Algorithmes distribués de plus court chemin.

Johanne Cohen[1]

[1]LORIA/CNRS, Nancy, France.

Références

Jim Kurose, Keith Ross
Computer Networking, a top down approach featuring the Internet (3rd edition)
publié par Addison-Wesley Longman

Christian Huitema
le routage dans l'Internet
publié aux éditions Eyrolles.

Le routage dans le réseau

Objectif :
Mécanisme qui permet un acheminement de données entre une source et une destination :

indication de la route à un paquet dans un réseau.

Dans quelle couche ? la couche réseau.

- Détermination des routes entre la source et la destination
- Aiguillage au sein d'un routeur : Sélection d'un port de sortie pour un paquet en fonction d'un port d'entrée.

Réseau utilisant les datagrames (le modèle Internet)

- Aucun établissement d'appel dans la couche réseau routeurs : aucun état sur les connections.
- Chaque paquet est achéminé de façon indépendante vers la destination ID.

Analogie : avec le service postale.

Le routage

- Protocole de routage : détermine le bon chemin (de routeurs) entre la source et la destination.
- Réseau = graphe pour les algorithmes de routages :
  - routeurs = sommets
  - liens physiques = arêtes
  - Coût = charge, délai, congestion

Bon chemin = plus court chemin en considérant une fonction de coût

Classification des protocoles de routage

1. Informations centralisées ou réparties
   1.1 Centralisées : Les routeurs connaissent la topologie complète,
   Link-State Routing Algorithm
   1.2 Réparties
   - Les routeurs ont une vue locale réseau.
   - échange d'information entre voisins.

   Distance-Vector Routing Algorithm
2. Routages statiques ou dynamiques
   - Statique : les routes changent lentement
   - Dynamique : les routes changent rapidement (Périodiquement ou lors d'une modification de coût).

Plan

RIP : Routing Information Protocol.

OSPF : Open Shortest Path First

Plan

RIP : Routing Information Protocol.

OSPF : Open Shortest Path First

Présentation

Le protocole de routage RIP est

- un protocole base sur les vecteurs de distances
- une version distribuée de l'algorithme de Bellman et Ford.
- décrit dans le `RFC 1058` (Request for Comments : 1058)

Pour plus d'information, voir
http : //www.freesoft.org/CIE/RFC/1058/index.htm

Survol

- Itératif, asynchrone : chaque itération locale est déclenchée par :
  - un changement local de coût de liens
  - Une réception de messages d'un voisin signifiant un changement de coût de chemin.
- Réparti : Chaque noeud informe son voisin seulement quand un coût d'un chemin change. (notification faite quand c'est nécessaire)

Algorithme centralisé de Bellman-Ford

Entrée : $\left\{ \begin{array}{l} \text{Un graphe } G = (V, E) \text{ avec une source } s \\ \text{Une fonction de poids } w : E \rightarrow R^+ \end{array} \right.$

Sortie : $\left\{ \begin{array}{l} \text{Un vecteur distance } d \\ \text{Une fonction père } \pi : V \rightarrow V \end{array} \right.$

1. Initialisation de la source s
   1.1 $d[s] \leftarrow 0$; $\pi[s] \leftarrow s$
   1.2 Pour chaque sommet v de V faire $\left\{ \begin{array}{l} d[v] \leftarrow \infty \\ \pi(v) \leftarrow NIL \end{array} \right.$
2. Pour $i \leftarrow 1$ à $|V| - 1$ faire
   2.1 Pour chaque arête $(u, v) \in E$ faire
   Si $(d[v] > d[u] + w(u, v))$ alors $\left\{ \begin{array}{l} d[v] \leftarrow d[u] + w(u, v) \\ \pi(v) \leftarrow u \end{array} \right.$
3. retourner d et π

Algorithme centralisé de Bellman-Ford

Entrée : $\left\{ \begin{array}{l} \text{Un graphe } G = (V, E) \text{ avec une source } s \\ \text{Une fonction de poids } w : E \rightarrow R^+ \end{array} \right.$

Sortie : $\left\{ \begin{array}{l} \text{Un vecteur distance } d \\ \text{Une fonction père } \pi : V \rightarrow V \end{array} \right.$

1. Initialisation de la source s
   1.1 $d[s] \leftarrow 0$; $\pi[s] \leftarrow s$
   1.2 Pour chaque sommet v de V faire $\left\{ \begin{array}{l} d[v] \leftarrow \infty \\ \pi(v) \leftarrow NIL \end{array} \right.$
2. Pour $i \leftarrow 1$ à $|V| - 1$ faire
   2.1 Pour chaque arête $(u, v) \in E$ faire
   Si $(d[v] > d[u] + w(u, v))$ alors $\left\{ \begin{array}{l} d[v] \leftarrow d[u] + w(u, v) \\ \pi(v) \leftarrow u \end{array} \right.$
3. retourner d et π

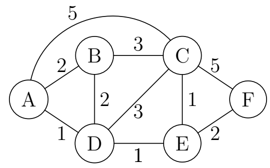
Algorithme centralisé de Bellman-Ford

Entrée : $\left\{ \begin{array}{l} \text{Un graphe } G = (V, E) \text{ avec une source } s \\ \text{Une fonction de poids } w : E \rightarrow R^+ \end{array} \right.$

Sortie : $\left\{ \begin{array}{l} \text{Un vecteur distance } d \\ \text{Une fonction père } \pi : V \rightarrow V \end{array} \right.$

1. Initialisation de la source s
   1.1 $d[s] \leftarrow 0$; $\pi[s] \leftarrow s$
   1.2 Pour chaque sommet v de V faire $\left\{ \begin{array}{l} d[v] \leftarrow \infty \\ \pi(v) \leftarrow NIL \end{array} \right.$
2. Pour $i \leftarrow 1$ à $|V| - 1$ faire
   2.1 Pour chaque arête $(u, v) \in E$ faire
   Si $(d[v] > d[u] + w(u, v))$ alors $\left\{ \begin{array}{r} d[v] \leftarrow d[u] + w(u, v) \\ \pi(v) \leftarrow u \end{array} \right.$
3. retourner d et π

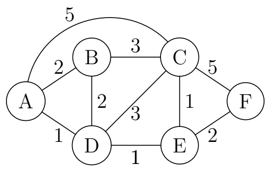
Exemple



	A	B	C	D	E	F
init	$(0, A)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$
1	$(0, A)$	$(2, A)$	$(5, A)$	$(1, A)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$
2	$(0, A)$	$(2, A)$	$(4, D)$	$(1, A)$	$(2, D)$	$(10, C)$
3	$(0, A)$	$(2, A)$	$(3, E)$	$(1, A)$	$(2, D)$	$(4, E)$

les couples correspondent à $(d(.), \pi(.))$

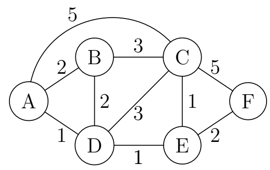
Exemple



	A	B	C	D	E	F
init	$(0, A)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$
1	$(0, A)$	$(2, A)$	$(5, A)$	$(1, A)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$
2	$(0, A)$	$(2, A)$	$(4, D)$	$(1, A)$	$(2, D)$	$(10, C)$
3	$(0, A)$	$(2, A)$	$(3, E)$	$(1, A)$	$(2, D)$	$(4, E)$

les couples correspondent à $(d(.), \pi(.))$

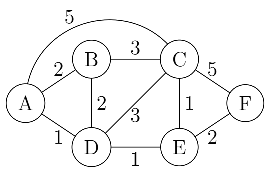
Exemple



	A	B	C	D	E	F
init	$(0, A)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$
1	$(0, A)$	$(2, A)$	$(5, A)$	$(1, A)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$
2	$(0, A)$	$(2, A)$	$(4, D)$	$(1, A)$	$(2, D)$	$(10, C)$
3	$(0, A)$	$(2, A)$	$(3, E)$	$(1, A)$	$(2, D)$	$(4, E)$

les couples correspondent à $(d(.), \pi(.))$

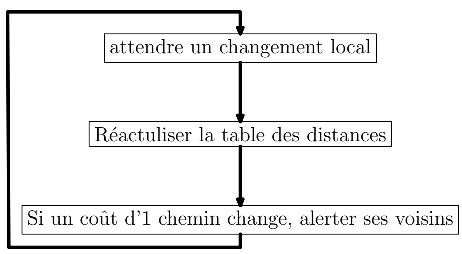
Exemple



	A	B	C	D	E	F
init	$(0, A)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$
1	$(0, A)$	$(2, A)$	$(5, A)$	$(1, A)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$
2	$(0, A)$	$(2, A)$	$(4, D)$	$(1, A)$	$(2, D)$	$(10, C)$
3	$(0, A)$	$(2, A)$	$(3, E)$	$(1, A)$	$(2, D)$	$(4, E)$

les couples correspondent à $(d(.), \pi(.))$

Comportement local d'un noeud



Version distribuée : au niveau du noeud courant i

- Une table avec une entrée pour chaque destination possible du système contenant

  (DISTANCE, LE PROCHAIN SAUT)

  Remarque : il devrait y avoir une entrée pour l'entité elle-même, de métrique 0.
- Périodiquement, envoyez une mise à jour de routage à chaque voisin. La mise à jour est un groupe de messages contenant toute l'information de la table de routage.
- lors d'un message de mise à jour de routage parvient, une comparaison entre les *nouvelles* distances et les distances dans sa table de routage.

Version distribuée : au niveau du noeud courant i

- Une table d avec une entrée pour chaque destination possible du système contenant

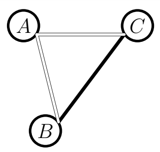
  (DESTINATION, LE PROCHAIN SAUT, DISTANCE)
- Initialisation $\rightarrow$
  1. insérer $(i, local, 0)$ dans la table de routage
  2. envoyer le vecteur de distance $(i, 0)$ à tous ses voisins
- Réception d'un vecteur de distance sur le lien ℓ $\rightarrow$
  - pour chacun (DEST,DIST) du vecteur de distance faire,
    1. si DEST est inconnue, alors insérer une nouvelle ligne

       (DEST, ℓ ,DIST +COÛT(ℓ))
    2. sinon, si $d(dest) > dist + cout(\ell)$ alors

       modifier la table : (DEST, ℓ ,DISTANCE +COÛT(ℓ))

       envoyer le vecteur de distance à tous ses voisins

Exemple



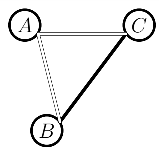
d^A	lien	coût
A	local	0

d^C	lien	coût
C	local	0

d^B	lien	coût
B	local	0

Initialisation
Construction des tables de routage

Exemple



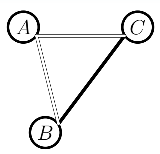
d^A	lien	coût
A	local	0

d^C	lien	coût
C	local	0
B	(C-B)	1

d^B	lien	coût
B	local	0

Envoi par B du vecteur de distance $< (B, 0) >$
réception par C

Exemple



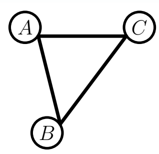
d^A	lien	coût
A	local	0

d^C	lien	coût
C	local	0
B	(C-B)	1

d^B	lien	coût
B	local	0
C	(B-C)	1

Envoi par C du vecteur de distance $< (C, 0), (B, 1) >$
réception par B

Exemple



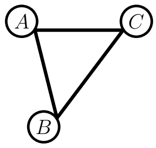
d^A	lien	coût
A	local	0
B	(A-B)	1
C	(A-B)	2

d^C	lien	coût
C	local	0
B	(C-B)	1

d^B	lien	coût
B	local	0
C	(B-C)	1

Envoi par B du vecteur de distance $< (B, 0), (C, 1) >$
réception par C et A

Exemple



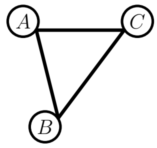
d^A	lien	coût
A	local	0
B	(A-B)	1
C	(A-B)	2

d^C	lien	coût
C	local	0
B	(C-B)	1
A	(C-A)	1

d^B	lien	coût
B	local	0
C	(B-C)	1
A	(B-A)	1

Envoi par A du vecteur de distance $<(A,0),(B,1),(C,2)>$
réception par C et B

Exemple



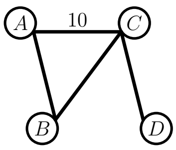
d^A	lien	coût
A	local	0
B	(A-B)	1
C	A-C	1

d^C	lien	coût
C	local	0
B	(C-B)	1
A	(C-A)	1

d^B	lien	coût
B	local	0
C	(B-C)	1
A	(B-A)	1

Envoi par C du vecteur de distance $<(C,0),(B,1)>$
réception par B et A

Exemple : panne du lien $B-C$

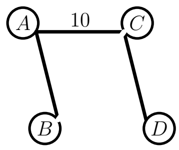


d^A	lien	coût
A	local	0
B	(A-B)	1
C	(A-B)	2
D	(A-B)	3

d^B	lien	coût
B	local	0
A	(B-A)	1
C	(B-C)	1
D	(B-C)	2

d^C	lien	coût
C	local	0
A	(C-B)	2
B	(C-B)	1
D	(C-D)	1

Exemple : panne du lien $B-C$



(B-C) tombe en panne :

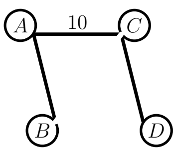
- C détecte
- il envoie le message $<(C,0),(A,inf),(B,inf),(D,1)>$

d^A	lien	coût
A	local	0
B	(A-B)	1
C	(A-B)	2
D	(A-B)	3

d^B	lien	coût
B	local	0
A	(B-A)	1
C	(B-C)	1
D	(B-C)	2

d^C	lien	coût
C	local	0
A	??	∞
B	??	∞
D	(C-D)	1

Exemple : panne du lien $B-C$



A gère du msg avant la réaction de B
$<(C,0),(A,inf),(B,inf),(D,1)>$

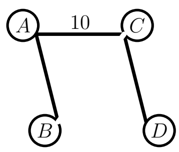
- Aucune modification de la table
- émission du msg $<(A,0),(B,1),(C,2),(D,3)>$

d^A	lien	coût
A	local	0
B	(A-B)	1
C	(A-B)	2
D	(A-B)	3

d^B	lien	coût
B	local	0
A	(B-A)	1
C	(B-C)	1
D	(B-C)	2

d^C	lien	coût
C	local	0
A	??	∞
B	??	∞
D	(C-D)	1

Exemple : panne du lien $B-C$



Gestion du msg.

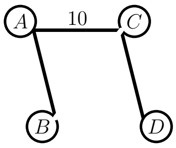
- $<(A,0),(B,1),(C,2),(D,3)>$ par C

d^A	lien	coût
A	local	0
B	(A-B)	1
C	(A-B)	2
D	(A-B)	3

d^B	lien	coût
B	local	0
A	(B-A)	1
C	(B-C)	1
D	(B-C)	2

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

Exemple : panne du lien $B - C$



Gestion du msg.

- $< (A,0),(B,1),(C,2),(D,3) >$ par C

Émission du msg. par C

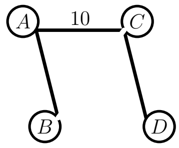
- $< (C,0),(A,10),(C,11),(D,1) >$

d^A	lien	coût
A	local	0
B	(A-B)	1
C	(A-B)	2
D	(A-B)	3

d^B	lien	coût
B	local	0
A	(B-A)	1
C	(B-C)	1
D	(B-C)	2

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

Exemple : détection de la panne par B



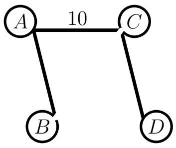
- B détecte la panne
- Et B envoie le message
  $< (B,0),(A,1),(C,inf),(D,inf) >$

d^A	lien	coût
A	local	0
B	(A-B)	1
C	(A-B)	2
D	(A-B)	3

d^B	lien	coût
B	local	0
A	(B-A)	1
C	(B-C)	1
D	(B-C)	2

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

Exemple : détection de la panne par B



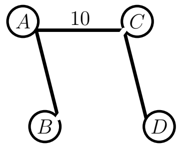
- B détecte la panne
- Et B envoie le message
  $< (B,0),(A,1),(C,inf),(D,inf) >$

d^A	lien	coût
A	local	0
B	(A-B)	1
C	(A-B)	2
D	(A-B)	3

d^B	lien	coût
B	local	0
A	(B-A)	1
C	? ?	∞
D	? ?	∞

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

Exemple : détection de la panne par B



Gestion simultanée des msg.

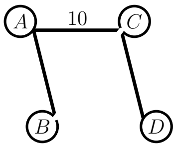
- $< (B,0),(A,1),(C,inf),(D,inf) >$ par A
- $< (A,0),(B,1),(C,2),(D,3) >$ par B

d^A	lien	coût
A	local	0
B	(A-B)	1
C	(A-B)	2
D	(A-B)	3

d^B	lien	coût
B	local	0
A	(B-A)	1
C	? ?	∞
D	? ?	∞

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

Exemple : détection de la panne par B



Gestion simultanée des msg.

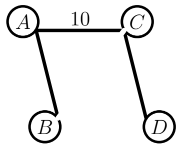
- $< (B,0),(A,1),(C,inf),(D,inf) >$ par A
- $< (A,0),(B,1),(C,2),(D,3) >$ par B

d^A	lien	coût
A	local	0
B	(A-B)	1
C	? ?	∞
D	? ?	∞

d^B	lien	coût
B	local	0
A	(B-A)	1
C	(B-A)	3
D	(B-A)	4

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

Exemple : lente convergence



Émission simultanée des msg.

- $< (A,0),(B,1),(C,inf),(D,inf) >$ par A
- $< (A,0),(B,1),(C,3),(D,4) >$ par B

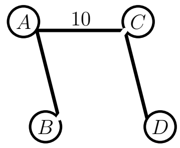
d^A	lien	coût
A	local	0
B	(A-B)	1
C	? ?	∞
D	? ?	∞

d^B	lien	coût
B	local	0
A	(B-A)	1
C	(B-A)	3
D	(B-A)	4

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

1

Exemple : lente convergence



Réception simultanée des msg.

- $<(A,0),(B,1),(C,3),(D,4)>$ par A
- $<(A,0),(B,1),(C,inf),(D,inf)>$ par B

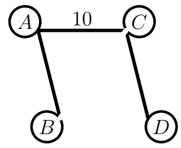
d^A	lien	coût
A	local	0
B	(A-B)	1
C	??	∞
D	??	∞

d^B	lien	coût
B	local	0
A	(B-A)	1
C	(B-A)	3
D	(B-A)	4

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

2

Exemple : lente convergence



Réception simultanée des msg.

- $<(A,0),(B,1),(C,3),(D,4)>$ par A
- $<(A,0),(B,1),(C,inf),(D,inf)>$ par B

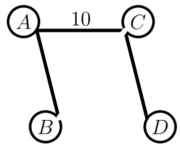
d^A	lien	coût
A	local	0
B	(A-B)	1
C	(A-B)	4
D	(A-B)	5

d^B	lien	coût
B	local	0
A	(B-A)	1
C	??	∞
D	??	∞

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

3

Exemple : lente convergence



Émission simultanée des msg.

- $<(A,0),(B,1),(C,4),(D,5)>$ par A
- $<(A,0),(B,1),(C,inf),(D,inf)>$ par B

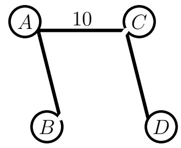
d^A	lien	coût
A	local	0
B	(A-B)	1
C	(A-B)	4
D	(A-B)	5

d^B	lien	coût
B	local	0
A	(B-A)	1
C	??	∞
D	??	∞

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

4

Exemple : lente convergence



Réception simultanée des msg.

- $<(A,0),(B,1),(C,inf),(D,inf)>$ par A
- $<(A,0),(B,1),(C,4),(D,5)>$ par B

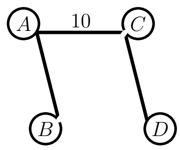
d^A	lien	coût
A	local	0
B	(A-B)	1
C	(A-B)	4
D	(A-B)	5

d^B	lien	coût
B	local	0
A	(B-A)	1
C	??	∞
D	??	∞

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

5

Exemple : lente convergence



Réception simultanée des msg.

- $<(A,0),(B,1),(C,inf),(D,inf)>$ par A
- $<(A,0),(B,1),(C,4),(D,5)>$ par B

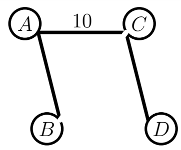
d^A	lien	coût
A	local	0
B	(A-B)	1
C	??	∞
D	??	∞

d^B	lien	coût
B	local	0
A	(B-A)	1
C	(B-A)	5
D	(B-A)	6

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

6

Exemple : lente convergence



Émission simultanée des msg.

- $<(A,0),(B,1),(C,inf),(D,inf)>$ par A
- $<(A,0),(B,1),(C,5),(D,6)>$ par B

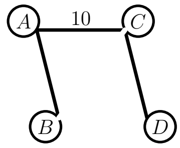
d^A	lien	coût
A	local	0
B	(A-B)	1
C	??	∞
D	??	∞

d^B	lien	coût
B	local	0
A	(B-A)	1
C	(B-A)	5
D	(B-A)	6

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

7

Exemple : lente convergence



lente convergence :

- les distances de A et B par rapport à C et à D vont augmenter lentement
- et ainsi de suite....
- jusqu'à ce que le lien (A-C) (par les messages de C) soit pris en compte.

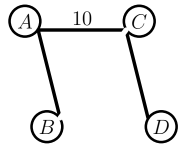
d^A	lien	coût
A	local	0
B	(A-B)	1
C	? ?	∞
D	? ?	∞

d^B	lien	coût
B	local	0
A	(B-A)	1
C	(B-A)	5
D	(B-A)	6

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

8

Exemple : lente convergence



lente convergence :

- les distances de A et B par rapport à C et à D vont augmenter lentement
- et ainsi de suite....
- jusqu'à ce que le lien (A-C) (par les messages de C) soit pris en compte.

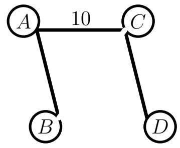
d^A	lien	coût
A	local	0
B	(A-B)	1
C	(A-B)	6
D	(A-B)	7

d^B	lien	coût
B	local	0
A	(B-A)	1
C	? ?	∞
D	? ?	∞

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

9

Exemple : lente convergence



lente convergence :

- les distances de A et B par rapport à C et à D vont augmenter lentement
- et ainsi de suite....
- jusqu'à ce que le lien (A-C) (par les messages de C) soit pris en compte.

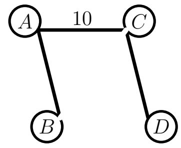
d^A	lien	coût
A	local	0
B	(A-B)	1
C	? ?	∞
D	? ?	∞

d^B	lien	coût
B	local	0
A	(B-A)	1
C	(B-A)	7
D	(B-A)	8

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

10

Exemple : lente convergence



lente convergence :

- les distances de A et B par rapport à C et à D vont augmenter lentement
- et ainsi de suite....
- jusqu'à ce que le lien (A-C) (par les messages de C) soit pris en compte.

d^A	lien	coût
A	local	0
B	(A-B)	1
C	(A-B)	12
D	(A-B)	13

d^B	lien	coût
B	local	0
A	(B-A)	1
C	? ?	∞
D	? ?	∞

d^C	lien	coût
C	local	0
A	(C-A)	10
B	(C-A)	11
D	(C-D)	1

11

Horizon partagé

- Pourquoi le système converge lentement vers un état correct ?
  Car
  - entretien mutuel de l'erreur par A et par B
  - chacun prétend être capable de joindre C via l'autre.
- Comment résoudre ce problème ?
  - Par le mécanisme d'horizon partagé (deux versions)

Horizon partagé

- Pourquoi le système converge lentement vers un état correct ?
  Car
  - entretien mutuel de l'erreur par A et par B
  - chacun prétend être capable de joindre C via l'autre.
- Comment résoudre ce problème ?
  - Par le mécanisme d'horizon partagé (deux versions)

Horizon partagé

- Pourquoi le système converge lentement vers un état correct ? Car
  - entretien mutuel de l'erreur par A et par B
  - chacun prétend être capable de joindre *C* via l'autre.
- Comment résoudre ce problème ?
  - Par le mécanisme d'horizon partagé (deux versions)
    - suppression des informations provenant du destinataire dans les messages de vecteurs de distances qui lui sont destinés.
    - insertion dans le vecteur de distance d'informations indiquant que l'on ne peut pas joindre une destination si la route passe par le destinataire de ce message.

Horizon partagé

- Pourquoi le système converge lentement vers un état correct ? Car
  - entretien mutuel de l'erreur par A et par B
  - chacun prétend être capable de joindre *C* via l'autre.
- Comment résoudre ce problème ?
  - Par le mécanisme d'horizon partagé (deux versions)
    - suppression des informations provenant du destinataire dans les messages de vecteurs de distances qui lui sont destinés.
    - insertion dans le vecteur de distance d'informations indiquant que l'on ne peut pas joindre une destination si la route passe par le destinataire de ce message.

Horizon partagé

- Pourquoi le système converge lentement vers un état correct ? Car
  - entretien mutuel de l'erreur par A et par B
  - chacun prétend être capable de joindre *C* via l'autre.
- Comment résoudre ce problème ?
  - Par le mécanisme d'horizon partagé (deux versions)
    - suppression des informations provenant du destinataire dans les messages de vecteurs de distances qui lui sont destinés.
    - insertion dans le vecteur de distance d'informations indiquant que l'on ne peut pas joindre une destination si la route passe par le destinataire de ce message.

Horizon partagé

- Dans notre exemple : la convergence est rapide.
- Mais ce mécanisme ne fonctionne pas si le cycle de noeuds impliqués est ≥ 3.

Limite

- protocole très simple mais inadapté pour des réseaux complexes
- risque d'erreurs si le taux de pannes est important.

Plan

RIP : Routing Information Protocol.

OSPF : Open Shortest Path First

Présentation

Basé sur l'algorithme de Dijkstra

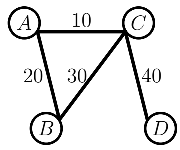
- la topologie du réseau et les coût des liens de communications sont connus par tous les noeuds
  - grâce à des mises des états de liens (diffusion)
  - tous les noeuds ont la même information
- les plus court chemins du noeud courant vers les autres sont calculés

résultat = une table de routage incluant les chemins et les ports permettant d'atteindre chaque noeud

Gestion de la table des liens

La topologie est stoquée sous forme de table.

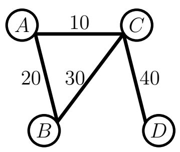
De	Vers	Liens	Distance
A	B	20	1
A	C	10	1
B	A	20	1
B	C	30	1
C	A	10	1
C	B	30	1
C	D	40	1
D	C	40	1



Gestion de la table des liens

La topologie est stoquée sous forme de table.

De	Vers	Liens	Distance
A	B	20	1
A	C	10	1
B	A	20	1
B	C	30	1
C	A	10	1
C	B	30	1
C	D	40	1
D	C	40	1



Comment la construire ?

Connaissance des voisins

Par le protocole HELLO

- Chaque noeud envoie périodiquement sur chaque lien un message HELLO contenant
  - son identifiant
  - une liste de noeuds (voisins) dont le noeud a reçu un message HELLO

Objectif :

- connaître tous ses voisins
- vérifier l'état du lien de communication

Conséquence : utilisation des connexions *full-duplex*

Connaissance de la topologie

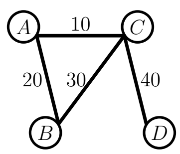
Par un protocole INONDATION (*Flooding* en anglais)

Objectif :

- connaître toute la topologie
- informer tout le réseau d'un changement local afin que tous les noeuds adaptent ces routes.

Précaution à prendre :

Si le lien 30 tombe en panne,
Après détection, *C* alertera ses voisins par l'envoi du msg <C,B,20, DIST= INFINI >



<C,B,20, DIST= INFINI,NBRE=2 >

Connaissance de la topologie

Par un protocole INONDATION (*Flooding* en anglais)

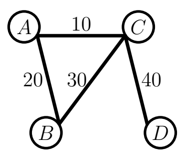
Objectif :

- connaître toute la topologie
- informer tout le réseau d'un changement local afin que tous les noeuds adaptent ces routes.

Précaution à prendre :

Si le lien 30 tombe en panne,
Après détection, *C* alertera ses voisins par l'envoi du msg <C,B,20, DIST= INFINI >
ATTENTION : il risque d'avoir des vieux messages transitant dans le réseau

Nécessité de datation (sur 32 bits)



<C,B,20, DIST= INFINI,NBRE=2 >

protocole d'INNODATION

Lors d'une réception de message <A,B,LIEN ℓ, DISTANCE, NBR> :

1. Regarder si l'information est dans la base.
2. Si ce n'est pas le cas, alors

   stoquer dans la base puis diffuser cette information
3. Sinon, si le nombre dans la base < que celui du message, alors

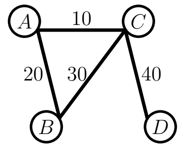
   réactualiser la base, puis diffuser cette information
4. Sinon, si le nombre dans la base > que celui du message, alors

   renvoyer l'information stoquée dans la base au destinataire du message.

Retour à l'exemple

Avant la panne du lien 30

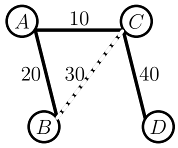
De	Vers	Liens	Distance	Nbr
A	B	20	1	1
A	C	10	1	1
B	A	20	1	1
B	C	30	1	1
C	A	10	1	1
C	B	30	1	1
C	D	40	1	1
D	C	40	1	1



Retour à l'exemple

Après la panne du lien 30

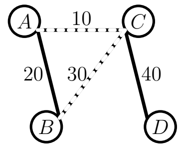
De	Vers	Liens	Distance	Nbr
A	B	20	1	1
A	C	10	1	1
B	A	20	1	1
B	C	30	∞	2
C	A	10	1	1
C	B	30	∞	2
C	D	40	1	1
D	C	40	1	1



Retour à l'exemple

Après la panne du lien 10

De	Vers	Liens	Distance	Nbr
A	B	20	1	1
A	C	10	∞	2
B	A	20	1	1
B	C	30	∞	2
C	A	10	∞	2
C	B	30	∞	2
C	D	40	1	1
D	C	40	1	1



2 réseaux distincts évoluant séparément.
Que se passe-t-il lors d'une nouvelle fusion ?

Conservation de la connexité

Lors d'une fusion : rétablissement d'un lien connectant (A-C)

- Synchronisation des deux bases par l'échange complète des deux bases

  sur chaque information : conservation de l'information la plus récente.
  insérer les nouvelles informations
- Diffusion des nouvelles informations ou des modifications par le protocole d'inondation.

Comportement d'un noeud

1. Connaissance d'un réseau
   - Protocole HELLO
   - Protocole d'INONDATION
2. Calcul de la table de routage
   - algorithme de Dijkstra sur la vue locale du réseau.
   -

Algorithme centralisé de Dijkstra

Entrée : $\left\{ \begin{array}{l} \text{Un graphe } G = (V, E) \text{ avec une source } s \\ \text{Une fonction de poids } w : E \to R^+ \end{array} \right.$

Sortie : $\left\{ \begin{array}{l} \text{Un vecteur distance } d \\ \text{Une fonction père } \pi : V \to V \end{array} \right.$

1. Initialisation de la source s
   - 1.1 $d[s] \leftarrow 0$; $\pi[s] \leftarrow s$
   - 1.2 Pour chaque sommet v de V faire $\left\{ \begin{array}{l} \pi(v) \leftarrow NIL \\ d(v) \leftarrow \infty^+ \end{array} \right.$
2. $\mathcal{Q} \leftarrow V$; $\mathcal{S} \leftarrow \emptyset$;
3. Tant que $(\mathcal{Q} \neq \emptyset)$ faire
   - 3.1 $u \leftarrow$ EXTRAIRE-LE-MINIMUM$(\mathcal{Q}, d)$;
   - 3.2 $\mathcal{S} \leftarrow \mathcal{S} \cup \{u\}$;
   - 3.3 Pour chaque sommet v voisin de u faire

     Si $(d[v] > d[u] + w(u, v))$ alors $\left\{ \begin{array}{r} d[v] \leftarrow d[u] + w(u, v) \\ \pi(v) \leftarrow u \end{array} \right.$
4. retourner d et π

Algorithme centralisé de Dijkstra

Entrée : $\left\{ \begin{array}{l} \text{Un graphe } G = (V, E) \text{ avec une source } s \\ \text{Une fonction de poids } w : E \to R^+ \end{array} \right.$

Sortie : $\left\{ \begin{array}{l} \text{Un vecteur distance } d \\ \text{Une fonction père } \pi : V \to V \end{array} \right.$

1. Initialisation de la source s
   - 1.1 $d[s] \leftarrow 0$; $\pi[s] \leftarrow s$
   - 1.2 Pour chaque sommet v de V faire $\left\{ \begin{array}{l} \pi(v) \leftarrow NIL \\ d(v) \leftarrow \infty^+ \end{array} \right.$
2. $\mathcal{Q} \leftarrow V$; $\mathcal{S} \leftarrow \emptyset$;
3. Tant que $(\mathcal{Q} \neq \emptyset)$ faire
   - 3.1 $u \leftarrow$ EXTRAIRE-LE-MINIMUM$(\mathcal{Q}, d)$;
   - 3.2 $\mathcal{S} \leftarrow \mathcal{S} \cup \{u\}$;
   - 3.3 Pour chaque sommet v voisin de u faire

     Si $(d[v] > d[u] + w(u, v))$ alors $\left\{ \begin{array}{r} d[v] \leftarrow d[u] + w(u, v) \\ \pi(v) \leftarrow u \end{array} \right.$
4. retourner d et π

Algorithme centralisé de Dijkstra

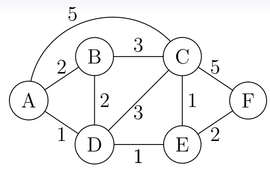
Entrée : $\left\{ \begin{array}{l} \text{Un graphe } G = (V, E) \text{ avec une source } s \\ \text{Une fonction de poids } w : E \to R^+ \end{array} \right.$

Sortie : $\left\{ \begin{array}{l} \text{Un vecteur distance } d \\ \text{Une fonction père } \pi : V \to V \end{array} \right.$

1. Initialisation de la source s
   - 1.1 $d[s] \leftarrow 0$; $\pi[s] \leftarrow s$
   - 1.2 Pour chaque sommet v de V faire $\left\{ \begin{array}{l} \pi(v) \leftarrow NIL \\ d(v) \leftarrow \infty^+ \end{array} \right.$
2. $\mathcal{Q} \leftarrow V$; $\mathcal{S} \leftarrow \emptyset$;
3. Tant que $(\mathcal{Q} \neq \emptyset)$ faire
   - 3.1 $u \leftarrow$ EXTRAIRE-LE-MINIMUM$(\mathcal{Q}, d)$;
   - 3.2 $\mathcal{S} \leftarrow \mathcal{S} \cup \{u\}$;
   - 3.3 Pour chaque sommet v voisin de u faire

     Si $(d[v] > d[u] + w(u, v))$ alors $\left\{ \begin{array}{r} d[v] \leftarrow d[u] + w(u, v) \\ \pi(v) \leftarrow u \end{array} \right.$
4. retourner d et π

Exemple

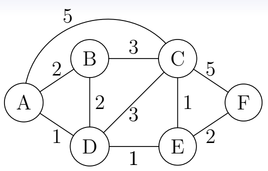


les couples correspondent à $(d(.), \pi(.))$

	$\mathcal{Q}$	A	B	C	D	E	F
I	$\{ABCDEF\}$	$(0, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$
1	$\{BCDEF\}$	$(0, A)$	$(2, A)$	$(5, A)$	$(1, A)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$
2	$\{BCEF\}$	$(0, A)$	$(2, A)$	$(4, D)$	$(1, A)$	$(2, D)$	$(\infty, \emptyset)$
3	$\{CEF\}$	$(0, A)$	$(2, A)$	$(4, D)$	$(1, A)$	$(2, D)$	$(\infty, \emptyset)$
4	$\{CF\}$	$(0, A)$	$(2, A)$	$(3, E)$	$(1, A)$	$(2, D)$	$(4, E)$
5	$\{F\}$	$(0, A)$	$(2, A)$	$(3, E)$	$(1, A)$	$(2, D)$	$(4, E)$
6	$\emptyset$	$(0, A)$	$(2, A)$	$(3, E)$	$(1, A)$	$(2, D)$	$(4, E)$

Complexité de l'algorithme : $O(|V|^2)$

Exemple

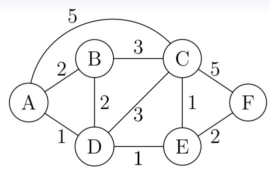


les couples correspondent à $(d(.), \pi(.))$

	$\mathcal{Q}$	A	B	C	D	E	F
I	$\{ABCDEF\}$	$(0, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$
1	$\{BCDEF\}$	$(0, A)$	$(2, A)$	$(5, A)$	$(1, A)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$
2	$\{BCEF\}$	$(0, A)$	$(2, A)$	$(4, D)$	$(1, A)$	$(2, D)$	$(\infty, \emptyset)$
3	$\{CEF\}$	$(0, A)$	$(2, A)$	$(4, D)$	$(1, A)$	$(2, D)$	$(\infty, \emptyset)$
4	$\{CF\}$	$(0, A)$	$(2, A)$	$(3, E)$	$(1, A)$	$(2, D)$	$(4, E)$
5	$\{F\}$	$(0, A)$	$(2, A)$	$(3, E)$	$(1, A)$	$(2, D)$	$(4, E)$
6	$\emptyset$	$(0, A)$	$(2, A)$	$(3, E)$	$(1, A)$	$(2, D)$	$(4, E)$

Complexité de l'algorithme : $O(|V|^2)$

Exemple

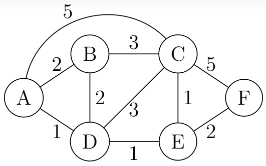


les couples correspondent à $(d(.), \pi(.))$

	$\mathcal{Q}$	A	B	C	D	E	F
I	$\{ABCDEF\}$	$(0, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$
1	$\{BCDEF\}$	$(0, A)$	$(2, A)$	$(5, A)$	$(1, A)$	$(\infty, \emptyset)$	$(\infty, \emptyset)$
2	$\{BCEF\}$	$(0, A)$	$(2, A)$	$(4, D)$	$(1, A)$	$(2, D)$	$(\infty, \emptyset)$
3	$\{CEF\}$	$(0, A)$	$(2, A)$	$(4, D)$	$(1, A)$	$(2, D)$	$(\infty, \emptyset)$
4	$\{CF\}$	$(0, A)$	$(2, A)$	$(3, E)$	$(1, A)$	$(2, D)$	$(4, E)$
5	$\{F\}$	$(0, A)$	$(2, A)$	$(3, E)$	$(1, A)$	$(2, D)$	$(4, E)$
6	$\emptyset$	$(0, A)$	$(2, A)$	$(3, E)$	$(1, A)$	$(2, D)$	$(4, E)$

Complexité de l'algorithme : $O(|V|^2)$

Exemple

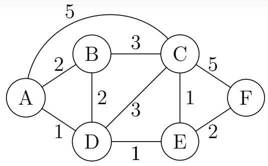


les couples correspondent à $(d(.), \pi(.))$

	Q	A	B	C	D	E	F
I	$\{ABCDEF\}$	$(0,\emptyset)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$
1	$\{BCDEF\}$	$(0,A)$	$(2,A)$	$(5,A)$	$(1,A)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$
2	$\{BCEF\}$	$(0,A)$	$(2,A)$	$(4,D)$	$(1,A)$	$(2,D)$	$(\infty,\emptyset)$
3	$\{CEF\}$	$(0,A)$	$(2,A)$	$(4,D)$	$(1,A)$	$(2,D)$	$(\infty,\emptyset)$
4	$\{CF\}$	$(0,A)$	$(2,A)$	$(3,E)$	$(1,A)$	$(2,D)$	$(4,E)$
5	$\{F\}$	$(0,A)$	$(2,A)$	$(3,E)$	$(1,A)$	$(2,D)$	$(4,E)$
6	$\emptyset$	$(0,A)$	$(2,A)$	$(3,E)$	$(1,A)$	$(2,D)$	$(4,E)$

Complexité de l'algorithme : $O(|V|^2)$

Exemple

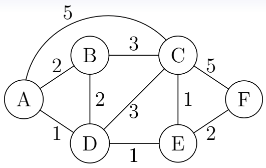


les couples correspondent à $(d(.), \pi(.))$

	Q	A	B	C	D	E	F
I	$\{ABCDEF\}$	$(0,\emptyset)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$
1	$\{BCDEF\}$	$(0,A)$	$(2,A)$	$(5,A)$	$(1,A)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$
2	$\{BCEF\}$	$(0,A)$	$(2,A)$	$(4,D)$	$(1,A)$	$(2,D)$	$(\infty,\emptyset)$
3	$\{CEF\}$	$(0,A)$	$(2,A)$	$(4,D)$	$(1,A)$	$(2,D)$	$(\infty,\emptyset)$
4	$\{CF\}$	$(0,A)$	$(2,A)$	$(3,E)$	$(1,A)$	$(2,D)$	$(4,E)$
5	$\{F\}$	$(0,A)$	$(2,A)$	$(3,E)$	$(1,A)$	$(2,D)$	$(4,E)$
6	$\emptyset$	$(0,A)$	$(2,A)$	$(3,E)$	$(1,A)$	$(2,D)$	$(4,E)$

Complexité de l'algorithme : $O(|V|^2)$

Exemple

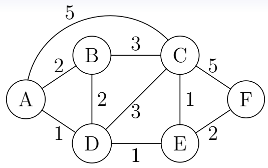


les couples correspondent à $(d(.), \pi(.))$

	Q	A	B	C	D	E	F
I	$\{ABCDEF\}$	$(0,\emptyset)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$
1	$\{BCDEF\}$	$(0,A)$	$(2,A)$	$(5,A)$	$(1,A)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$
2	$\{BCEF\}$	$(0,A)$	$(2,A)$	$(4,D)$	$(1,A)$	$(2,D)$	$(\infty,\emptyset)$
3	$\{CEF\}$	$(0,A)$	$(2,A)$	$(4,D)$	$(1,A)$	$(2,D)$	$(\infty,\emptyset)$
4	$\{CF\}$	$(0,A)$	$(2,A)$	$(3,E)$	$(1,A)$	$(2,D)$	$(4,E)$
5	$\{F\}$	$(0,A)$	$(2,A)$	$(3,E)$	$(1,A)$	$(2,D)$	$(4,E)$
6	$\emptyset$	$(0,A)$	$(2,A)$	$(3,E)$	$(1,A)$	$(2,D)$	$(4,E)$

Complexité de l'algorithme : $O(|V|^2)$

Exemple



les couples correspondent à $(d(.), \pi(.))$

	Q	A	B	C	D	E	F
I	$\{ABCDEF\}$	$(0,\emptyset)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$
1	$\{BCDEF\}$	$(0,A)$	$(2,A)$	$(5,A)$	$(1,A)$	$(\infty,\emptyset)$	$(\infty,\emptyset)$
2	$\{BCEF\}$	$(0,A)$	$(2,A)$	$(4,D)$	$(1,A)$	$(2,D)$	$(\infty,\emptyset)$
3	$\{CEF\}$	$(0,A)$	$(2,A)$	$(4,D)$	$(1,A)$	$(2,D)$	$(\infty,\emptyset)$
4	$\{CF\}$	$(0,A)$	$(2,A)$	$(3,E)$	$(1,A)$	$(2,D)$	$(4,E)$
5	$\{F\}$	$(0,A)$	$(2,A)$	$(3,E)$	$(1,A)$	$(2,D)$	$(4,E)$
6	$\emptyset$	$(0,A)$	$(2,A)$	$(3,E)$	$(1,A)$	$(2,D)$	$(4,E)$

Complexité de l'algorithme : $O(|V|^2)$

Amélioration (1/2)

- $\mathcal{S}$: l'ensemble des noeuds dont les plus court chemins sont connus vers s.
- $\mathcal{Q}$: l'ensemble des autres noeuds
- $\mathcal{O}$: liste ordonnée des chemins

Différence : les sélections se font en fonction de l'ordre des chemins et non de celui des sommets

Amélioration (2/2)

Entrée : $\begin{cases} \text{Un graphe } G = (V, E) \text{ avec une source } s \\ \text{Une fonction de poids } w : E \rightarrow R^+ \end{cases}$

Sortie : $\{$ La liste de couples (sommet, chemin)

1. Initialisation de la source s
   1.1 $\mathcal{O} \leftarrow \emptyset$;Pour chaque sommet v de s faire $\mathcal{O} \leftarrow \mathcal{O} \cup \{s, v\}$
   1.2 Trier $\mathcal{O}$.
2. $\mathcal{Q} \leftarrow V$; $\mathcal{S} \leftarrow \emptyset$;
3. Tant que $((\mathcal{O} \neq \emptyset) \wedge (Premier(\mathcal{O})$ n'a pas un poids ∞)) faire
   3.1 $P \leftarrow$ Extraire-Le-Minimum$(\mathcal{O})$;
   3.2 Soit u le dernier sommet de P
   3.3 Si $u \in \mathcal{Q}$ alors
   - $\mathcal{S} \leftarrow \mathcal{S} \cup \{u\}$; $\mathcal{Q} \leftarrow \mathcal{Q} \backslash \{u\}$;
   - associer P à u
   - Pour chaque sommet v de u faire $\mathcal{O} \leftarrow \mathcal{O} \cup (P \oplus \{u, v\})$
   - Trier $\mathcal{O}$.
4. retourner la liste de couples (sommet, chemin)

Amélioration (2/2)

Entrée : $\begin{cases} \text{Un graphe } G = (V, E) \text{ avec une source } s \\ \text{Une fonction de poids } w : E \rightarrow R^{+} \end{cases}$

Sortie : $\{$ La liste de couples (sommet, chemin)

1. Initialisation de la source s
   1.1 $\mathcal{O} \leftarrow \emptyset$;Pour chaque sommet v de s faire $\mathcal{O} \leftarrow \mathcal{O} \cup \{s, v\}$
   1.2 Trier $\mathcal{O}$.
2. $\mathcal{Q} \leftarrow V$; $\mathcal{S} \leftarrow \emptyset$;
3. Tant que $((\mathcal{O} \neq \emptyset) \wedge (Premier(\mathcal{O})$ n'a pas un poids ∞)) faire
   3.1 $P \leftarrow$ EXTRAIRE-LE-MINIMUM($\mathcal{O}$);
   3.2 Soit u le dernier sommet de P
   3.3 Si $u \in \mathcal{Q}$ alors
   - $\mathcal{S} \leftarrow \mathcal{S} \cup \{u\}$; $\mathcal{Q} \leftarrow \mathcal{Q} \backslash \{u\}$;
   - associer P à u
   - Pour chaque sommet v de u faire $\mathcal{O} \leftarrow \mathcal{O} \cup (P \oplus \{u, v\})$
   - Trier $\mathcal{O}$.
4. retourner la liste de couples (sommet, chemin)

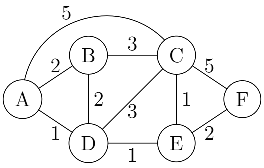
Amélioration (2/2)

Entrée : $\begin{cases} \text{Un graphe } G = (V, E) \text{ avec une source } s \\ \text{Une fonction de poids } w : E \rightarrow R^{+} \end{cases}$

Sortie : $\{$ La liste de couples (sommet, chemin)

1. Initialisation de la source s
   1.1 $\mathcal{O} \leftarrow \emptyset$;Pour chaque sommet v de s faire $\mathcal{O} \leftarrow \mathcal{O} \cup \{s, v\}$
   1.2 Trier $\mathcal{O}$.
2. $\mathcal{Q} \leftarrow V$; $\mathcal{S} \leftarrow \emptyset$;
3. Tant que $((\mathcal{O} \neq \emptyset) \wedge (Premier(\mathcal{O})$ n'a pas un poids ∞)) faire
   3.1 $P \leftarrow$ EXTRAIRE-LE-MINIMUM($\mathcal{O}$);
   3.2 Soit u le dernier sommet de P
   3.3 Si $u \in \mathcal{Q}$ alors
   - $\mathcal{S} \leftarrow \mathcal{S} \cup \{u\}$; $\mathcal{Q} \leftarrow \mathcal{Q} \backslash \{u\}$;
   - associer P à u
   - Pour chaque sommet v de u faire $\mathcal{O} \leftarrow \mathcal{O} \cup (P \oplus \{u, v\})$
   - Trier $\mathcal{O}$.
4. retourner la liste de couples (sommet, chemin)

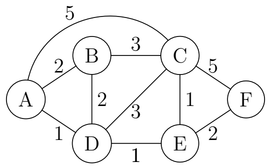
Exemple



$\mathcal{Q}$	$\mathcal{O}$	P	Association
$\{BCDEF\}$	$\{(AD), (AB), (AC)\}$		
$\{BCEF\}$	$\{(AB), (ADE), (ADB), (ADC), (AC)\}$	AD	(D, AD)
$\{CEF\}$	$\{(ADE), (ADB), (ADC), (ABD), (AC), (ABC)\}$	AB	(B, AB)
$\{CF\}$	$\{(ADEC), (ADB), (ADC), (ADEF), (ABD), (AC), (ABC)\}$	AB	(E, ADE)

Complexité de l'algorithme : $O(|E| \log |E|)$

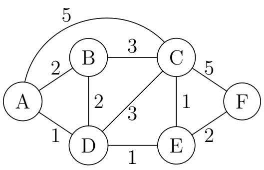
Exemple



$\mathcal{Q}$	$\mathcal{O}$	P	Association
$\{BCDEF\}$	$\{(AD), (AB), (AC)\}$		
$\{BCEF\}$	$\{(AB), (ADE), (ADB), (ADC), (AC)\}$	AD	(D, AD)
$\{CEF\}$	$\{(ADE), (ADB), (ADC), (ABD), (AC), (ABC)\}$	AB	(B, AB)
$\{CF\}$	$\{(ADEC), (ADB), (ADC), (ADEF), (ABD), (AC), (ABC)\}$	AB	(E, ADE)

Complexité de l'algorithme : $O(|E| \log |E|)$

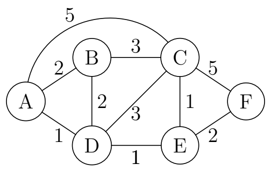
Exemple



$\mathcal{Q}$	$\mathcal{O}$	P	Association
$\{BCDEF\}$	$\{(AD), (AB), (AC)\}$		
$\{BCEF\}$	$\{(AB), (ADE), (ADB), (ADC), (AC)\}$	AD	(D, AD)
$\{CEF\}$	$\{(ADE), (ADB), (ADC), (ABD), (AC), (ABC)\}$	AB	(B, AB)
$\{CF\}$	$\{(ADEC), (ADB), (ADC), (ADEF), (ABD), (AC), (ABC)\}$	AB	(E, ADE)

Complexité de l'algorithme : $O(|E| \log |E|)$

Exemple



$\mathcal{Q}$	$\mathcal{O}$	P	Association
$\{BCDEF\}$	$\{(AD), (AB), (AC)\}$		
$\{BCEF\}$	$\{(AB), (ADE), (ADB), (ADC), (AC)\}$	AD	(D, AD)
$\{CEF\}$	$\{(ADE), (ADB), (ADC), (ABD), (AC), (ABC)\}$	AB	(B, AB)
$\{CF\}$	$\{(ADEC), (ADB), (ADC), (ADEF), (ABD), (AC), (ABC)\}$	AB	(E, ADE)

Complexité de l'algorithme : $O(|E| \log |E|)$

Chemins multiples

- Adaptation possible pour associer un sommet à plusieurs chemin

  Lors du test de l'instruction 3.3, il faut traiter le cas `SINON` en conservant tous les plus court chemin.

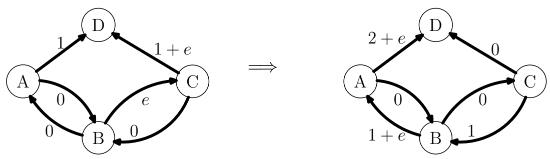
- Avantage : équilibrage de charge, meilleure robustesse du routage....

Différentes métriques

- le nombre de liens
- le débit des liens
- le coût des liens
- la fiabilité des liens
- le temps de transmission des liens
- ....

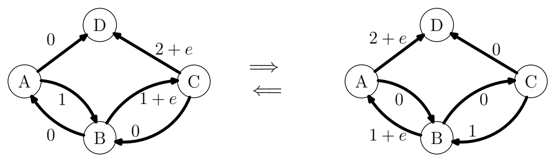
Risque d'oscillation.

Scénario : $\begin{cases} \text{métrique considérée : la charge du trafic} \\ A \to D, C \to D : 1 \text{ unité et } B \to D, : \text{e unité} \end{cases}$



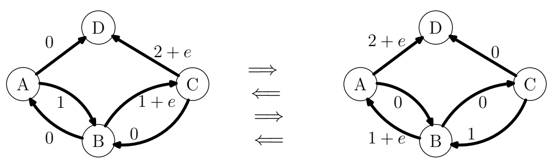
Risque d'oscillation.

Scénario : $\begin{cases} \text{métrique considérée : la charge du trafic} \\ A \to D, C \to D : 1 \text{ unité et } B \to D, : \text{e unité} \end{cases}$



Risque d'oscillation.

Scénario : $\begin{cases} \text{métrique considérée : la charge du trafic} \\ A \to D, C \to D : 1 \text{ unité et } B \to D, : \text{e unité} \end{cases}$



Instabilité des routes ! ! ! !