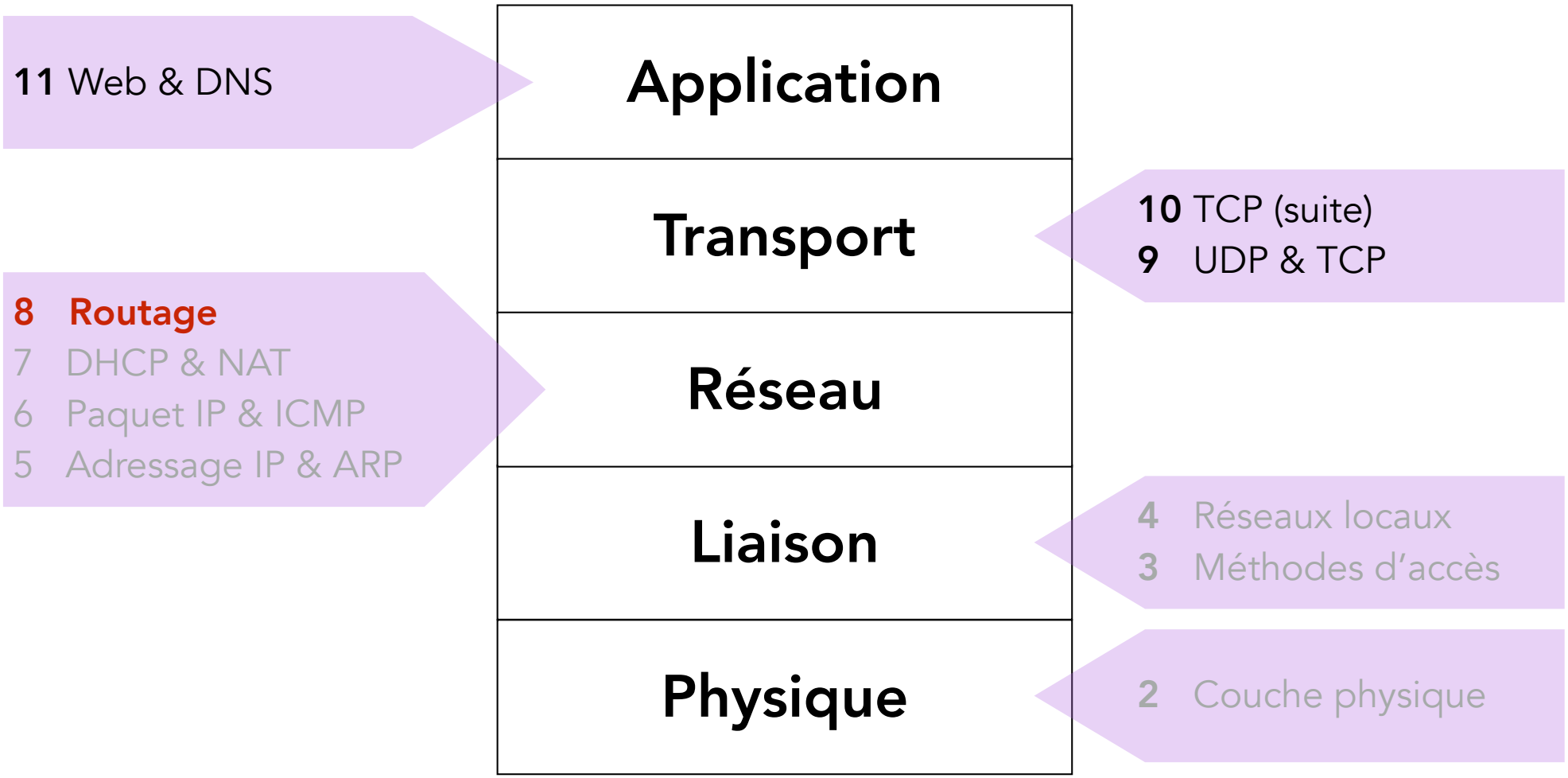
Routage

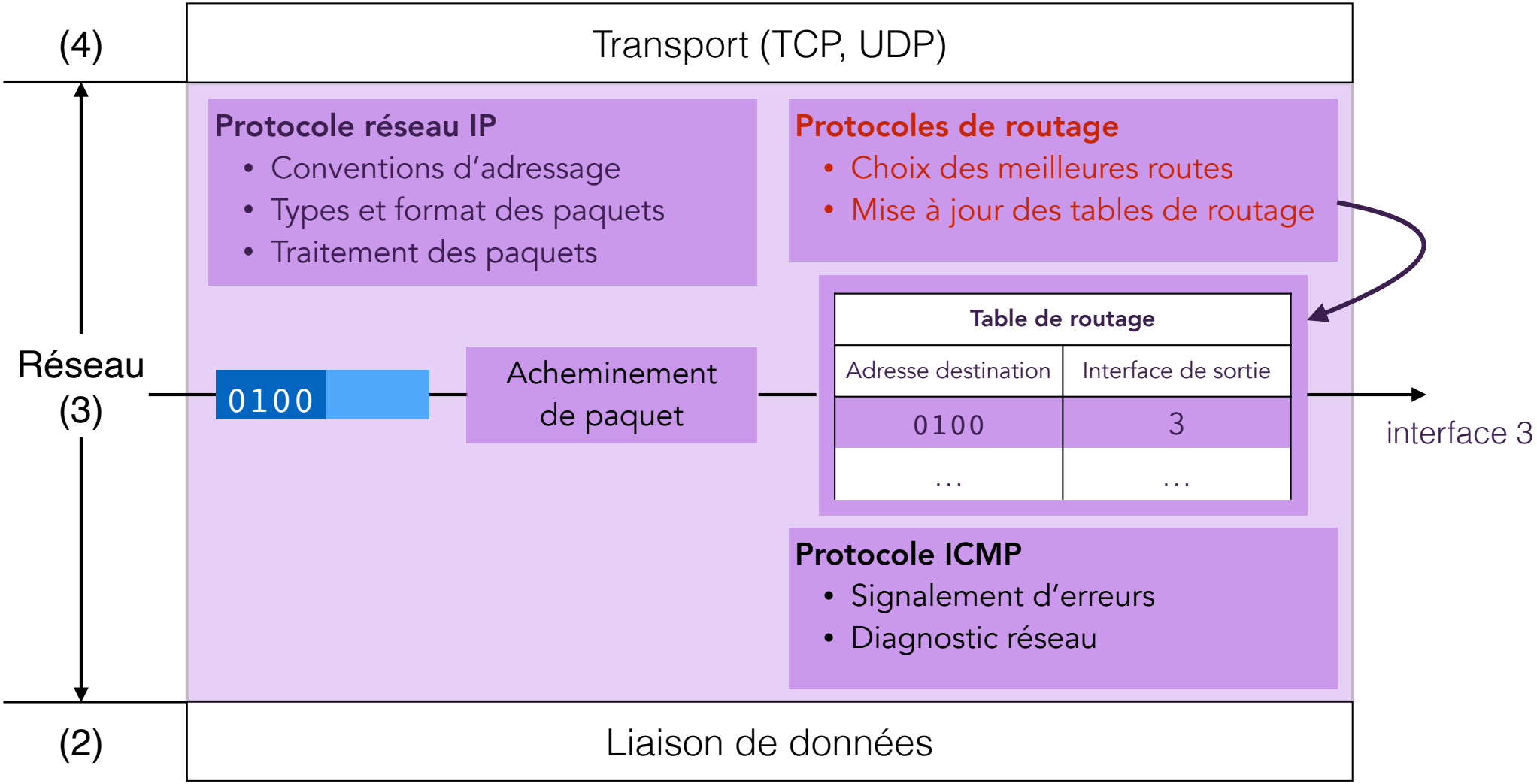
UE LU3IN033 Réseaux
2024-2025

Bruno Baynat
Bruno.Baynat@sorbonne-universite.fr

Programme de l'UE LU3IN033



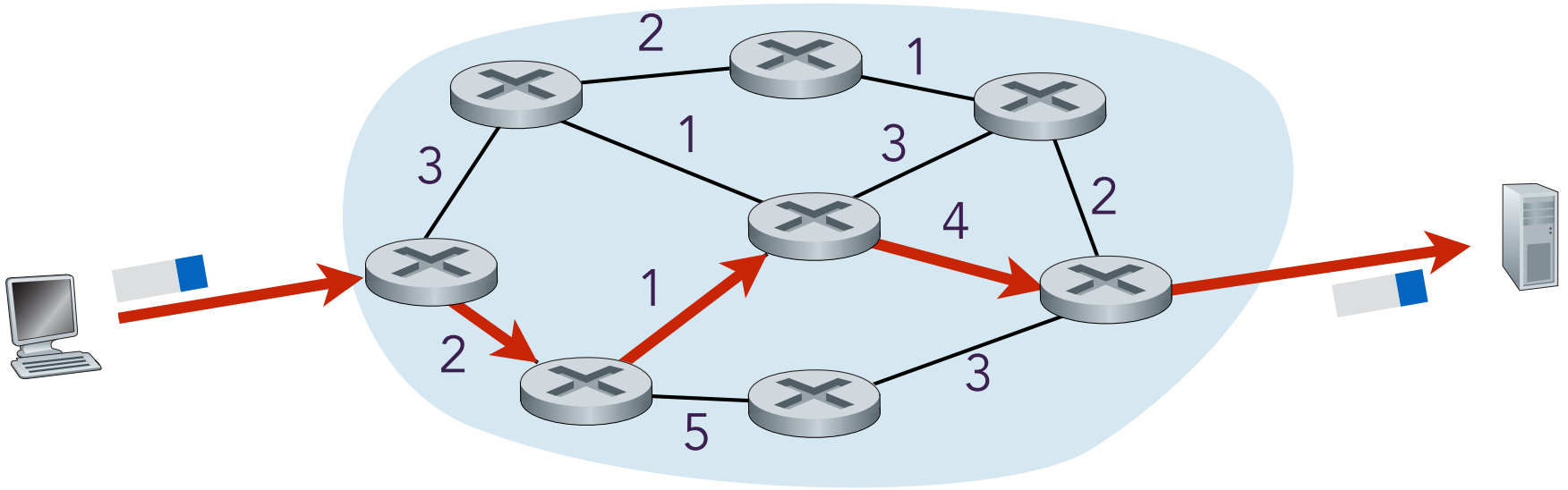
Couche réseau



Plan du cours

- Routage vs Acheminement
- Algorithme de routage vs Protocole de routage
- Routage Distribué vs Centralisé
- Protocoles à état de liens
  - Basé sur l'algorithme de Dijkstra
  - Ex : OSPF - *Open Shortest Path First*
- Protocoles à vecteur de distance
  - Basé sur l'algorithme de Bellman-Ford
  - Ex : RIP - *Routing Information Protocol*
- Protocoles à état de liens vs Protocoles à vecteur de distance
- Routage hiérarchique à deux niveaux
  - Routage intra-domaine : RIP, OSPF
  - Routage inter-domaine : BGP

Acheminement des données



Les paquets de données sont acheminés (« forwardés ») de proche en proche vers leur destination finale en suivant le meilleur chemin

Tables de routage

- Chaque routeur maintient une table de routage
  - une entrée par destination
  - chaque entrée indique l'interface de sortie et le saut suivant pour atteindre la destination correspondante
- À la réception d'un paquet
  - un routeur inspecte l'adresse de destination du paquet
  - inspecte sa table de routage pour déterminer la « meilleure » entrée correspondant à cette adresse
  - achemine le paquet sur l'interface indiquée par cette entrée
- Les routeurs suivants sur le chemin du paquet répètent le même processus
  - le paquet se rapproche saut par saut de sa destination finale

Destination	Masque	Suivant	Interface

D'où proviennent les tables de routage ?

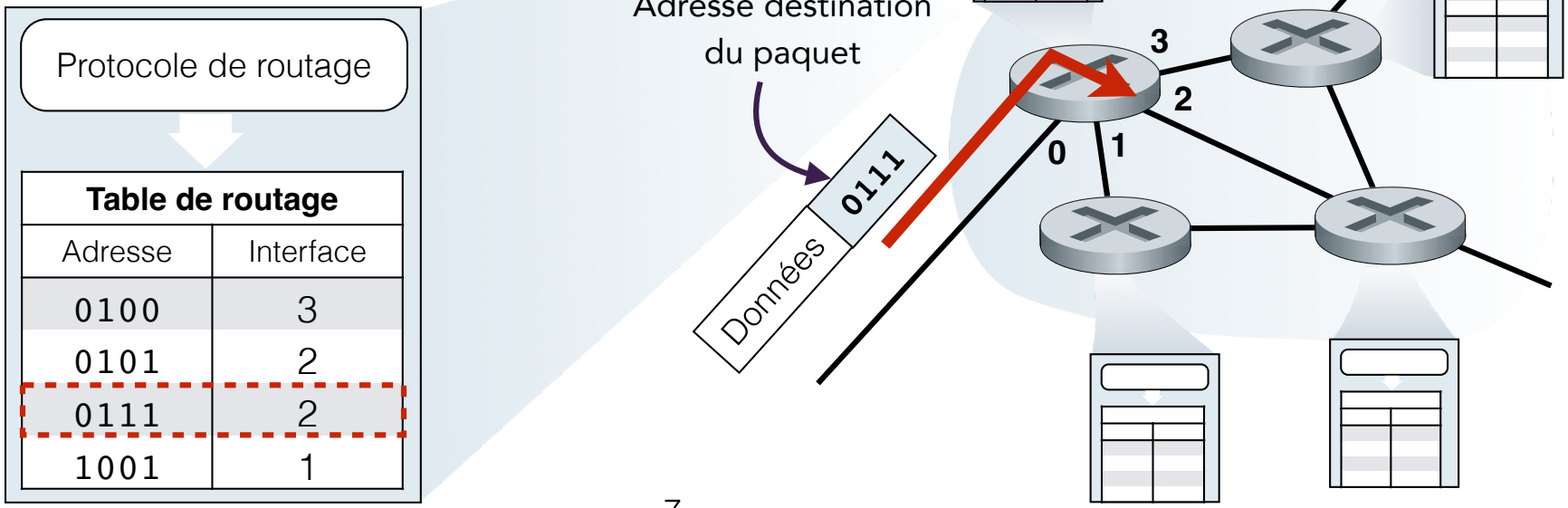
Routage vs Acheminement

Routage

- Création et mise à jour des tables de routage
- Algorithmes de routage
  - Dijkstra, Bellman-Ford, …
- Protocoles de routage
  - OSPF, IS-IS, RIP, BGP, …

Acheminement

- Commutation des paquets de proche en proche
  - choix de l'interface de sortie sur laquelle aiguiller les paquets
  - par inspection des tables de routage



Impact du choix des routes

- Performances de bout-en-bout
  - Le choix du chemin affecte les performances de la communication
  - Métriques de performance : délai de propagation, débit, pertes, …

- Utilisation des ressources réseau
  - Le choix du chemin détermine les ressources utilisées
  - Répartir le trafic sur tous les routeurs et les liens permet d'éviter la congestion du réseau

- Délai de réaction face à des perturbations
  - pannes, changement de topologie, opérations de maintenance, …
  - Limiter les conséquences sur l'acheminement des paquets pendant la convergence des protocoles de routage

Calcul des routes

Statique

L'administrateur configure manuellement les entrées des tables de routage

- Autorise plus de contrôle
- Permet de choisir des routes basés des critères plus subjectifs
- Ne passe pas à l'échelle
- Adaptation lente aux changements de topologie

Dynamique

Les routeurs s'échangent des informations permettant de configurer automatiquement les entrées des tables de routage

- Permet une adaptation rapide aux changements de topologie
- Passe à l'échelle
- Algorithmes distribués complexes à mettre en œuvre
- Consomme CPU, RAM et BP
- Choix des routes basés sur des critères formels

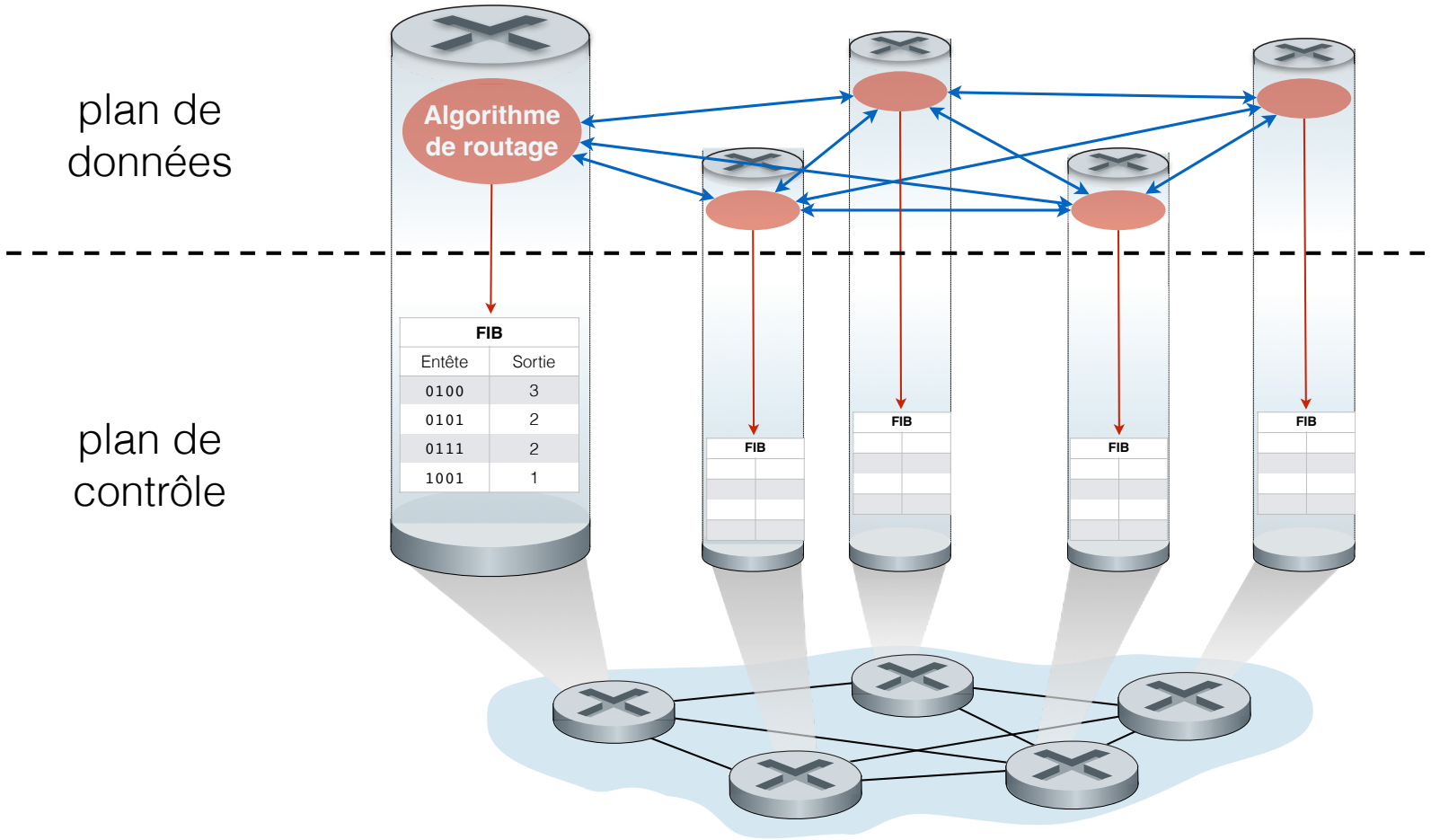
En pratique : un mix des deux
Routage statique aux extrémités, dynamique dans le cœur de réseau

Algorithme de routage vs Protocole de routage

- Algorithme de routage
  - Méthode de calcul théorique permettant de déterminer le « meilleur » chemin entre un expéditeur et un destinataire
    - le plus court (en distance, en nombre de sauts)
    - le plus rapide (en temps)
    - le moins cher
    - le plus sûr
    - …
  - Pour s'exécuter, les algorithmes ont besoin de connaître la topologie du réseau
    - Dijkstra : topologie complète
    - Bellman-Ford : topologie partielle (locale)
- Protocole de routage
  - Mise en oeuvre pratique d'un (ou de plusieurs) algorithme(s) de routage
    - centralisé
    - distribué
  - Définit le format des messages et les règles d'échange entre les routeurs
  - Dans le but de maintenir à jour des tables de routage offrant les meilleures routes

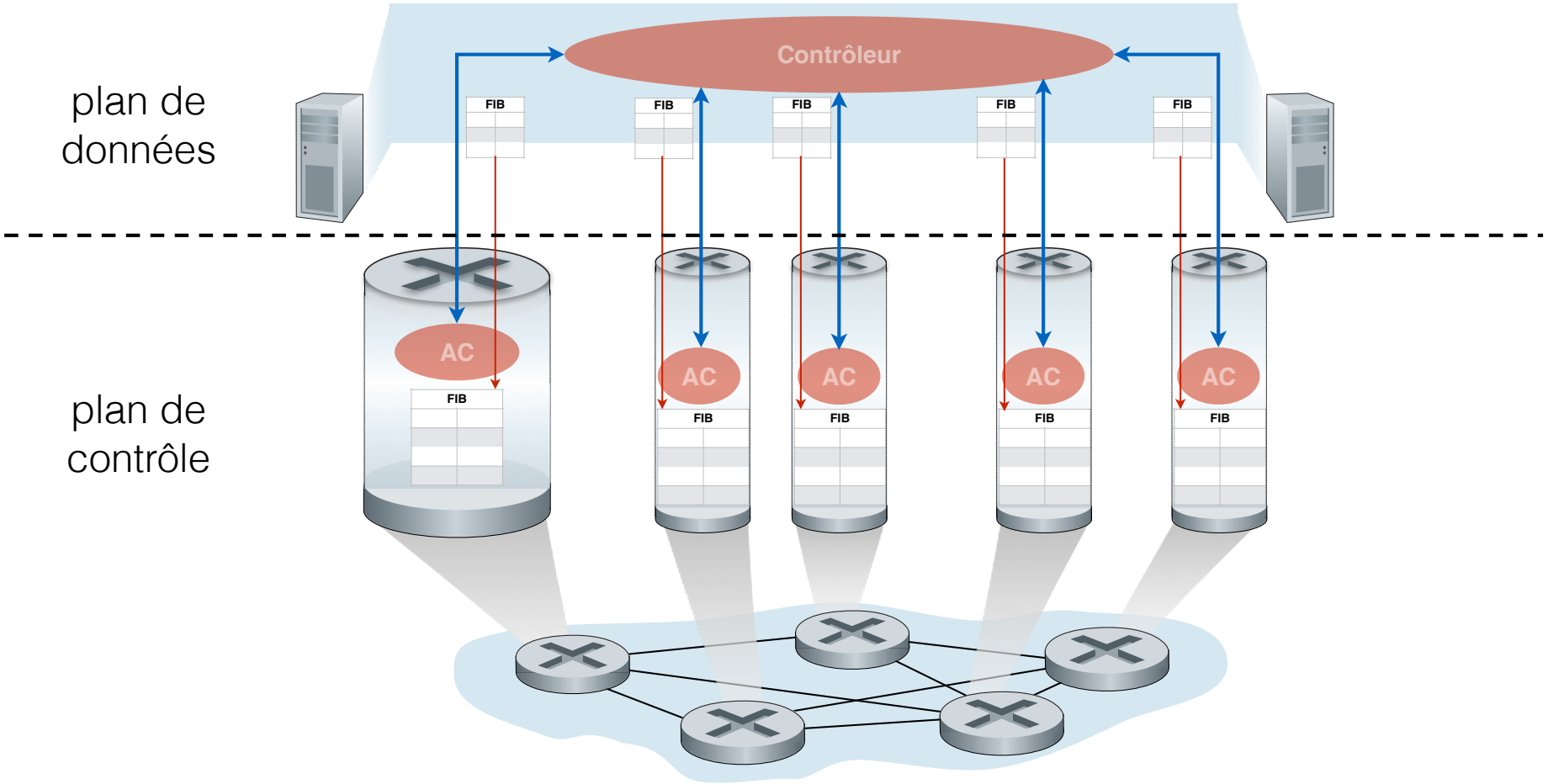
Protocoles distribués

Les routeurs exécutent un algorithme distribué et calculent eux-même les routes incluses dans leur propre table de routage



Protocoles centralisés

Un contrôleur central calcule les routes pour le compte de tous les routeurs et envoie à chacun sa table de routage



Protocoles de routage distribués

Protocoles à état de liens

- Les routeurs envoient périodiquement
  - des paquets « état de liens »
  - à tous les routeurs du réseau (inondation)
- Chaque routeur construit sa propre base de données d'états de liens
  - qui contient la topologie complète du réseau
  - de manière distribuée
- Algorithme de Dijkstra
  - exécuté localement
  - calcul des plus courts chemins vers toutes les destinations du réseau
  - mise à jour des tables de routage
- Protocoles à état de lien
  - OSPF, IS-IS

Protocoles à vecteur de distance

- Les routeurs envoient périodiquement
  - des messages « vecteur de distance »
  - contenant des informations de leur table de routage
  - à leurs voisins directs
- Algorithme de Bellman-Ford
  - algorithme distribué
  - mise à jour de sa propre table de routage sur réception des vecteurs de distance de ses voisins
  - sans avoir la connaissance complète du réseau
- Protocoles à vecteur de distance
  - RIP

Routage à état de liens

Routage à état de lien

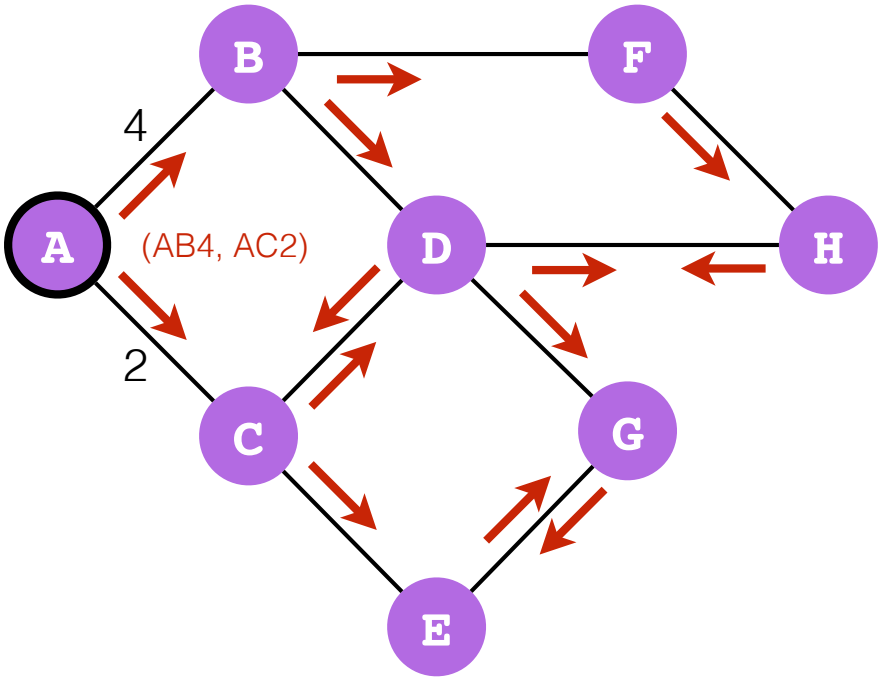
- Un routeur qui s'allume doit
  - découvrir ses voisins
  - estimer le coût des liens qui le relient à ses voisins

- Chaque routeur inonde le réseau de paquets « état de liens » (LSP)
  - périodiquement ou suite à un changement de topologie
  - pour donner à tous les routeurs une vue globale du réseau
  - conservés dans une base de données locale à chaque routeur : LSDB (*Link-State Data Base*)
- Parallèlement chaque routeur exécute localement l'algorithme de Dijkstra
  - calcul du chemin complet du routeur considéré vers toutes les destinations (connues) du réseau
  - calcul de sa propre table de routage

Paquet LSP

- Un paquet LSP (*Link-State Packet*) contient
  - l'identificateur du nœud qui a créé le LSP
  - une liste de voisins directement connectés avec le coût vers chacun
  - un numéro de séquence
  - une durée de validité (TTL)
- Chaque routeur envoie ses paquets LSP
  - périodiquement
  - suite à un changement de topologie
    - rupture ou réparation d'un lien
    - panne ou rétablissement d'un routeur
    - changement de coût d'un lien
- Un routeur qui reçoit un paquet LSP
  - le compare à ceux de sa base de données de liens
  - si le numéro de séquence du LSP reçu est plus récent que celui contenu dans sa base
    - il met à jour sa base de donnée
    - il le retransmet sur chacune de ses voies de sortie (sauf celle sur laquelle le LSP lui est parvenu)

Inondation des LSPs



Inondation fiable

- L'inondation doit être fiable
  - Garantir que tous les routeurs ont reçu les mêmes informations sur la topologie du réseau
  - Synchronisation des BDs des routeurs
- Problèmes
  - Pertes de paquets LSP
  - Réception dans le désordre
  - Boucles
- Solutions
  - Contrôle d'erreur : acquittements et retransmissions
  - Numérotation en séquence des paquets LSP
  - Réduction de la durée de vie des paquets LSP

Quand inonder ?

- Immédiatement
  - à l'initialisation
  - sur changement de topologie
    - panne de lien ou de routeur
    - recouvrement de lien ou de routeur
  - sur changement de configuration
    - changement de coût d'un lien
- Périodiquement
  - pour rafraîchir les informations d'état de lien des BD des routeurs
    - à chaque lien est associé un TTL (MaxAge d'1 heure dans OSPF)
  - pour corriger les corruptions éventuelles d'information

Algorithme de Dijkstra

Initialisation

- **Pour** tous les sommets $v \in S$
  - $d(v) = \infty$
  - $pred(v) = nil$
- $d(s) = 0$
- $E = \varnothing$
- $R = S$

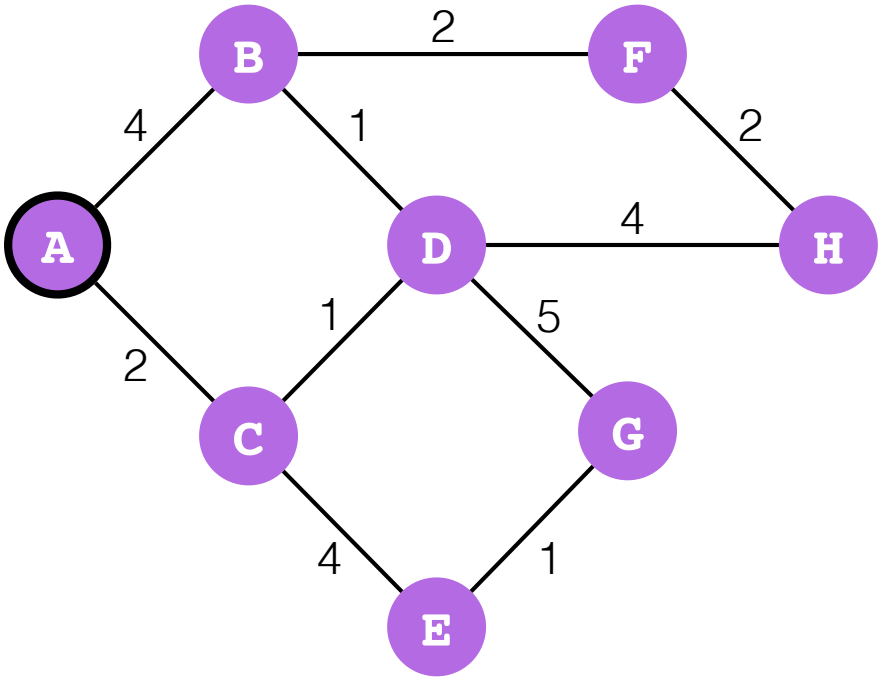
S : ensemble des sommets du Graphe
$d(v)$: distance estimée pour joindre v
$pred(v)$: prédécesseur de v
s : sommet source de tous les chemins
E : ensemble des sommets estimés
R : ensemble des sommets restants

Tant que $R \neq \varnothing$

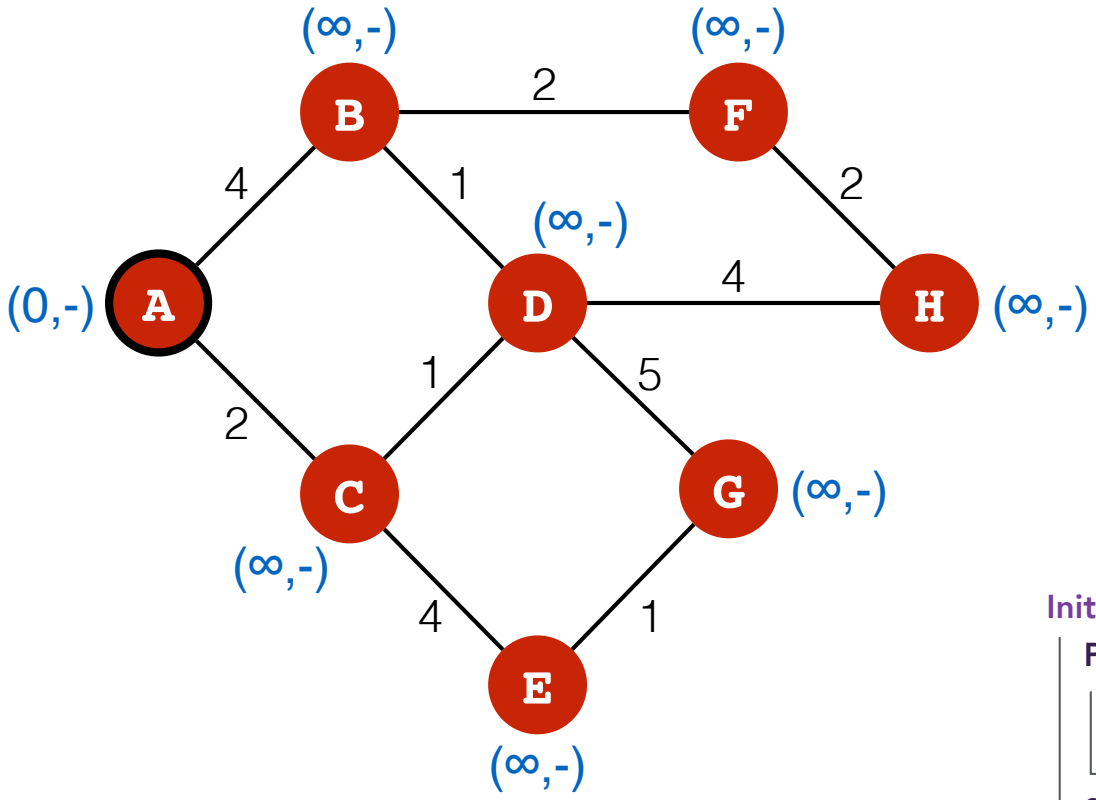
- u = sommet avec l'estimation de pondération minimum de R
- $E = E \cup \{u\}$
- **Pour** tous les sommets $v \in$ voisin(u)
  - si $d(v) > d(u) + c(u,v)$ alors
    - $d(v) = d(u) + c(u,v)$
    - $pred(v) = u$
- $R = R - \{u\}$

$c(u,v)$: coût du lien reliant u à v

Algorithme de Dijkstra



Algorithme de Dijkstra



E = {}

R = {A, B, C, D, E, F, G, H}

Initialisation

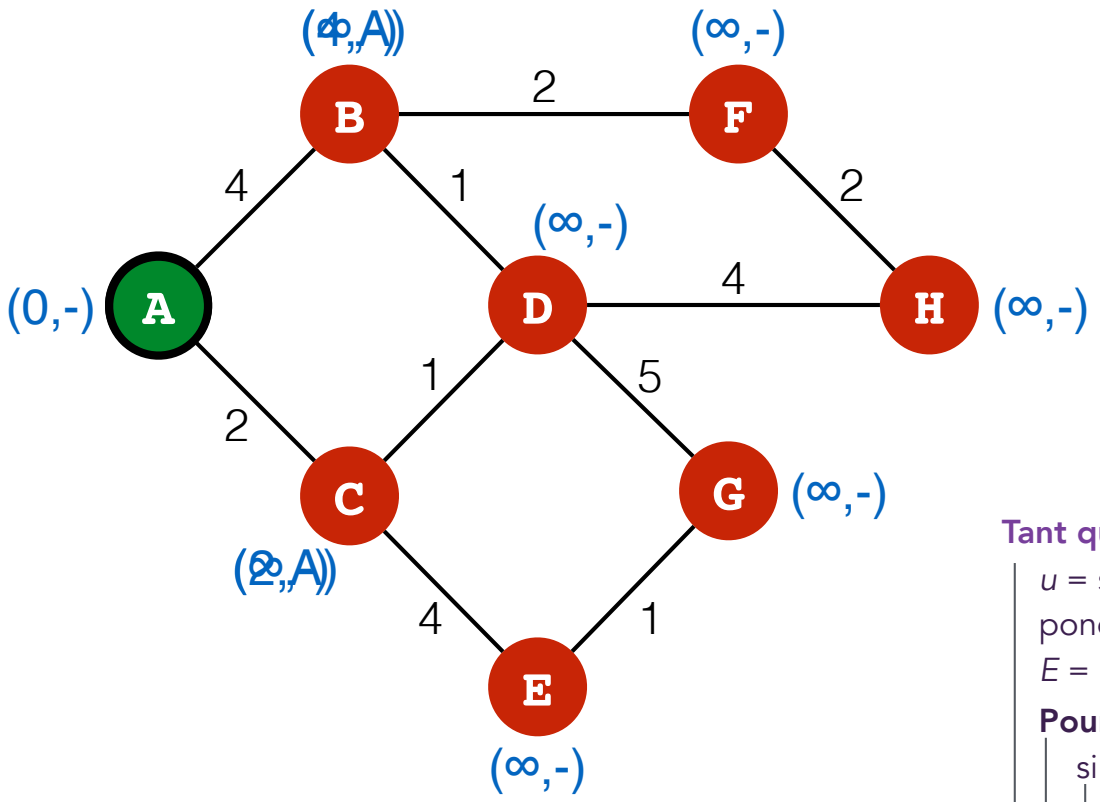
Pour tous les sommets $v \in S$
- $d(v) = \infty$
- $pred(v) = nil$

$d(s) = 0$

$E = \emptyset$

$R = S$

Algorithme de Dijkstra

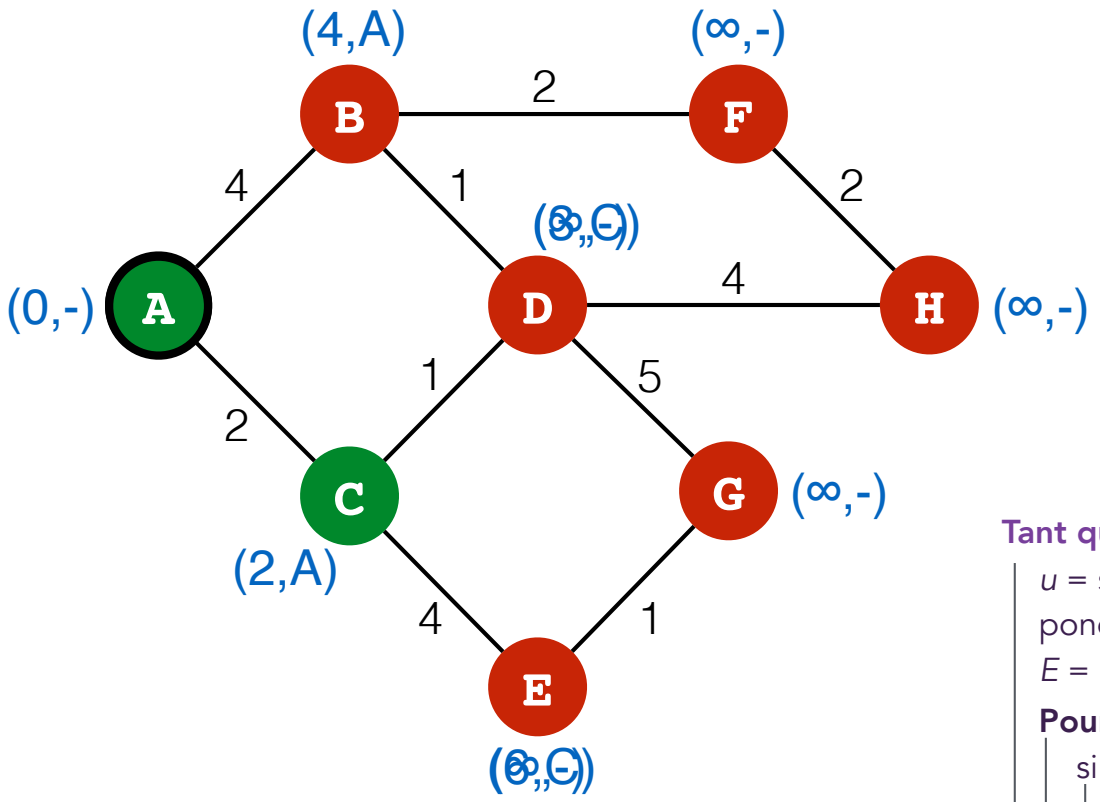


E = {A}

R = {B, C, D, E, F, G, H}

Tant que R ≠ ∅
- *u* = sommet avec l'estimation de pondération minimum de R
- *E* = *E* ∪ {*u*}
- **Pour** tous les sommets v ∈ voisin(u)
  - si d(*v*) > d(*u*) + c(*u*,*v*) alors
    - d(*v*) = d(*u*) + c(*u*,*v*)
    - pred(*v*) = *u*
- R = R - {*u*}

Algorithme de Dijkstra

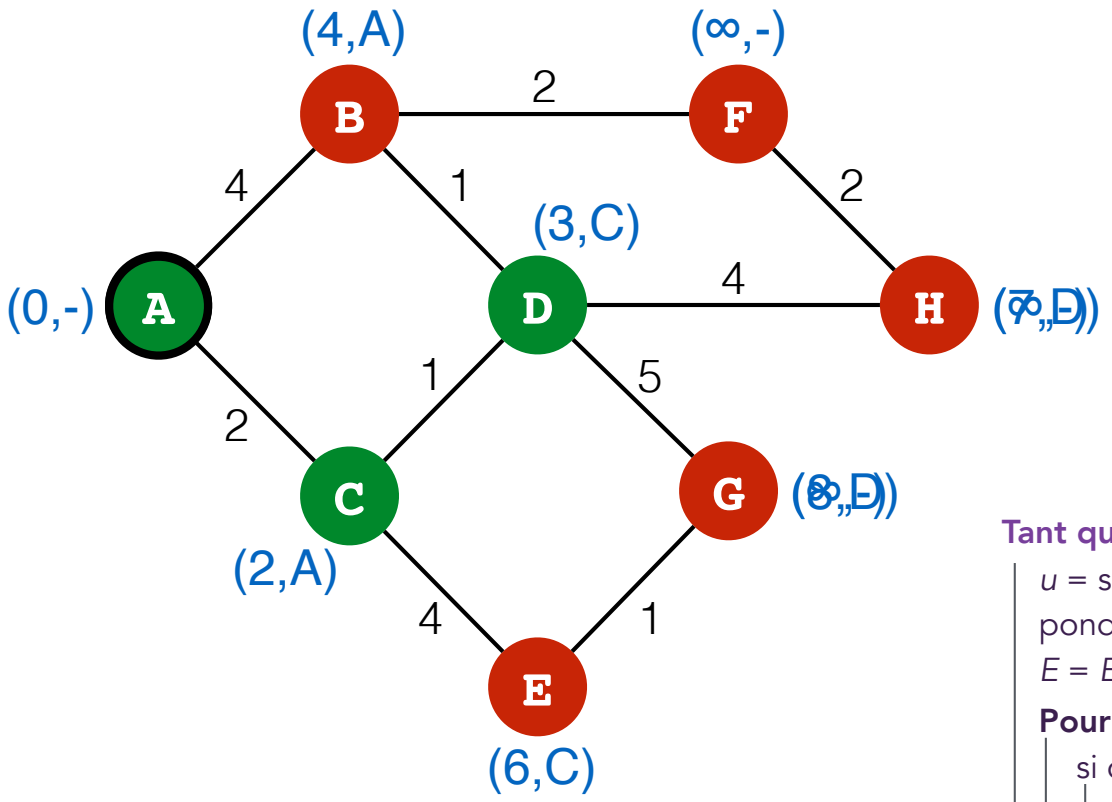


E = {A, C}

R = {B, D, E, F, G, H}

Tant que R ≠ ∅
- *u* = sommet avec l'estimation de pondération minimum de R
- *E* = *E* ∪ {*u*}
- **Pour** tous les sommets v ∈ voisin(u)
  - si d(*v*) > d(*u*) + c(*u*,*v*) alors
    - d(*v*) = d(*u*) + c(*u*,*v*)
    - pred(*v*) = *u*
- R = R - {*u*}

Algorithme de Dijkstra

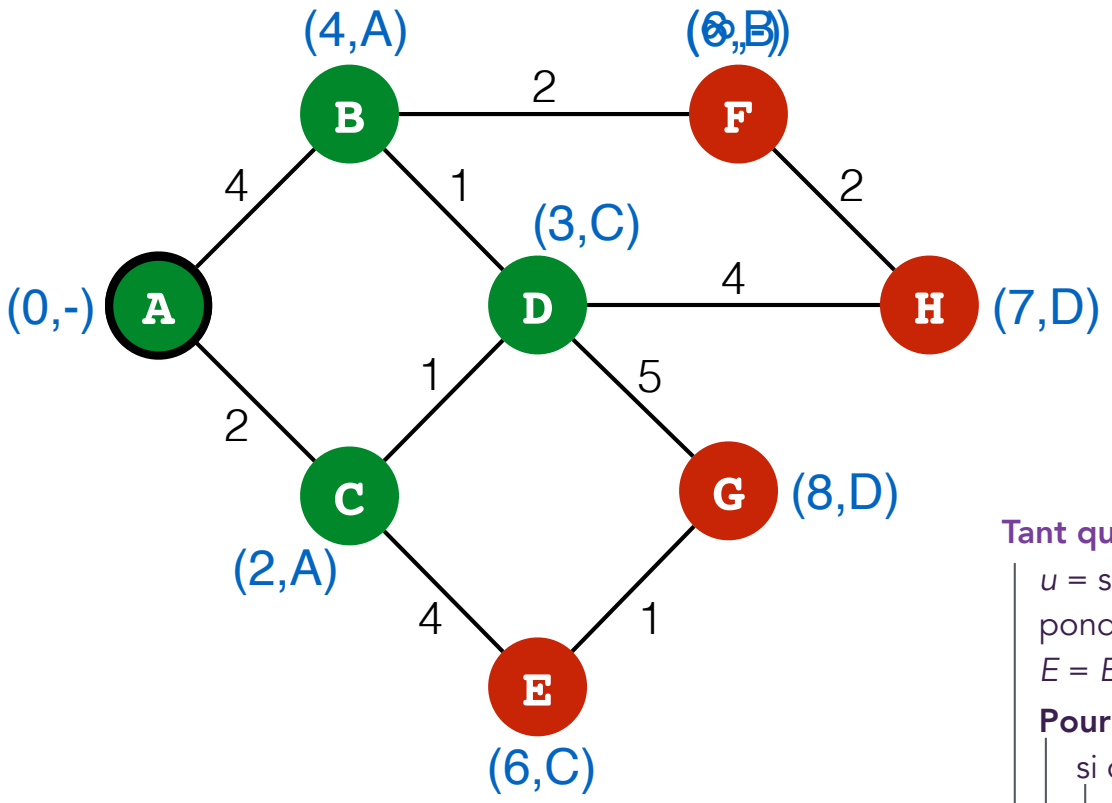


E = {A, C, D}

R = {B, F, E, G, H}

Tant que R ≠ ∅
- *u* = sommet avec l'estimation de pondération minimum de R
- *E* = *E* ∪ {*u*}
- **Pour** tous les sommets v ∈ voisin(u)
  - si d(*v*) > d(*u*) + c(*u*,*v*) alors
    - d(*v*) = d(*u*) + c(*u*,*v*)
    - pred(*v*) = *u*
- R = R - {*u*}

Algorithme de Dijkstra



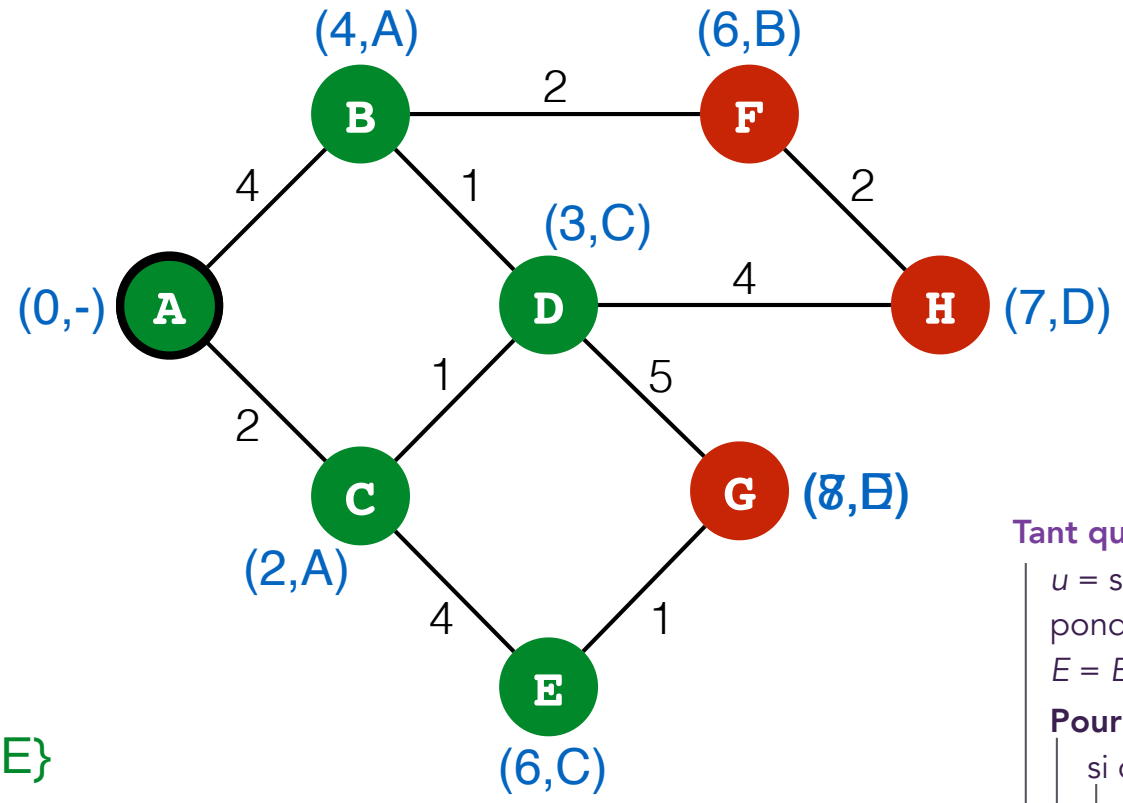
E = {A, C, D, B}

R = {B, E, G, H} F}

Tant que R ≠ ∅

- *u* = sommet avec l'estimation de pondération minimum de R
- *E* = *E* ∪ {*u*}
- **Pour** tous les sommets v ∈ voisin(u)
  - si d(*v*) > d(*u*) + c(*u*,*v*) alors
    - d(*v*) = d(*u*) + c(*u*,*v*)
    - pred(*v*) = *u*
- R = R - {*u*}

Algorithme de Dijkstra

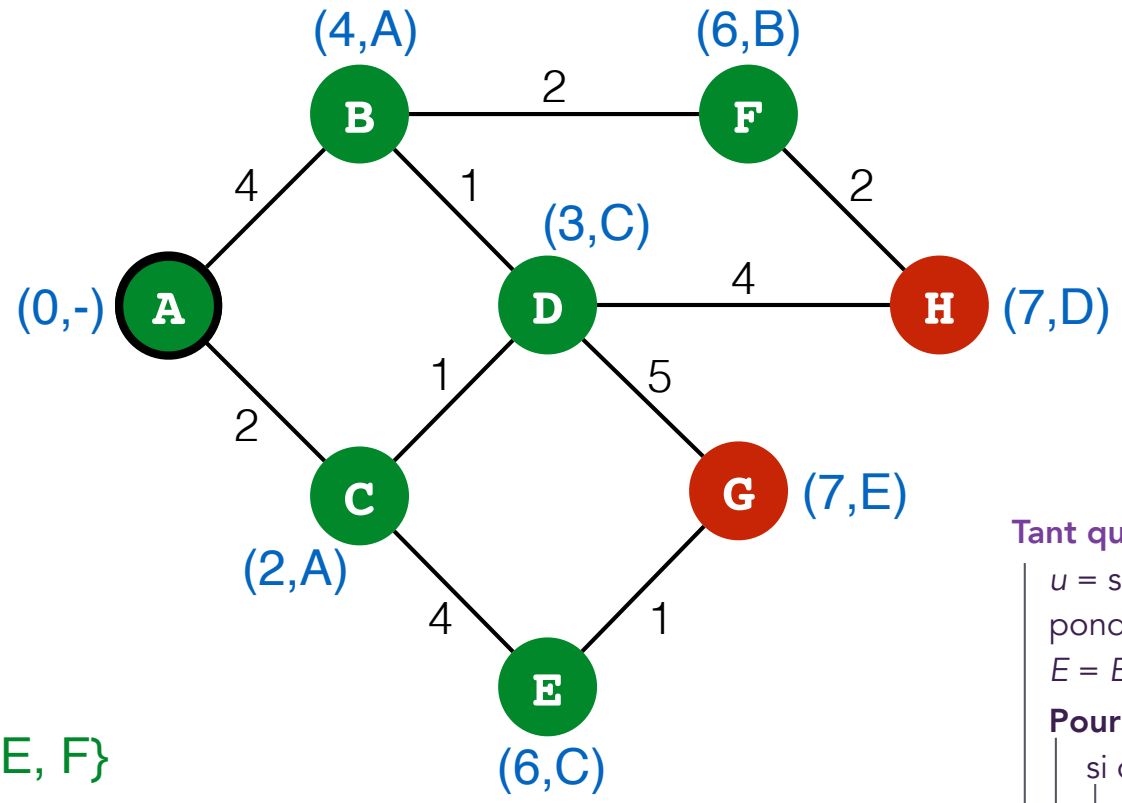


E = {A, C, D, B} E}

R = {E,, G,, H} H}

Tant que R ≠ ∅
- *u* = sommet avec l'estimation de pondération minimum de R
- *E* = *E* ∪ {*u*}
- **Pour** tous les sommets v ∈ voisin(u)
  - si d(*v*) > d(*u*) + c(*u*,*v*) alors
    - d(*v*) = d(*u*) + c(*u*,*v*)
    - pred(*v*) = *u*
- R = R - {*u*}

Algorithme de Dijkstra

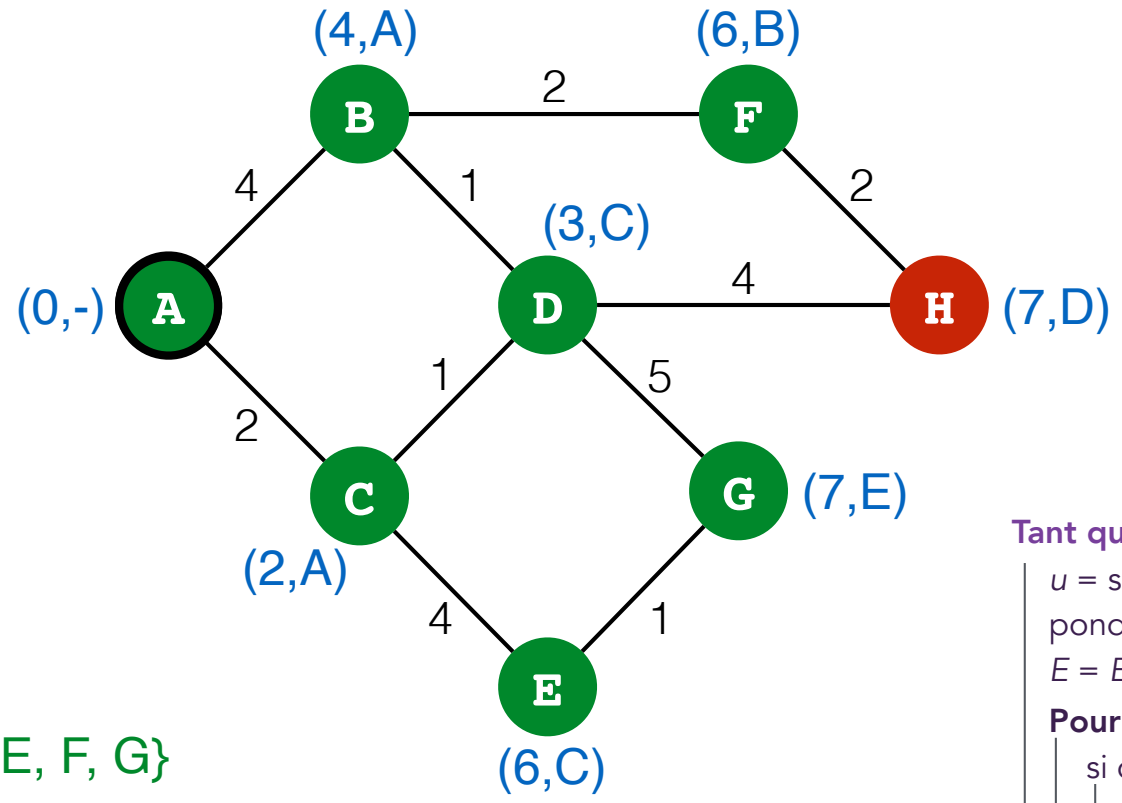


E = {A, C, D, B, E, F}

R = {G, H}

Tant que R ≠ ∅
- *u* = sommet avec l'estimation de pondération minimum de R
- *E* = *E* ∪ {*u*}
- **Pour** tous les sommets v ∈ voisin(u)
  - si d(*v*) > d(*u*) + c(*u*,*v*) alors
    - d(*v*) = d(*u*) + c(*u*,*v*)
    - pred(*v*) = *u*
- R = R - {*u*}

Algorithme de Dijkstra

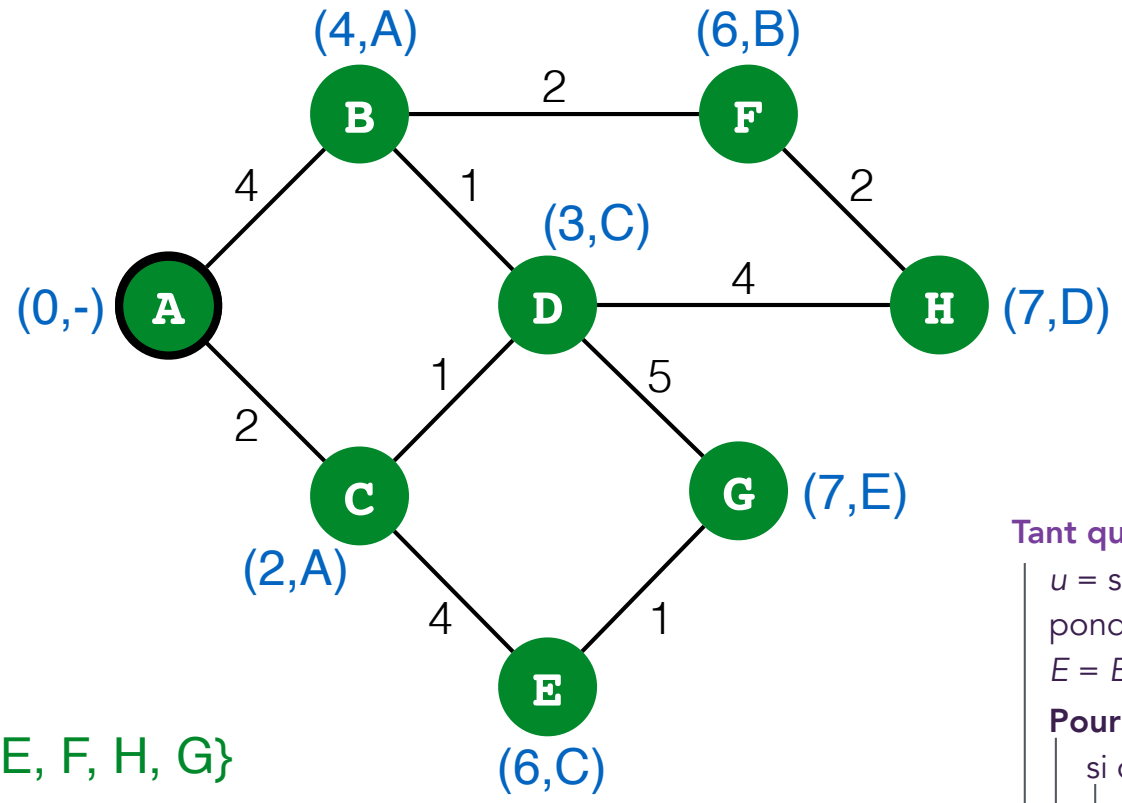


E = {A, C, D, B, E, F, G}

R = {H}

Tant que R ≠ ∅
- *u* = sommet avec l'estimation de pondération minimum de R
- *E* = *E* ∪ {*u*}
- **Pour** tous les sommets v ∈ voisin(u)
  - si d(*v*) > d(*u*) + c(*u*,*v*) alors
    - d(*v*) = d(*u*) + c(*u*,*v*)
    - pred(*v*) = *u*
- R = R - {*u*}

Algorithme de Dijkstra



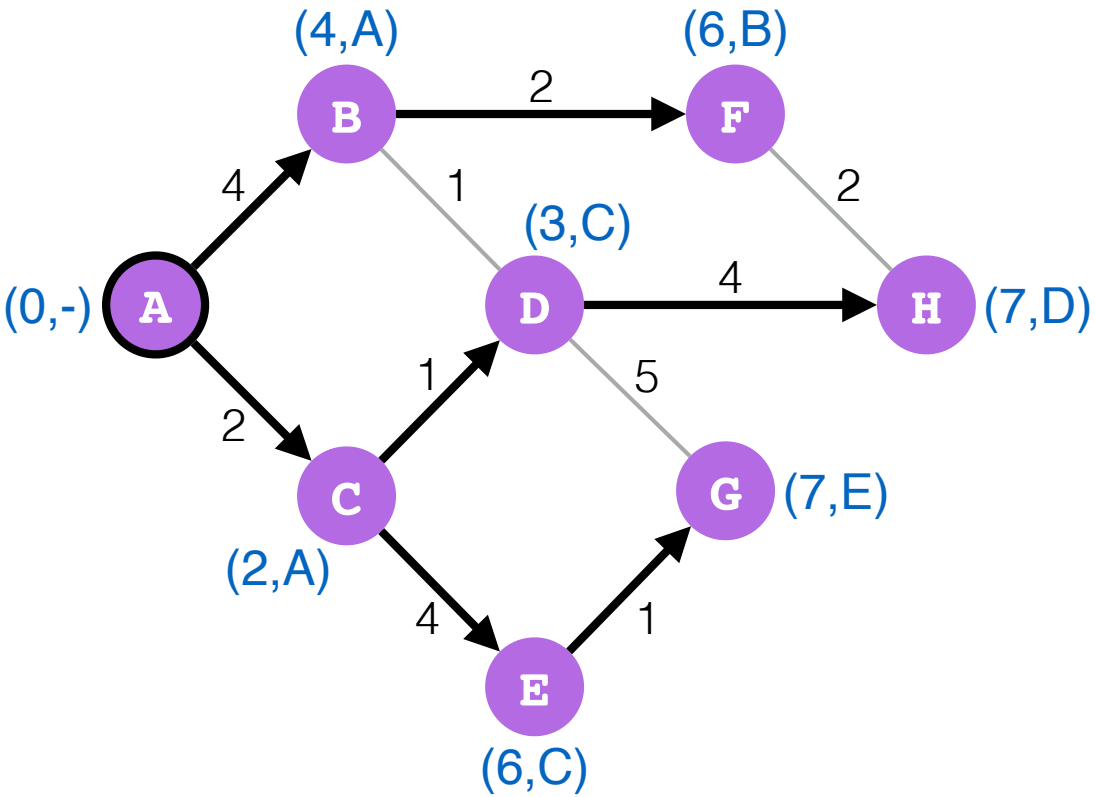
E = {A, C, D, B, E, F, H, G}

R = {}

Tant que R ≠ ∅
- *u* = sommet avec l'estimation de pondération minimum de R
- *E* = *E* ∪ {*u*}
- **Pour** tous les sommets v ∈ voisin(u)
  - si d(*v*) > d(*u*) + c(*u*,*v*) alors
    - d(*v*) = d(*u*) + c(*u*,*v*)
    - pred(*v*) = *u*
- R = R - {*u*}

Arbre des plus courts chemins

Arbre des plus courts chemins

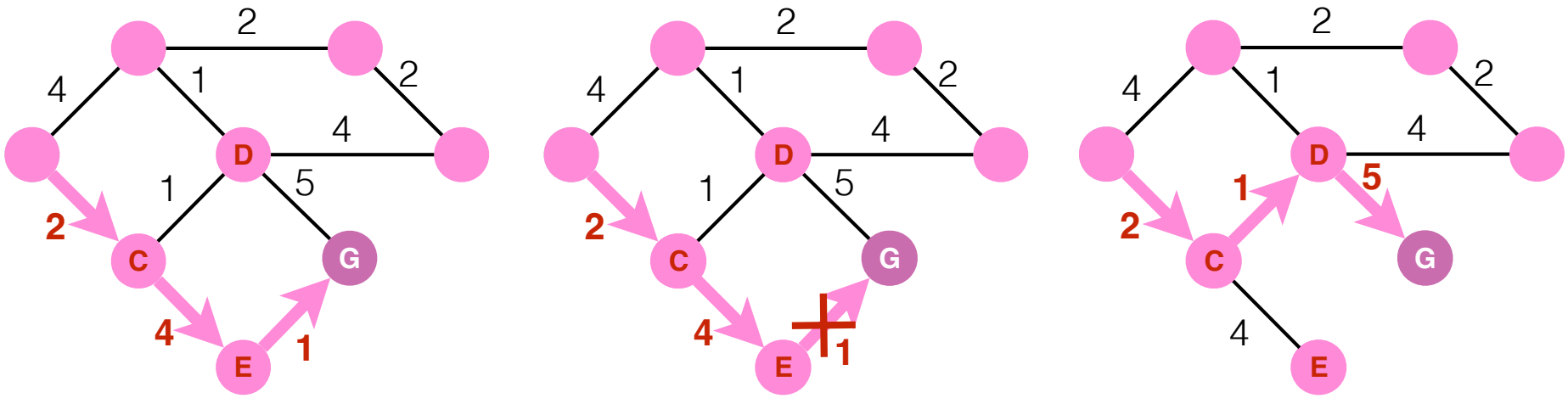


Extraction des sauts suivants

Table de routage de A	
Destination	Suivant
B	B
C	C
D	C
E	C
F	B
G	C
H	C

Changements de topologie

- Changements de topologie
  - pannes / réparations, changements de configuration, opérations de maintenance, …
- Les changements de topologie doivent être détectés et annoncés le plus rapidement possible
  - détection du changement
  - propagation de ce changement à tous les routeurs
  - recalcul des routes
  - mise à jour des tables de routage



Convergence du protocole

- Source du délai de convergence du protocole
  - latence de détection de la panne
  - inondation de l'information d'état de lien
  - recalcul de l'arbre couvrant des plus courts chemins
  - mise à jour de la table d'acheminement
- Conséquence possibles sur l'acheminement des paquets de données
  - pertes de paquets
    - « trous noirs »
    - expiration de TTL
  - déséquencements de paquets
  - boucles de routage
  - délai de réception excessifs
    - non compatible avec les applications temps réel (VoIP, vidéo, jeux en ligne, …)

Réduire le délai de convergence

- Réagir plus vite à des pannes
  - temporisateurs de messages « hello » plus petits
  - détection de la panne au niveau liaison
  - inondation sélective
  - priorisation des paquets LSP
- Recalculer plus rapidement les routes
  - augmentation de la puissance CPU des routeurs
  - algorithme de Dijkstra incrémental
- Mettre à jour les tables d'acheminement plus efficacement
  - envoie uniquement des LSPs qui ont changés
  - mises-à-jour incrémentales des bases de données

Open Shortest Path First (OSPF)

- Protocole à état de liens
  - Les routeurs envoient un message « hello » sur chaque lien
    - toutes les 10 secondes
  - L'absence de réponse à un « hello » pendant 40 secondes indique une rupture de lien ou la panne du voisin
  - Les routeurs inondent leurs LSPs
    - périodiquement : toutes les 30 minutes
    - immédiatement : suite à un changement de topologie
- OSPF est adapté
  - aux réseaux complexes (comportant plusieurs sous-réseaux)
  - aux réseaux de grande taille

Routage à vecteur de distance

Routage à vecteur de distance

- Un routeur doit uniquement connaître ses voisins et le coût pour les atteindre
- Chaque routeur envoie des messages « vecteur de distance »
  - contenant des informations issues de sa propre table de routage
    - la liste des destinations connues du routeur
    - la distance du chemin connu du routeur pour chacune de ces destinations
  - les vecteurs de distance sont envoyés uniquement aux voisins directs
    - périodiquement ou suite au changement des tables de routage
- Chaque routeur exécute l'algorithme distribué de Bellman-Ford
  - sur réception du vecteur de distance d'un de ses voisins
  - pour mettre à jour sa propre table de routage

Equation de Bellman-Ford

- Coût du plus court chemin de x vers y : $d_x(y)$
- Equation de Bellman-Ford

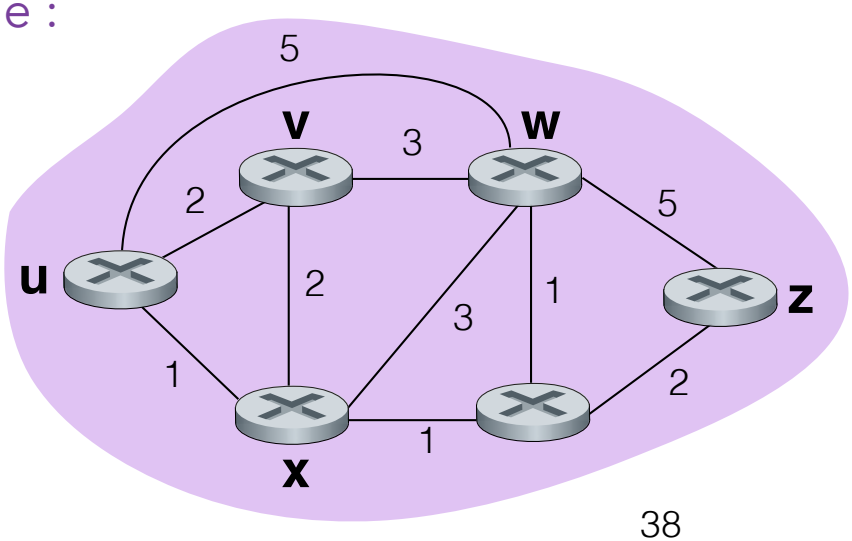
$$d_x(y) = \min_v\{c(x,v) + d_v(y)\}$$

$d_v(y)$: coût du plus court chemin de v vers y

$c(x,v)$: coût du lien de x vers v

$\min_v$: minimum sur tous les voisins v de x

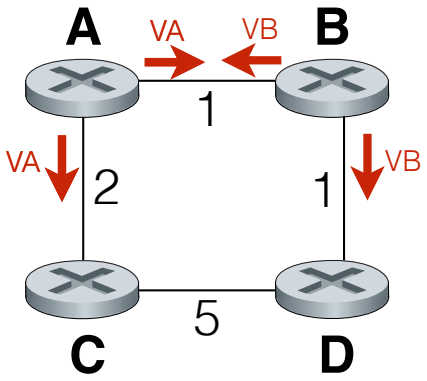
- Exemple :



$$\begin{aligned} d_u(z) &= \min\{c(u,v) + d_v(z), \\ &\qquad c(u,x) + d_x(z), \\ &\qquad c(u,w) + d_w(z)\} \\ &= \min\{2 + 5, \\ &\qquad 1 + 3, \\ &\qquad 5 + 3\} \\ &= 4 \end{aligned}$$

Algorithme de Bellman-Ford

- Algorithme distribué asynchrone
- Les routeurs maintiennent un vecteur de distance : $D_x = [d_x(y) : y \in N]$
- Initialement
  - les routeurs connaissent leurs voisins directs et le coût pour les atteindre
  - un routeur x remplit son vecteur D_x des routes directes vers ses voisins
- Périodiquement
  - les routeurs envoient spontanément leur vecteur de distance à leurs voisins directs
- Sur réception d'un vecteur de distance
  - un routeur compare chaque destination du vecteur reçu à celle de son propre vecteur en utilisant l'équation de Bellman-Ford
  - si la route annoncée correspond à une destination inconnue
    - il ajoute une entrée à son vecteur
  - si la route annoncée correspond à une destination connue mais avec un meilleur coût
    - il met à jour l'entrée correspondante de son vecteur



Routeur A

Dest.	Suiv.	Dist.
B	B	1
C	C	2

VA=(B1,C2)

Dest.	Suiv.	Dist.
A	A	1
D	D	1
C	A	3

VB=(A1,D1,C3)

Dest.	Suiv.	Dist.
B	B	1
C	C	2
D	B	2

VA=(B1,C2,D2)

Dest.	Suiv.	Dist.
A	A	1
D	D	1
C	A	3

Routeur B

Dest.	Suiv.	Dist.
A	A	1
D	D	1

Routeur C

Dest.	Suiv.	Dist.
A	A	2
D	D	5

Dest.	Suiv.	Dist.
A	A	2
D	D	5
B	A	3

Dest.	Suiv.	Dist.
A	A	2
D	A	4
B	A	3

Routeur D

Dest.	Suiv.	Dist.
B	B	1
C	C	5

Dest.	Suiv.	Dist.
B	B	1
C	B	4
A	B	2

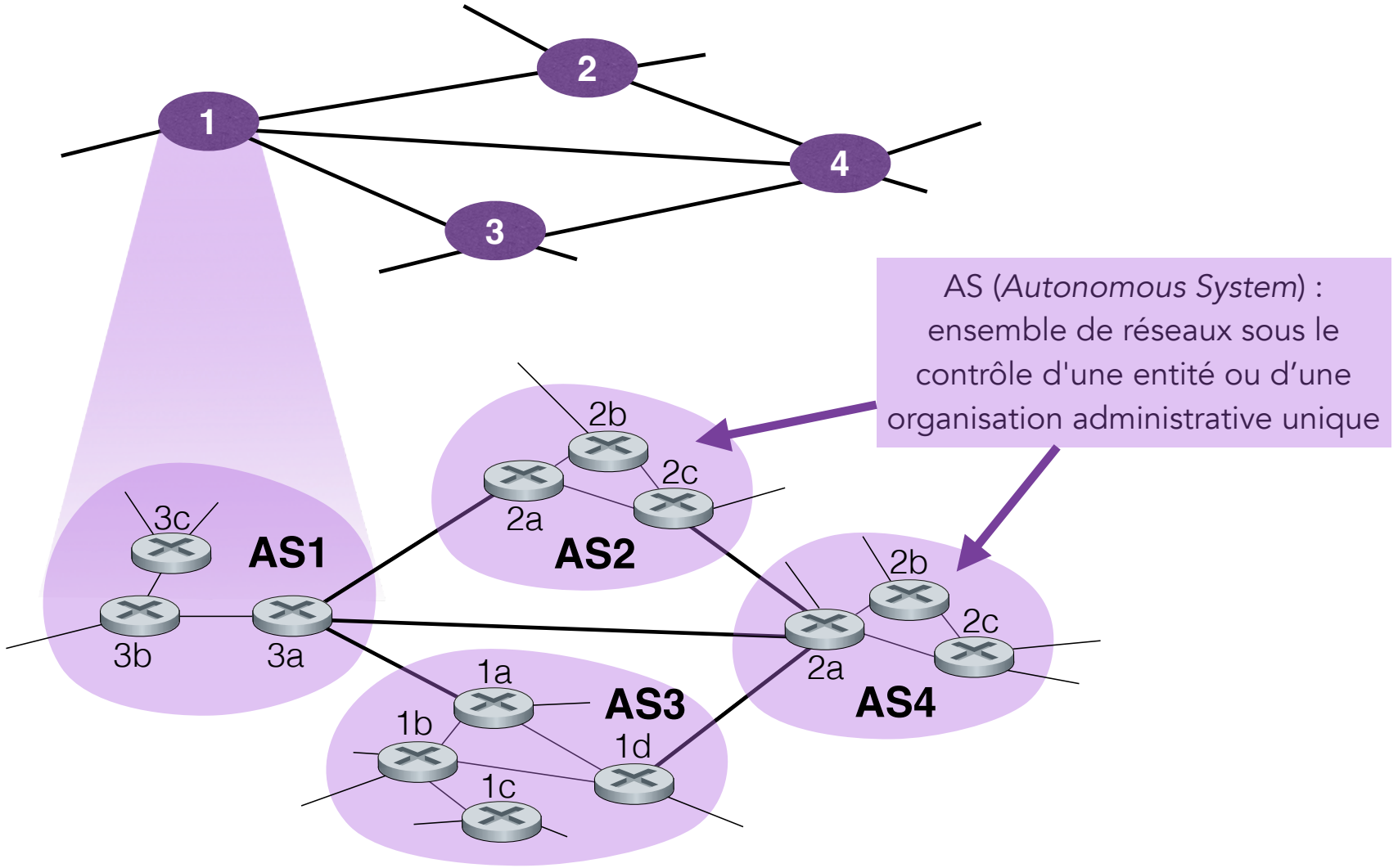
Routing Information Protocol (RIP)

- Protocole à vecteur de distance
  - Les routeurs envoient leurs vecteurs de distance
    - périodiquement : toutes les 30 secondes
    - immédiatement : suite à une mise à jour provoquant un changement de route
  - L'absence de réception de vecteurs de distance sur un lien pendant 180 secondes indique la rupture du lien ou la panne du voisin
- Coût des liens
  - Tous les liens ont un coût de 1
  - Les distances valides varient de 1 à 15
  - 16 représente l'infini
    - permet de réduire le problème du comptage à l'infini
- RIP est limité aux réseaux de taille restreinte

Comparaison des protocoles de routage

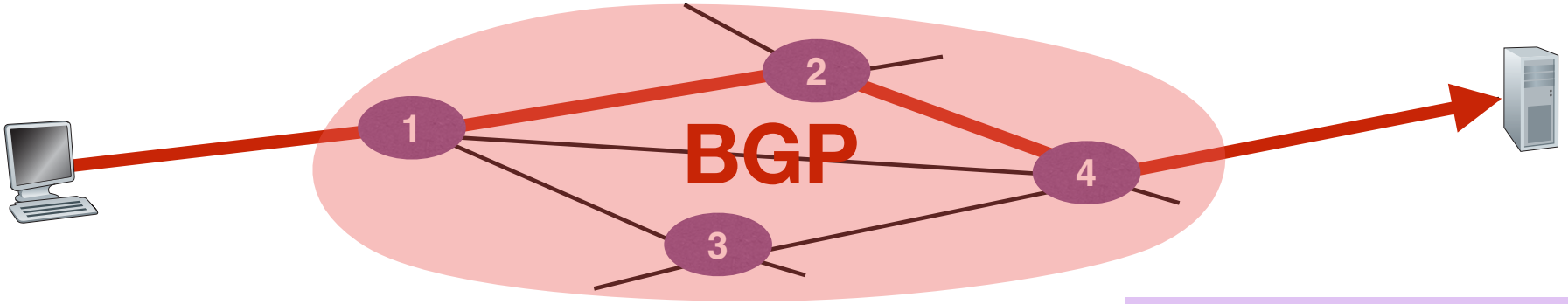
Etat de liens	Vecteur de distance
• Plus complexes à mettre en œuvre et à configurer (base de données d'état de lien)	• Plus simples à mettre en œuvre et à configurer
• Grosse consommation de bande passante (diffusion des informations d'état de lien à tous les routeurs)	• Faible consommation de bande passante (informations de routage envoyées uniquement aux voisins)
• Gourmand en CPU et en RAM	• Faible consommation de CPU et de RAM
• Convergence rapide (mises à jour d'état de lien entrainant immédiatement une mise à jour des tables)	• Convergence lente et problèmes d'instabilité dans certains cas (comptage à l'infini)
• Résistance au facteur d'échelle (adaptés aux réseaux de grande taille)	• Manque de scalabilité (pas adaptés aux réseaux de grande taille)
• Vision globale du réseau	• Vision locale du réseau

Internet : un réseau de réseaux



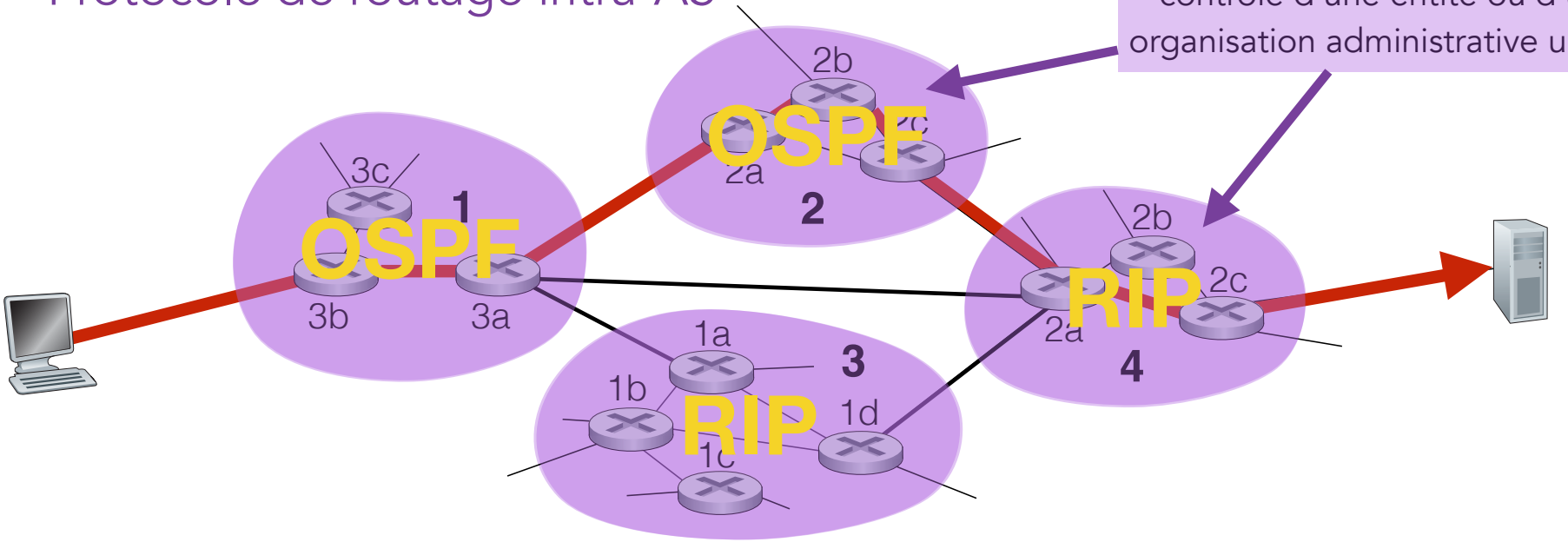
Routage à deux niveaux

- Protocole de routage inter-AS



- Protocole de routage intra-AS

AS : ensemble de réseaux sous le contrôle d'une entité ou d'une organisation administrative unique



Conclusion

- Le routage est l'opération qui consiste à trouver dans un réseau le « meilleur » chemin entre une source et une destination
- Les protocoles de routage permettent aux routeurs de créer et de mettre à jour leurs tables de routage
- Pour cela les routeurs s'échangent des informations selon des algorithmes distribués
  - Etats de lien (OSPF, IS-IS)
    - permettant à chaque routeur d'avoir une vue complète du réseau
    - d'exécuter localement un algorithme de plus court chemin : Dijkstra
    - et d'en déduire sa table de routage
  - Vecteurs de distance (RIP)
    - permettant à chaque routeur d'échanger avec leurs voisins les informations de leurs tables de routage
    - de construire et de mettre à jour leur propre table de routage
    - à l'aide d'un algorithme distribué : Ford-Bellman
- Le délai de convergence de ces protocoles est très important
  - Tant que les tables ne sont pas à jour, l'acheminement des paquets peut être incorrect
    - pertes de paquets, déséquencements, trous noirs, boucles d'acheminement, …
  - Ils doivent donc réagir rapidement à des changements de topologie

A faire

- Cours 8
  - à relire attentivement
- Devoir 8 sur Moodle
  - date de rendu : dimanche 27 octobre