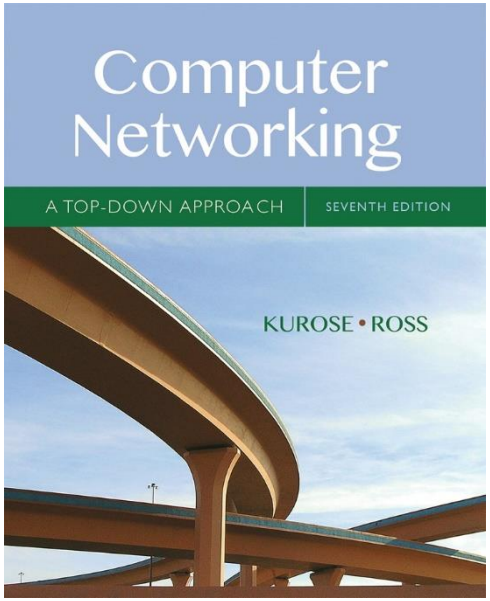
Chapitre IV
La couche réseau



Computer Networking: A Top Down Approach
7ème édition
Jim Kurose, Keith Ross
Addison-Wesley
2017

Le plan de contrôle

A note on the use of these Powerpoint slides:
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

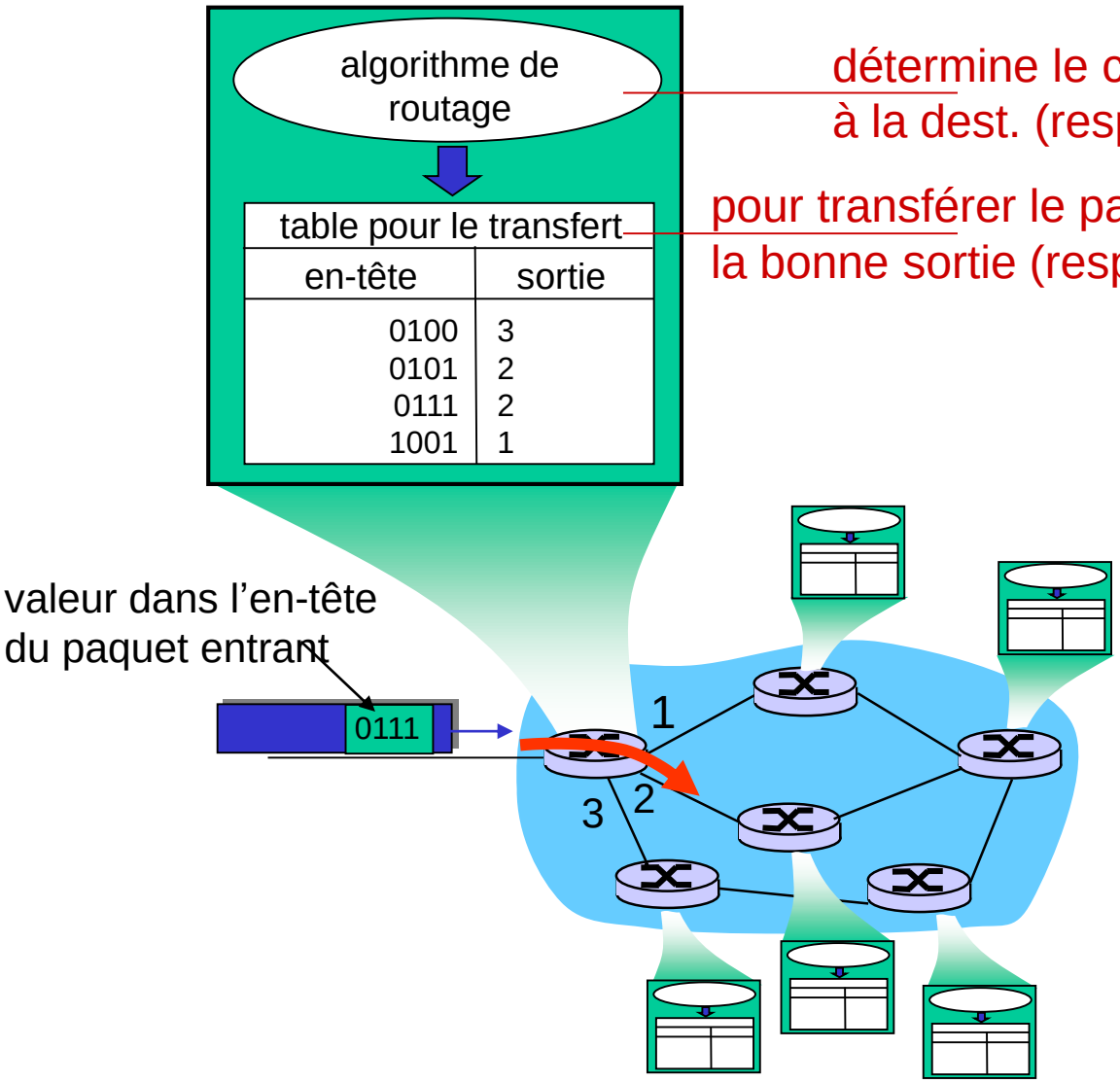
Thanks and enjoy! JFK/KWR

Chapitre IV (suite)

4.4 algorithmes de routage

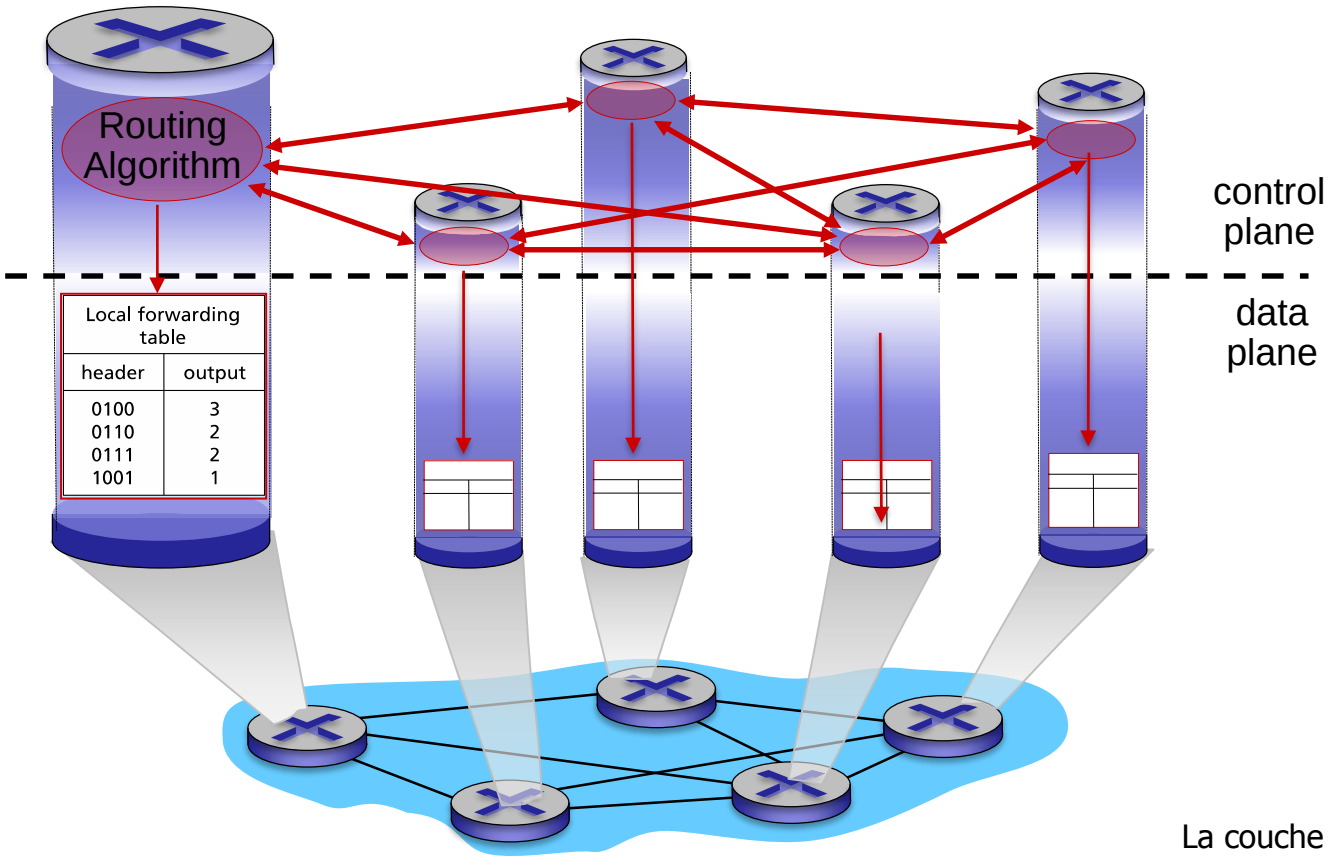
- par états des liens
- par vecteurs de distance

Routage et transfert



Plan de contrôle par routeur

Des composants, dans chaque routeur, d'un algorithme de routage individuel interagissent au niveau du plan de contrôle pour calculer les tables de transfert

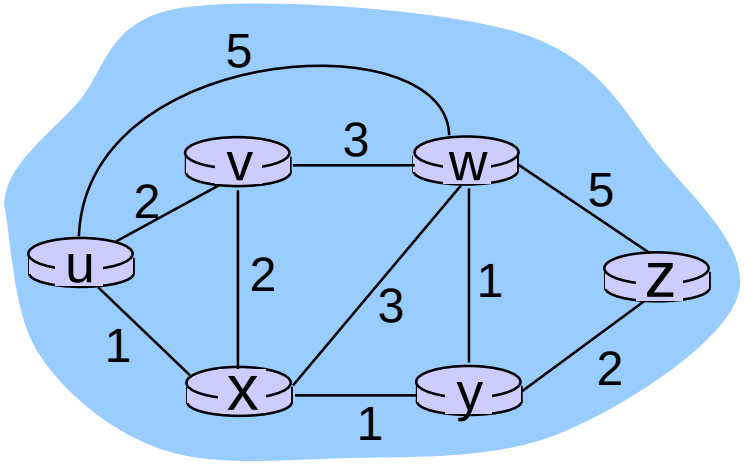


Protocoles de routage

Objectif du protocole de routage: déterminer les "bons" chemins (équivalent à des routes), de l'émetteur au récepteur, à travers un réseau de routeurs.

- chemin: séquence des routeurs traversés par les paquets allant d'une machine source initiale à une machine destination finale.
- "bon": moindre "coût", "plus rapide", "moins congestionné"

Graphe: une abstraction

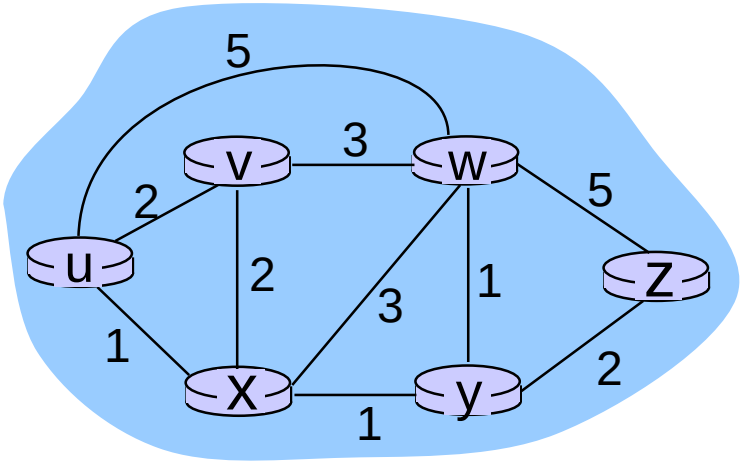


graphe: G = (N,E)

N = ensemble des routeurs= { u, v, w, x, y, z }

E = ensemble des liens={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Graphe: une abstraction



c(x,x') = coût du lien (x,x')
ex., c(w,z) = 5

Les coûts peuvent être égaux (par ex. = 1) ou une fonction de la bande passante du lien ou de sa congestion

coût du chemin $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Question: Quel le chemin le moins onéreux entre u et z ?
Algorithme de routage : algorithme qui trouve ce chemin

Classification des algorithmes de routage

Q: global ou décentralisé?

global:

- tous les routeurs connaissent tout sur les liens et la topologie
- algorithmes par état de lien

décentralisé:

- les routeurs connaissent seulement les états des voisins
- processus itératif, échange avec les voisins
- algorithmes à vecteur de distance

Q: statique ou dynamique?

statique:

- les routes changent peu souvent

dynamique:

- les routes changent souvent
  - mise à jour périodique
  - en réponse au changement des coûts des liens

Chapitre IV (suite)

4.4 algorithmes de routage

- par états des liens
- par vecteurs de distance

Un algorithme par état de lien

algorithme de Dijkstra

- connait la topologie et tous les coûts
  - par une diffusion des états des liens
  - Tous les nœuds possèdent la même information
- itératif: après k itérations, le routeur source détermine les chemins les moins onéreux vers k destinations
  - construit les tables de transfert

notation:

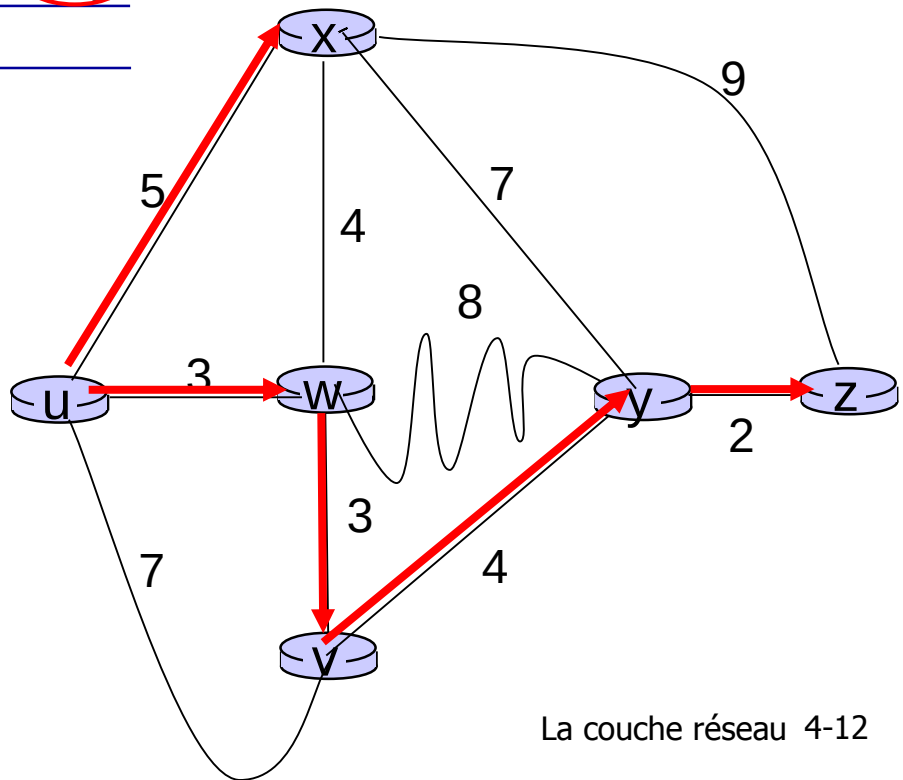
- c(x,y): coût du lien entre x et y; = ∞ si non-voisins
- D(v): coût actuel du chemin de la source à v
- p(v): prédécesseur dans le chemin entre la source et v
- N': l'ensemble des nœuds pour lesquels le chemin le moins onéreux est déjà connu

Algorithme de Dijsktra

```
1  Initialisation:
2    N' = {u}
3    pour tout nœud v
4      si v est adjacent à u
5          alors D(v) = c(u,v)
6      sinon D(v) = ∞
7
8  Faire
9     trouver w pas dans N' tel que D(w) soit minimum
10    ajouter w à N'
11    mettre à jour D(v) pour tout v adjacent à w est pas dans N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* le nouveau cout à v est soit l'ancien ou le chemin le moins
14     onéreux à w plus le cout de w à v */
15 tant que N' est différente de N
```

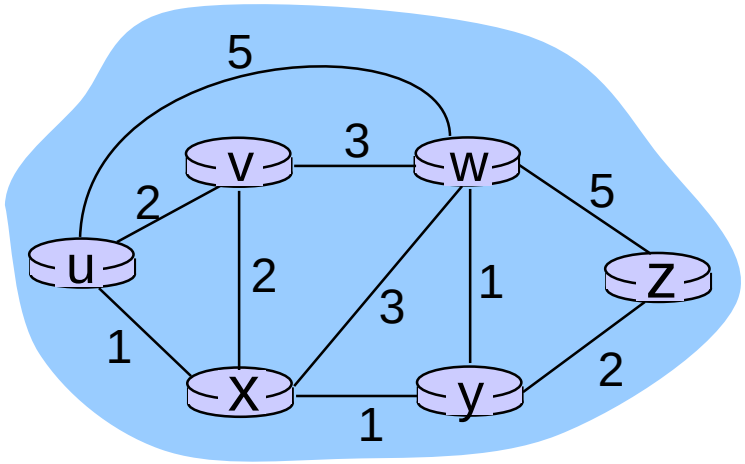
Algorithme de Dijkstra: exemple

Étape	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					



Algorithme de Dijkstra: exemple 2

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Algorithme de Dijkstra: exemple 2

arbre des chemins les moins onéreux à partir de u:

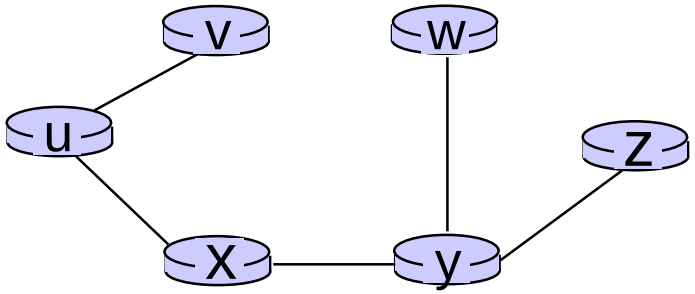


table de transfert de u:

destination	lien
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Algorithme de Dijkstra: discussion

complexité: n nœuds

- chaque itération: vérifie tous les nœuds qui ne sont pas dans N
- n(n+1)/2 comparaisons: $O(n^2)$
- une implémentation plus efficace est possible: O(nlogn)

Chapitre IV (suite)

4.4 algorithmes de routage

- par états des liens
- par vecteurs de distance

Algorithme à vecteur de distance

équation de Bellman-Ford (programmation dynamique)

soit

$d_x(y)$:= coût du chemin le moins onéreux de x à y

alors

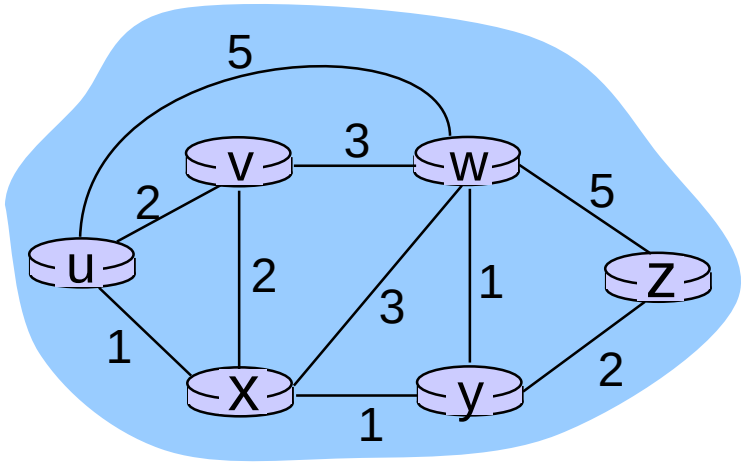
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

coût du voisin v vers y

coût jusqu'à v

min sur tous les voisins v de x

Bellman-Ford: exemple



facile, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

les équations B-F:

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

le nœud ayant le minimum est le prochain dans
le chemin le plus court, utilisé dans la table de transfert

Algorithme à vecteur de distance

- $D_x(y)$ = estimé du coût du chemin le moins onéreux de x à y
  - x maintient le vecteur des distances $\mathbf{D}_x = [D_x(y): y \in N]$
- nœud x:
  - connait le cout vers tout voisin v: $c(x,v)$
  - maintient les vecteurs des distances des voisins. pour chaque voisin v, x maintient $\mathbf{D}_v = [D_v(y): y \in N]$

Algorithme à vecteur de distance

idée:

- régulièrement, chaque nœud envoie son vecteur aux voisins
- quand x reçoit un nouveau vecteur, il met à jour son propre vecteur en utilisant:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- dans les conditions normales, l'estimé $D_x(y)$ *converge au coût réel* $d_x(y)$

Algorithme à vecteur de distance

itératif et asynchrone:

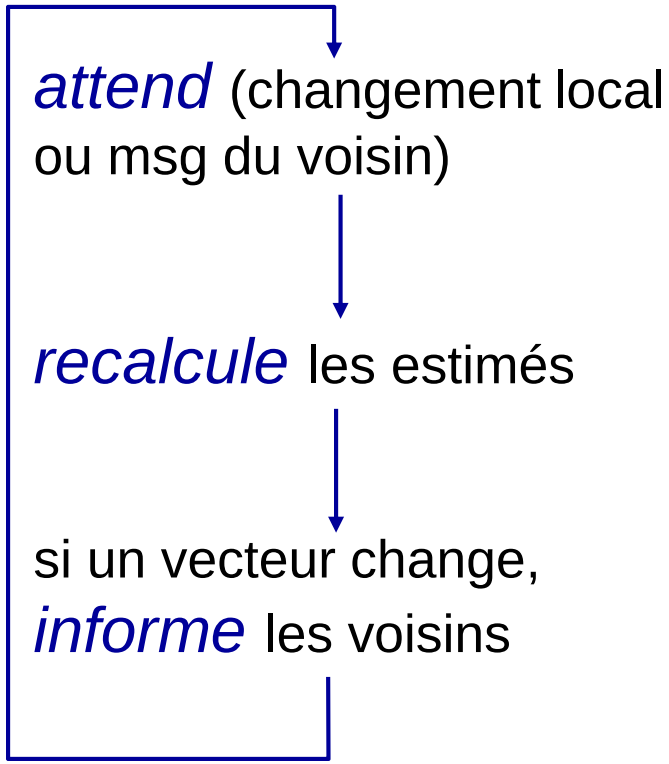
chaque itération est causée par:

- un changement de coût local
- un changement chez le voisin

distribué:

- chaque nœud informe les voisins si des changements arrivent

chaque noeud:



$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

noeud x table

de \ vers	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

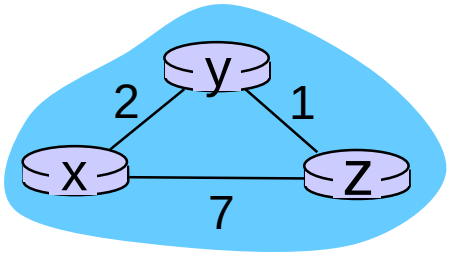
de \ vers	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

noeud y table

de \ vers	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

noeud z table

de \ vers	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0



temps

$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

noeud x table

de \ vers	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

de \ vers	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

de \ vers	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

noeud y table

de \ vers	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

de \ vers	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

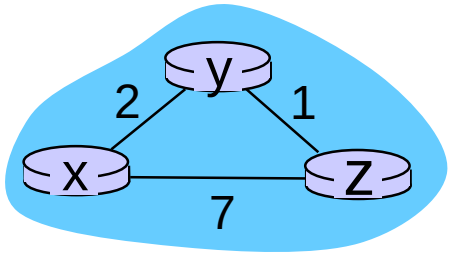
de \ vers	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

noeud z table

de \ vers	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

de \ vers	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

de \ vers	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

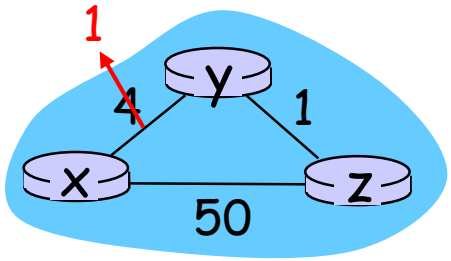


temps

Routage DV: changement d'un coût

Changement du coût d'un lien:

- un nœud détecte un changement local
- met à jour sa table et recalcule son vecteur
- si le vecteur change, informe les voisins



"bonne nouvelle voyage plus vite"

t_0 : *y* détecte le changement, met à jour son vecteur, informe ses voisins.

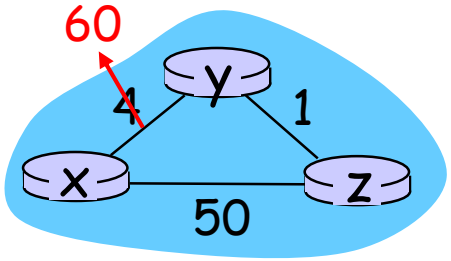
t_1 : *z* reçoit mise à jour de *y*, met à jour sa table, calcule le chemin vers *x* , envoie son vecteur aux voisins.

t_2 : *y* reçoit mise à jour de z, met à jour sa table. *la table de y ne change pas*, alors *y* n'envoie plus rien à *z*.

Routage DV: changement d'un coût

Changement du coût d'un lien:

- un nœud détecte un changement local
- *mauvaise nouvelle voyage lentement* – problème "count to infinity"!
- 44 itérations avant que l'algorithme s'arrête



poisoned reverse:

- si Z passe par Y pour atteindre X :
  - Z informe Y que sa distance à X est infinie (alors Y n'utilise pas X pour atteindre Z)

Comparaison des algorithmes

échange des msgs

- ***LS:*** avec n nœuds, E liens, O(nE) msgs envoyés
- ***DV:*** échange uniquement entre voisins
  - temps de convergence variant

convergence

- ***LS:*** $O(n^2)$ et O(nE) msgs
- ***DV:*** temps de convergence variant
  - boucle de routage
  - problème de count-to-infinity

robustesse: réponse au mauvais fonctionnement des routeurs?

LS:

- nœud peut émettre des coûts erronés
- chaque nœud calcule seulement sa propre table

DV:

- nœud peut émettre des coûts erronés
- chaque nœud utilise les tables des autres
  - propagation d'erreurs

EXEMPLE – ALGORITHME BELLMAN-FORD

Exemple

1 saut

Table for A		
Dst	Cst	Hop
A	0	A
B	4	B
C	∞	–
D	∞	–
E	2	E
F	6	F

Table for B		
Dst	Cst	Hop
A	4	A
B	0	B
C	∞	–
D	3	D
E	∞	–
F	1	F

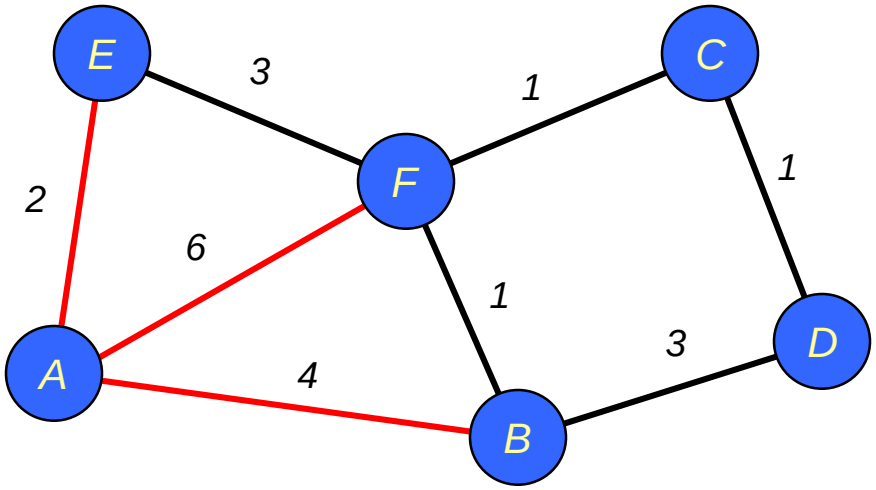


Table for C		
Dst	Cst	Hop
A	∞	–
B	∞	–
C	0	C
D	1	D
E	∞	–
F	1	F

Table for D		
Dst	Cst	Hop
A	∞	–
B	3	B
C	1	C
D	0	D
E	∞	–
F	∞	–

Table for E		
Dst	Cst	Hop
A	2	A
B	∞	–
C	∞	–
D	∞	–
E	0	E
F	3	F

Table for F		
Dst	Cst	Hop
A	6	A
B	1	B
C	1	C
D	∞	–
E	3	E
F	0	F

Exemple

2 sauts

Table for A		
Dst	Cst	Hop
A	0	A
B	4	B
C	7	F
D	7	B
E	2	E
F	5	E

Table for B		
Dst	Cst	Hop
A	4	A
B	0	B
C	2	F
D	3	D
E	4	F
F	1	F

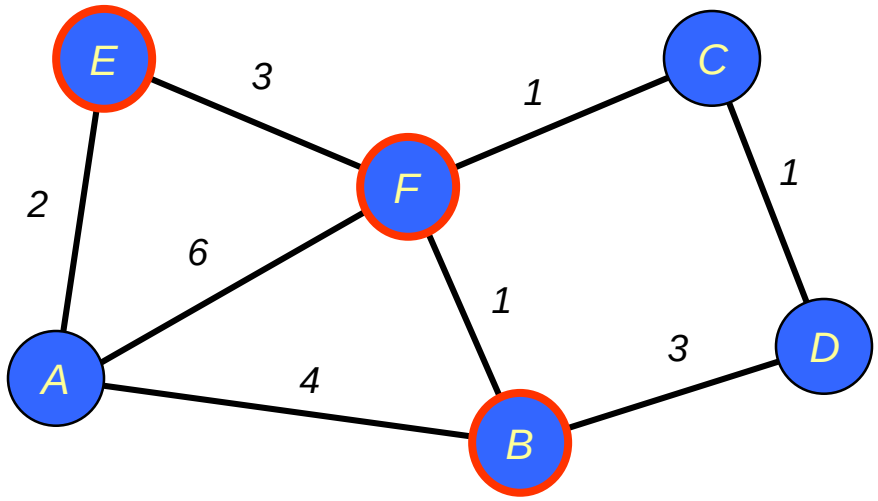


Table for C		
Dst	Cst	Hop
A	7	F
B	2	F
C	0	C
D	1	D
E	4	F
F	1	F

Table for D		
Dst	Cst	Hop
A	7	B
B	3	B
C	1	C
D	0	D
E	∞	–
F	2	C

Table for E		
Dst	Cst	Hop
A	2	A
B	4	F
C	4	F
D	∞	–
E	0	E
F	3	F

Table for F		
Dst	Cst	Hop
A	5	B
B	1	B
C	1	C
D	2	C
E	3	E
F	0	F

Exemple

3 sauts

Table for A		
Dst	Cst	Hop
A	0	A
B	4	B
C	6	E
D	7	B
E	2	E
F	5	E

Table for B		
Dst	Cst	Hop
A	4	A
B	0	B
C	2	F
D	3	D
E	4	F
F	1	F

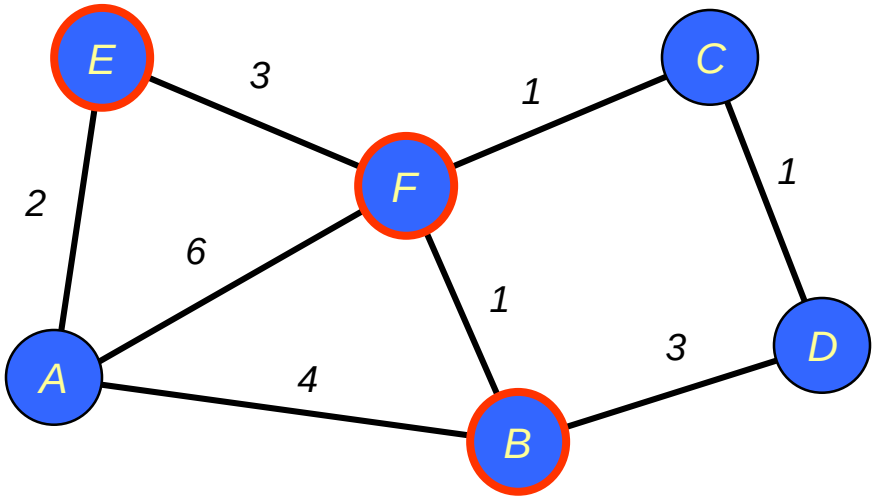


Table for C		
Dst	Cst	Hop
A	6	F
B	2	F
C	0	C
D	1	D
E	4	F
F	1	F

Table for D		
Dst	Cst	Hop
A	7	B
B	3	B
C	1	C
D	0	D
E	5	C
F	2	C

Table for E		
Dst	Cst	Hop
A	2	A
B	4	F
C	4	F
D	5	F
E	0	E
F	3	F

Table for F		
Dst	Cst	Hop
A	5	B
B	1	B
C	1	C
D	2	C
E	3	E
F	0	F