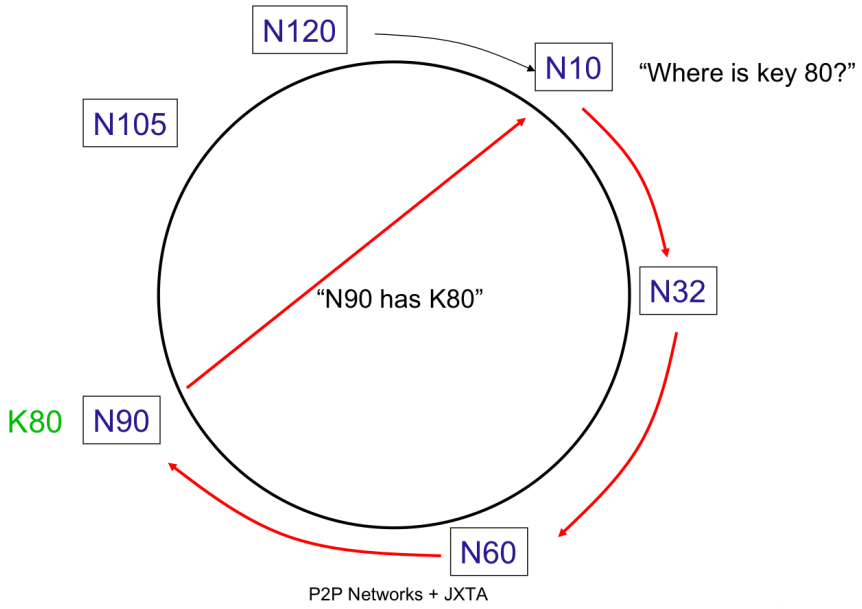
Hachage des Index Décentralisé (DHT)

- Chord
  - lors d'une requête, la demande est relayé dans l'ordre croissante des nœuds jusqu'à trouver le nœud avec l'index demandé



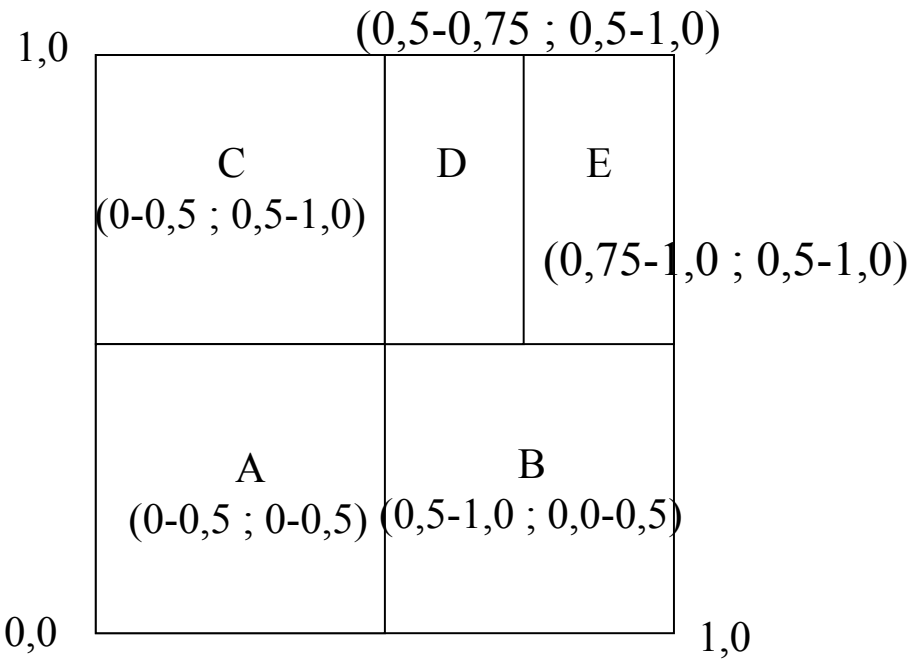
Hachage des Index Décentralisé (DHT)

- Chord – optimisation – **finger table**
  - Chaque nœud garde un tableau de *m* entrées. L'entrée *i* du nœud *n* contient l'adresse du successeur $(n + 2^i)$.
  - La recherche d'un nœud est réduite à *O(log N)*



Hachage des Index Décentralisé (DHT)

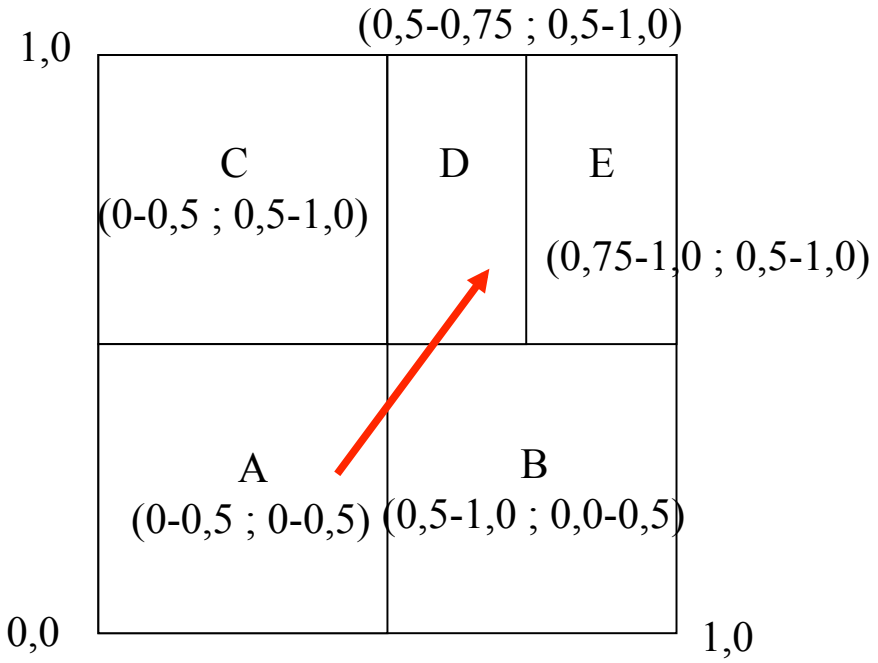
- **CAN (Content Addressable Network)**
- Tore cartésien virtuel
  - Dimension d
- Espace partagé en zones
  - Un nœud responsable par zone
- Association clé-point déterministe
  - Fonction de hachage
  - K => Point P dans l'espace des coordonnées
  - P => Zone Z => Responsable R
  - La paire (clé, valeur) est stockée sur R



Hachage des Index Décentralisé (DHT)

- **Routage CAN**
- Chaque nœud maintient une table
  - Adresses IP de ses voisins
  - Zones de coordonnées associées
- A est voisin de B
- B est voisin de D
  - mais A n'est pas voisin de D

- **Routage glouton**
  - Dans la topologie à d dimensions
- Distance moyenne = O ($n^{1/d}$)
  - (pour n zones)



Petite note sur le routage avec des Circuits Virtuels

- Les protocoles étudiés ici sont pour le routage de datagrammes
  - Chaque paquet est routé individuellement
- Dans un système à circuits virtuels, le routage se fait par "session"
  - Tous les paquets d'une session suivent la même route établie lors de la création du CV
- Ce principe est partiellement repris par IP lors de l'usage de la technologie MPLS
  - But : accélérer le routage et séparer les flux

En résumé

- Le routage est l'un des points clés d'un réseau "large"
- Différentes contraintes et techniques
  - Réseaux classiques
    - Routage +- hiérarchique
    - Routage statique ou dynamique avec vecteurs de distances ou état des liens
  - Réseaux mobiles/ad-hoc/P2P
    - Mécanismes de découverte dynamique et/ou de rédirectionnement des données
  - Routage multicast