Systèmes et Réseaux (ASR 2) - Notes de cours
Cours 13

Anne Benoit

May 6, 2015

PARTIE 1: Systèmes

PARTIE 2: Réseaux

1 Architecture des réseaux de communication

2 La couche 2-liaison

3 La couche 3-réseau

4 Algorithmes de routage

4.1 Introduction: les différents types d'algorithmes de routage

IP: tables de routage maintenues au niveau des hôtes et des routeurs, tables utilisées par l'algorithme de transfert de paquets IP.

Routage: méthode de contrôle qui maintient les tables de routage automatiquement au niveau des routeurs. Au niveau des hôtes, utilisation de règles par défaut, et de ICMP.

Différence avec les ponts au niveau du LAN en couche 2? Comment étaient maintenues les informations de routage? En couche 2, les ponts apprennent des paquets qu'ils observent, et utilisent du broadcast initialement.

Différence entre routage et transfert de paquets. Transfert de paquets IP: fait en temps réel pour chaque paquet.

Routage: calculer les tables de routages, uniquement entre routeurs. Pas forcément temps réel: peut mettre jusqu'à 2 minutes! Le but est de minimiser une métrique, comme par exemple le nombre de hops, la capacité des liens, le coût (métriques statiques), ou encore la charge des liens et le délai courant (métriques dynamiques qui dépendent de l'état du réseau).

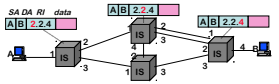
Méthodes de routage de base.

1. Configuration statique: pour jouer uniquement (tous petits réseaux).

2. Inondation: chaque paquet est dupliqué sur chaque lien sortant (noter l'id du paquet pour éviter les boucles). La destination peut recevoir plusieurs fois le même paquet. C'est un algorithme simple (pas besoin de tables de routage), robuste (résiste aux pannes de routeurs ou de liens), et optimal en terme de plus court chemin. Par contre, très coûteux et génère beaucoup de traffic inutile. Méthode utilisée en partie par certaines méthodes de routage (AODV, OLSR).

3. Routage par la source: la route est écrite par la source dans l'entête du paquet: liste des numéros de port qu'il faut emprunter. Le routeur lit le prochain hop et déplace le pointeur. Les routes sont découvertes par inondation, puis une route indiquée dans le paquet.

   Exemple: routes qui peuvent être utilisées entre A et B?



4. Anneaux de jetons: inventés dans les années 1980 comme une alternative à Ethernet. Chaque anneau correspond à un domaine de collision, interconnectés par des ponts. A génère un paquet en broadcast-toute-route, tous les ponts écoutent sur tous les anneaux auxquels ils sont connectés, et lorsqu'ils voient un paquet broadcast-toute-route, ils le retransmettent vers tous les autres anneaux, sauf ceux déjà visités par le paquet (on garde cette liste dans l'entête du paquet), ce qui crée 5 paquets différents sur l'exemple: A-R1-B1-R2-B2-R3, A-R1-B1-R2-B3-R5-B6-R6, A-R1-B1-R2-B3-R5-B5-R4, A-R1-B4-R4-B5-R5-B3-R2-B2-R3, A-R1-B4-R4-B5-R5-B6-R6. Seuls les paquets 2 et 5 atteignent la destination B, A reçoit deux ACK qui prennent la route inverse, et peut choisir une de ces routes.

Classification des algorithmes de routage.

- Routage interne (interior) vs externe (exterior): Deux types de méthodes, suivant que ce soit dans un même domaine, ou entre domaines.
- Algorithme statique (les routes changent lentement, intervention humaine possible) vs dynamique (changement des chemins de routage avec les modifications de topologie ou de charge sur les liens, réactif aux modifications, mais plus sujet aux problèmes tels les boucles, l'oscillation entre routes).
- Algorithme sensible à la charge (coût des liens qui change dynamiquement, représentant le niveau de congestion du lien) vs non sensible à la charge (c'est le cas des algorithmes de routage actuels RIP, OSPF, BGP: le coût d'un lien ne représente pas explicitement le niveau de congestion courant).

Routage à états de liens (*link state*). Algorithme de routage global qui suppose une connaissance de l'état global du système: si on connait tous les coûts des liens, on peut calculer tous les plus courts chemins vers toutes les destinations.

Fonctionnement: chaque noeud diffuse des paquets avec l'état de ses liens vers tous les autres noeuds (broadcast), et ainsi chaque noeud dispose d'une connaissance globale du réseau. Chaque noeud utilise alors l'algorithme de Dijkstra pour calculer les plus courts chemins vers tous les autres noeuds.

Utilisé dans des protocoles de routage interne, tels OSPF, PNNI (ATM).

Routage par vecteur de distance (*distance vector*). Autre grande famille d'algorithmes de routage, basés sur Bellman-Ford. Les routeurs ne connaissent que leur état local (estimation vers les voisins).

Utilisé dans des protocoles internes (RIP, IGRP).

Variante avec vecteur de chemin, utilisé en protocole externe (BGP): maintient l'information sur les chemins, l'optimisation globale et la politique de routage.

4.2 Routage par vecteur de distance

Algorithme qui calcule les plus courts chemins vers toutes les destinations de façon totalement distribuée, en utilisant uniquement les distances entre soi-même et toutes les destinations.

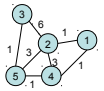
Utilisation de Bellman-Ford distribué (pas facile de distribuer Dijkstra). Coût $A(i,j)$ du lien (i,j), et le but est de calculer le plus court chemin PCC pour tout couple de sommet. $A(i,j) > 0$, et $A(i,j) = +\infty$ si i et j ne sont pas connectés.

Version centralisée, BFC. $p^k(i)$ est le coût de PCC de i vers j en au plus k hops, où j est fixé.

BFC (pour j fixé):
Initialement, $p^0(j) = 0$, et $p^0(i) = +\infty$ pour $i \neq j$.
Tant que $p^k \neq p^{k-1}$,
$p^k(i) = \min_{\ell \neq i}\{A(i,\ell) + p^{k-1}(\ell)\}$ pour $i \neq j$,
et $p^k(j) = 0$.

Théorème: si le réseau est connexe, l'algorithme s'arrête au plus tard pour $k = n$, et alors $p^n(i) = p(i)$ est le coût d'un PCC pour tout i. On définit le prédécesseur de i sur le chemin par $pred(i) = \text{argmin}_{\ell \neq i}\{A(i,\ell) + p(\ell)\}$.

Exemple: écrire $p^k(i)$, $pred(i)$, et dessiner les PCC depuis $j = 1$.



Impact des conditions initiales. Est-ce-que l'algorithme converge si l'on change les conditions initiales?

k\i	1	2	3	4	5
0	0	0	0	0	0
1					
2					
3					
4					

k\i	1	2	3	4	5
0	0	6	1	1	0
1					
2					
3					
4					

Théorème: l'algorithme converge en un nombre fini d'étapes vers les valeurs correctes pour toute condition initiale telle que $p^0(j) = 0$ et pour chaque noeud i connecté à j.

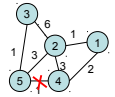
Version distribuée 1ère version BFD1. Chaque noeud i maintient une estimation $q(i)$ de la distance $p(i)$ de i à 1; $q(1) = 0$ tout le temps et les autres conditions initiales sont quelconques.

De temps en temps, i envoie sa valeur de $q(i)$ à tous ses voisins.

Lorsque i reçoit un $q(j_0)$ depuis un voisin j_0, il met à jour $q(j_0)$ et

$$q(i) = \min_{j \text{ voisin de } i}\{A(i,j) + q(j)\}. \qquad (1)$$

A possible run of algorithm BFD1:



Q: give a possible scenario after link 4-5 breaks

	i	1	2	3	4	5
		0	∞	∞	∞	∞
1 -> 2		0	1	∞	∞	∞
2 -> 5		0	1	∞	∞	4
2 -> 3		0	1	7	∞	4
5 -> 4		0	1	7	5	4
2 -> 4		0	1	7	4	4
1 -> 4		0	1	7	2	4
4 -> 5		0	1	7	2	3
5 -> 2		0	1	7	2	3
5 -> 3		0	1	4	2	3
link breaks		4 does as if received ∞ from 5 5 does as if received ∞ from 4 and continue computations from there				
		0	1	4	2	4
5 -> 3		0	1	5	2	4

Si le temps d'envoi d'un message est borné par T, l'algorithme converge au même résultat que la version centralisée en temps au plus nT (si le réseau est connexe et s'il n'y a pas de pannes).

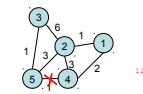
Limite de cet algorithme: chaque noeud doit se souvenir des estimations déjà reçues de tous ses voisins, même si ce ne sont pas les meilleurs. Problème en pratique si l'on doit calculer les plus courts chemins vers toutes les destinations (vecteur de distance = distances vers tous les noeuds du graphe).

Solution: remplacer l'équation (1) par $q(i) = \min(A(i,j_0) + q(j_0), q(i))$: on choisit le nouveau chemin s'il est plus court que l'ancien. Est-ce une solution convenable?

Version distribuée 2ème version BFD2. Version alternative qui permet de ne se souvenir que du meilleur voisin ($pred(i)$): l'équation (1) est remplacée par:

(2) Si $j_0 = pred(i)$, alors $q(i) = A(i,j_0) + q(j_0)$.
Sinon, si $A(i,j_0) + q(j_0) < q(i)$ alors $q(i) = A(i,j_0) + q(j_0)$ et $pred(i) = j_0$.

Différence avec BFD1 si $j_0 = pred(i)$, car alors on repart de la valeur $A(i,j_0) + q(j_0)$ au lieu de faire le minimum sur tous les voisins. Cette valeur est toujours supérieure à la valeur obtenue par l'équation (1). Après un échange avec les voisins, ceci est réparé et on peut retrouver la valeur de l'équation (1).



Q: give a possible scenario after link 4-5 breaks

A possible run of algorithm BFD2:

i	1	2	3	4	5
	0	∞	∞	∞	∞
1 -> 2	0	1	∞	∞	∞
2 -> 5	0	1	∞	∞	4
2 -> 3	0	1	7	∞	4
5 -> 4	0	1	7	5	4
2 -> 4	0	1	7	4	4
1 -> 4	0	1	7	2	4
4 -> 5	0	1	7	2	3
5 -> 2	0	1	7	2	3
5 -> 3	0	1	4	2	3

link breaks

4 does as if received ∞ from 5
5 ≠ pred(4)

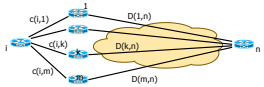
	0	1	4	2	3

3 does as if received ∞ from 4
4 == pred(5)

	0	1	4	2	∞
5 -> 3	0	1	∞	2	∞
2 -> 3	0	1	7	2	∞
2 -> 5	0	1	7	2	4
5 -> 3	0	1	5	2	4

En pratique...

- Le noeud i calcule le plus court chemin et next hop pour chaque préfixe de réseau n qu'il connaît.
- Initialement, $D(i,n) = 0$ si i est directement connectée à n, $+\infty$ sinon.
- Le noeud i reçoit du voisin k sa dernière valeur $D(k,n)$ pour tout n (c'est le **vecteur de distance**), et le noeud i calcule les meilleures estimations avec BFD2.
- L'algorithme converge si le réseau est stable. Mécanisme de "hello" pour reprendre les calculs après des modifications: si un voisin k disparaît, timeout car i ne reçoit plus de messages "hello" de k. Equivalent à un message $D(k,n) = +\infty$ pour tout n.



Exemples: voir transparents.

4.3 Protocoles de routage

4.3.1 RIP/RIPv2

Protocole utilisant les vecteurs de distance. La métrique est le nombre de hops. Taille du réseau limitée à 15: $\infty = 16$. Heuristique de l'horizon coupé. Réseau destination identifié par son adresse IP (masques dans RIPv2). Paquets UDP. Broadcast toutes les 30 secondes ou lors de la détection d'une modification, et si une route n'est pas annoncée pendant 3 minutes, timeout (le coût devient ∞).

4.3.2 IGRP (Interior Gateway Routing Protocol)

Protocole propriétaire de CISCO, avec une métrique qui estime le délai global. Plusieurs routes de coût identique sont maintenues afin d'équilibrer la charge. Pas de limite de 15, mais le nombre de routeurs est inclus dans les messages. Boadcast toutes les 90 secondes.

4.4 Routage dépendant de la charge

Le plus court chemin ne produit pas toujours un routage qui maximise le flot total.

Solution: prendre comme coût le délai: si la charge est élevée sur un lien, alors le coût est plus élevé et le lien sera moins utilisé.

Cependant, l'ajout d'un nouveau lien peut diminuer le débit total (paradoxe de Braess): le routage pour avoir les délais les plus courts n'est pas non plus un optimum global.

Routage optimal. Changer l'objectif du routage: minimiser le délai total sujet à des contraintes de flots. La solution optimale dépend de tous les flots. Algorithme distribué proche de l'algorithme de contrôle de congestion dans TCP [BertsekasGallager92].

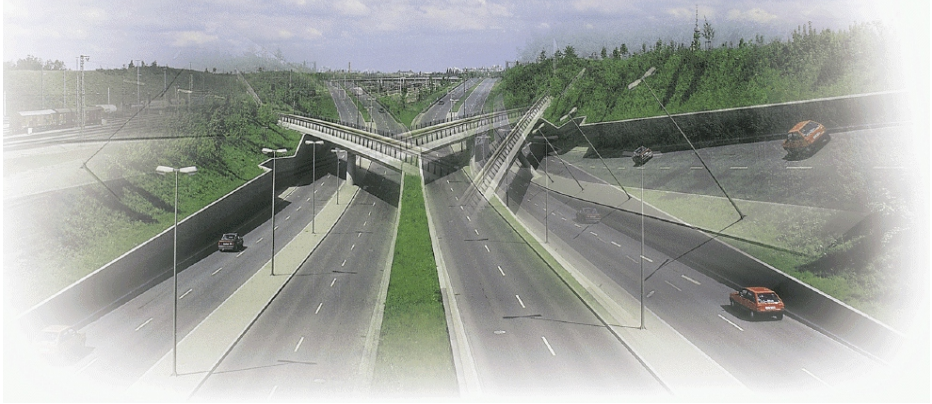
4.5 Conclusion

L'algorithme à vecteur de distance est intelligent: il est totalement distribué, peu d'information doit être stockée, et c'est simple. Ainsi, il est largement déployé.

Cependant, la convergence est relativement lente, il n'est donc pas adapté à des réseaux grands et complexes. Dans ce cas, on utilise plutôt des protocoles à états de liens.

Exemples sur les algorithmes de routage

Anne Benoit



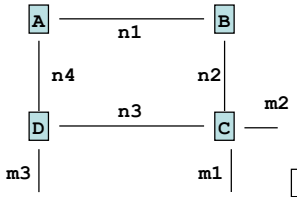
Exemple 1

A

net	dist	nxt
n1	0	n1,A
n4	0	n4,A

B

net	dist	nxt
n1	0	n1,B
n2	0	n2,B



D

net	dist	nxt
n3	0	n3,D
n4	0	n4,D
m3	0	m3,D

C

net	dist	nxt
n2	0	n2,C
n3	0	n3,C
m1	0	m1,C
m2	0	m2,C

Exemple 1

A

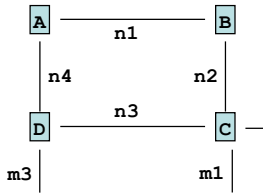
net	dist	nxt
n1	0	n1,A
n4	0	n4,A

from A

n1	0
n4	0

B

net	dist	nxt
n1	0	n1,B
n2	0	n2,B
n4	1	n1,A



D

net	dist	nxt
n3	0	n3,D
n4	0	n4,D
m3	0	m3,D

from D

n3	0
n4	0
m3	0

C

net	dist	nxt
n2	0	n2,C
n3	0	n3,C
m1	0	m1,C
m2	0	m2,C
n4	1	n3,D
m3	1	n3,D

Exemple 1

A

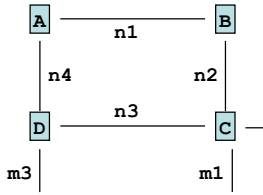
net	dist	nxt
n1	0	n1,A
n4	0	n4,A

B

net	dist	nxt
n1	0	n1,B
n2	0	n2,B
n3	1	n2,C
n4	1	n1,A
m1	1	n2,C
m2	1	n2,C
m3	2	n2,C

from C

n2	0
n3	0
m1	0
m2	0
n4	1
m3	1



D

net	dist	nxt
n3	0	n3,D
n4	0	n4,D
m3	0	m3,D

C

net	dist	nxt
n2	0	n2,C
n3	0	n3,C
m1	0	m1,C
m2	0	m2,C
n4	1	n3,D
m3	1	n3,D

Exemple 1: Final

A

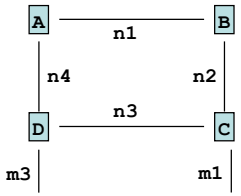
net	dist	nxt
n1	0	n1,A
n2	1	n1,B
n3	1	n4,D
n4	0	n4,A
m1	2	n4,D
m2	2	n4,D
m3	1	n4,D

B

net	dist	nxt
n1	0	n1,B
n2	0	n2,B
n3	1	n2,C
n4	1	n1,A
m1	1	n2,C
m2	1	n2,C
m3	2	n2,C

D

net	dist	nxt
n1	1	n4,A
n2	1	n3,C
n3	0	n3,D
n4	0	n4,D
m1	1	n3,C
m2	1	n3,C
m3	0	m3,D



C

net	dist	nxt
n1	1	n2,B
n2	0	n2,C
n3	0	n3,C
m1	0	m1,C
m2	0	m2,C
n4	1	n3,D
m3	1	n3,D

Exemple 1: Panne

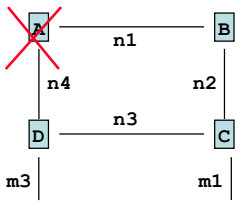
(On ne montre que le routeur dans la colonne nxt, pour simplifier)

B

net	dist	nxt
n1	0	B
n2	0	B
n3	1	C
n4	1	A
m1	1	C
m2	1	C
m3	2	C

D

net	dist	nxt
n1	1	A
n2	1	C
n3	0	D
n4	0	D
m1	1	C
m2	1	C
m3	0	D



C

net	dist	nxt
n1	1	B
n2	0	C
n3	0	C
m1	0	C
m2	0	C
n4	1	D
m3	1	D

Exemple 1: Panne

B

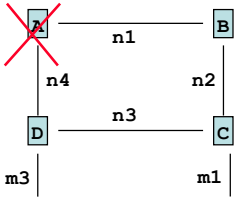
net	dist	nxt
n1	0	B
n2	0	B
n3	1	C
n4	1	A
m1	1	C
m2	1	C
m3	2	C

timeout

D

timeout

net	dist	nxt
n1	1	A
n2	1	C
n3	0	D
n4	0	D
m1	1	C
m2	1	C
m3	0	D



C

net	dist	nxt
n1	1	B
n2	0	C
n3	0	C
m1	0	C
m2	0	C
n4	1	D
m3	1	D

Exemple 1: Panne

B

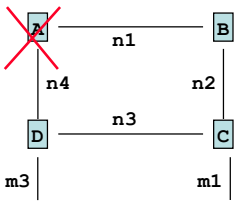
net	dist	nxt
n1	0	B
n2	0	B
n3	1	C
m1	1	C
m2	1	C
m3	2	C

D

net	dist	nxt
n1	2	C
n2	1	C
n3	0	D
n4	0	D
m1	1	C
m2	1	C
m3	0	D

From C:

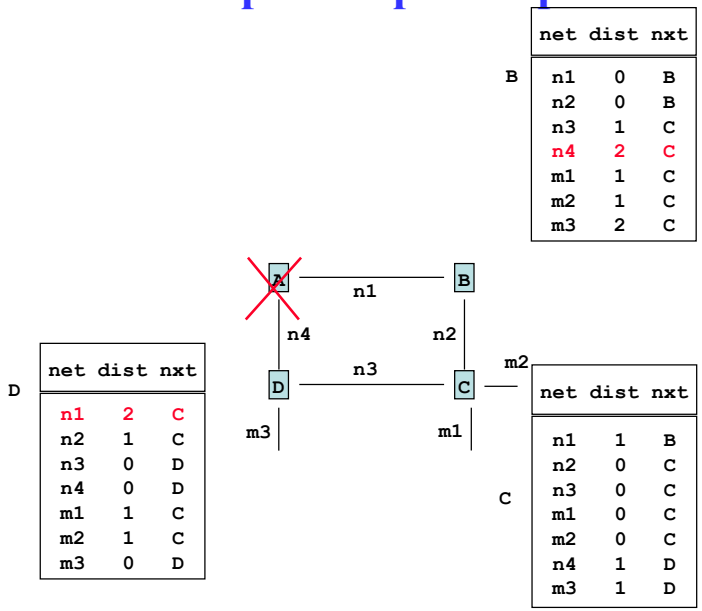
n1	1	B
n2	0	C
n3	0	C
m1	0	C
m2	0	C
n4	1	D
m3	1	D



C

net	dist	nxt
n1	1	B
n2	0	C
n3	0	C
m1	0	C
m2	0	C
n4	1	D
m3	1	D

Exemple 1: Après la panne



B

net	dist	nxt
n1	0	B
n2	0	B
n3	1	C
n4	2	C
m1	1	C
m2	1	C
m3	2	C

D

net	dist	nxt
n1	2	C
n2	1	C
n3	0	D
n4	0	D
m1	1	C
m2	1	C
m3	0	D

C

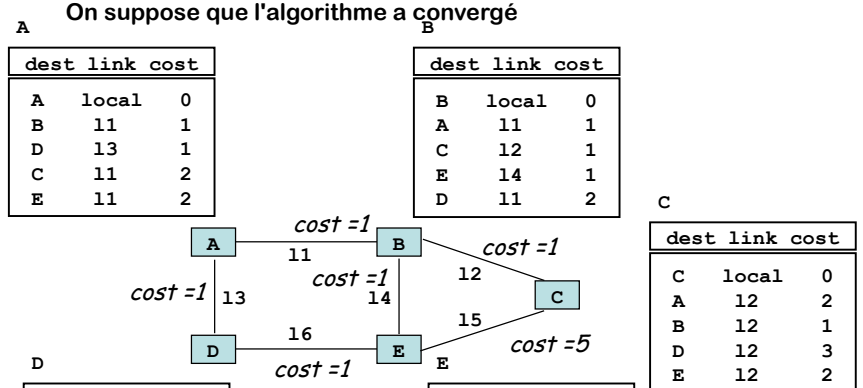
net	dist	nxt
n1	1	B
n2	0	C
n3	0	C
m1	0	C
m2	0	C
n4	1	D
m3	1	D

Exemple 1: Conclusions

- Cet exemple illustre :
  - Comment Bellman-Ford s'applique concrètement sur un réseau
  - Comment les modifications de topologies sont prises en compte
    - Les annonces les plus récentes remplacent les anciennes
    - Les annonces non rafraîchies deviennent obsolètes
  - Comment le vecteur de distance transporte les informations sur les sommets atteignables

Exemple 2

Pour simplifier, destination = routeur.
On suppose que l'algorithme a convergé



A

dest	link	cost
A	local	0
B	11	1
D	13	1
C	11	2
E	11	2

B

dest	link	cost
B	local	0
A	11	1
C	12	1
E	14	1
D	11	2

C

dest	link	cost
C	local	0
A	12	2
B	12	1
D	12	3
E	12	2

D

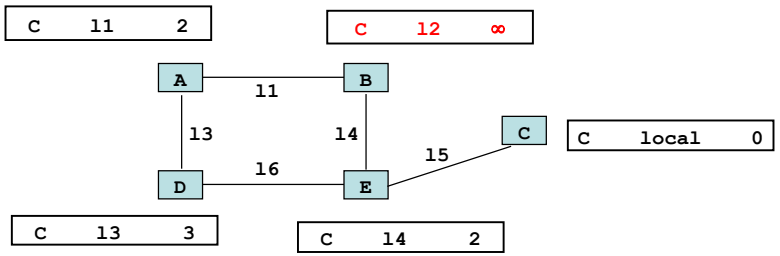
dest	link	cost
D	local	0
A	13	1
B	13	2
C	13	3
E	16	1

E

dest	link	cost
E	local	0
A	14	2
B	14	1
D	16	1
C	14	2

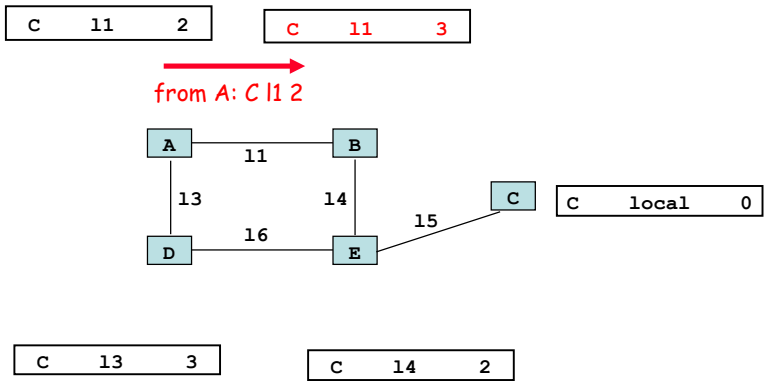
Exemple 2

- Uniquement les entrées des tables vers C
- Panne du lien l2 entre B et C
- B met à jour sa table



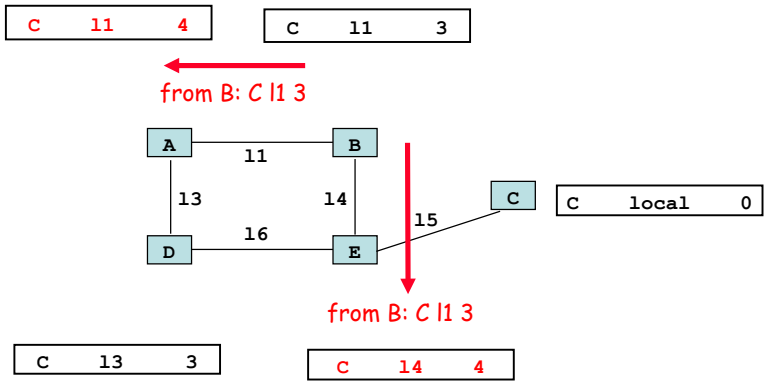
Exemple 2: Panne sur un lien

- Juste avant cette mise à jour, A broadcaste sa table avec coût 2 vers C
- B met à jour sa table



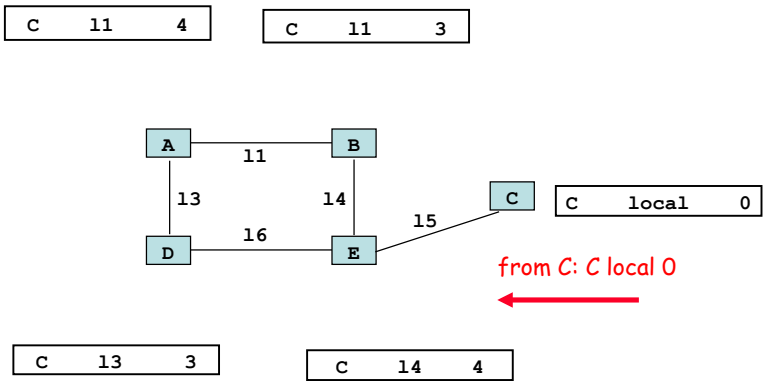
Exemple 2: Panne sur un lien

- B envoie une mise à jour à A et à E
- A et E mettent leurs tables à jour



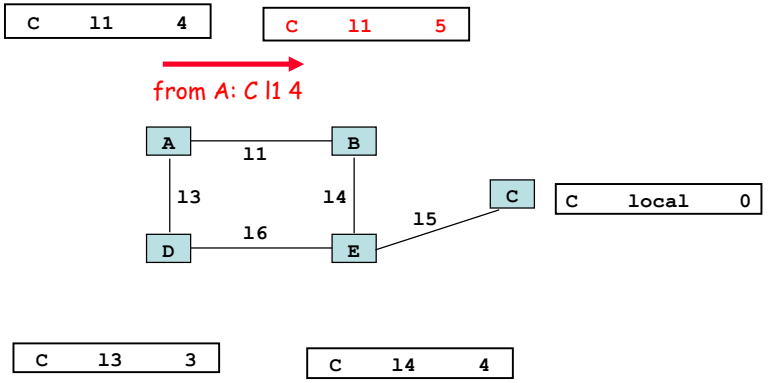
Exemple 2: Panne sur un lien

- C envoie une mise à jour
- Ignorée par E (moins bonne, lien E-C de coût 5)



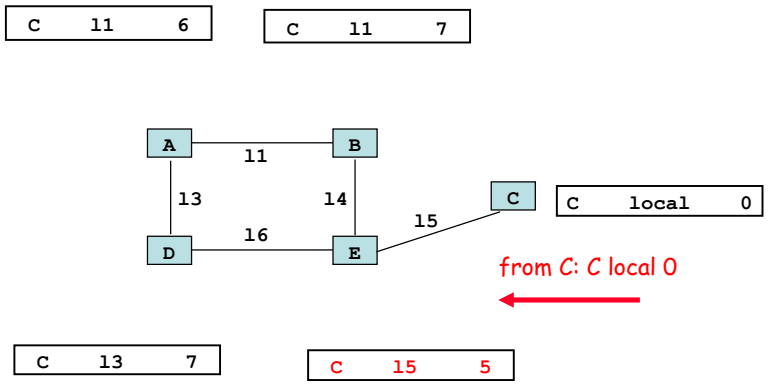
Exemple 2: Panne sur un lien

- A broadcaste sa table avec un coût 4 vers C
- B met à jour… Boucle entre A et B !
- Le coût augmente de 2 à chaque itération



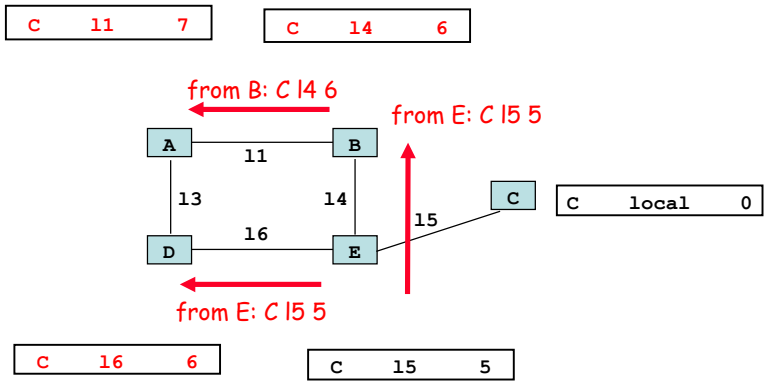
Exemple 2: Panne sur un lien

- E accepte maintenant l'annonce de C



Exemple 2: Panne sur un lien

- E envoie à D et B
- B et D envoient à A
- Et finalement l'algorithme converge dans un état stable



Exemple 2: Conclusions

- L'algorithme converge après modification de la topologie, mais la convergence peut être très lente: effet de rebond.
- Question: que dire des tables de routage avant la convergence?

Exemple 3

Tous les coûts sont maintenant à 1
Pannes des liens l1 et l6
D détecte la panne et met son coût à ∞

A

dest	link	cost
A	local	0
B	13	3
D	13	1
C	13	3
E	13	2

B

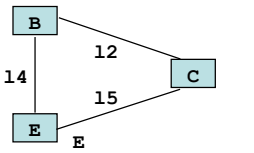
dest	link	cost
B	local	0
A	14	3
C	12	1
E	14	1
D	14	2

C

dest	link	cost
C	local	0
A	15	3
B	12	1
D	15	2
E	15	1

D

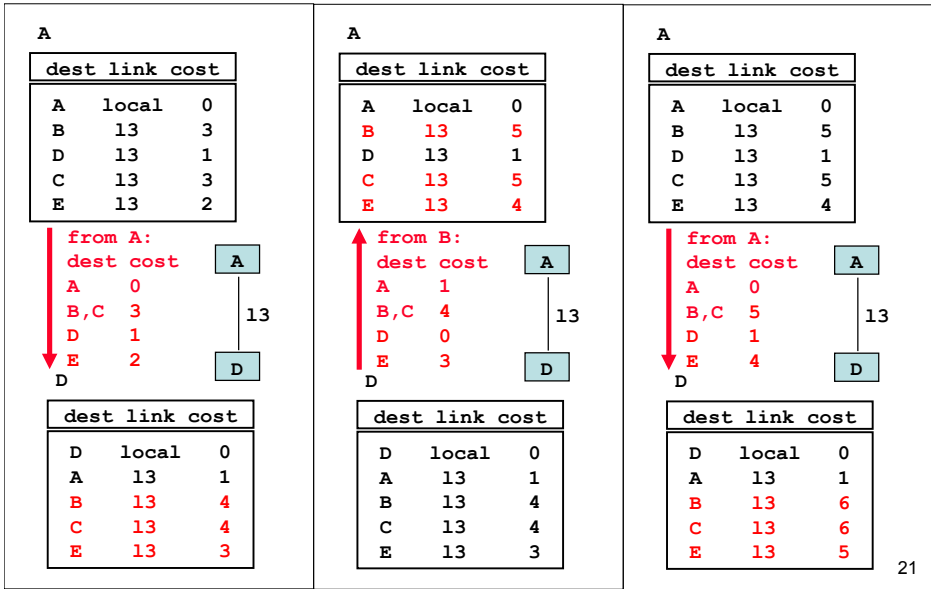
dest	link	cost
D	local	0
A	13	1
B	16	∞
C	16	∞
E	16	∞



E

dest	link	cost
E	local	0
A	16	2
B	14	1
D	16	1
C	15	1

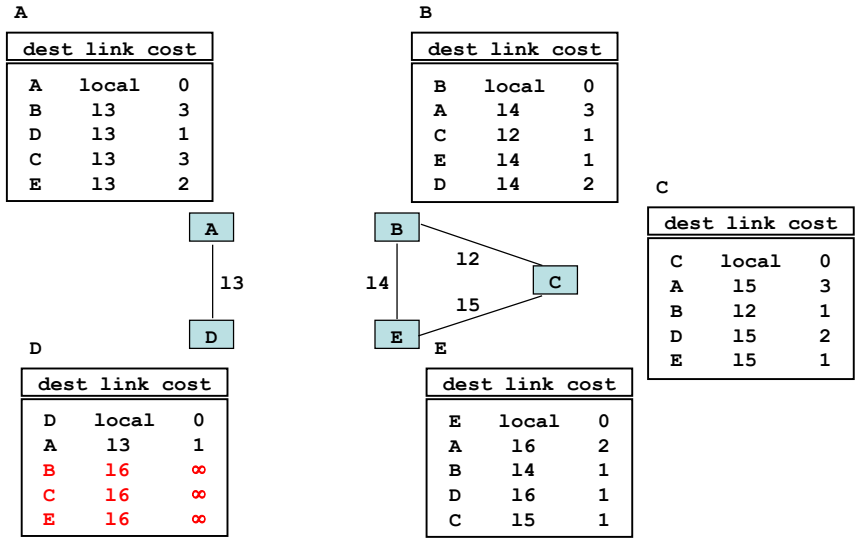
Exemple 3



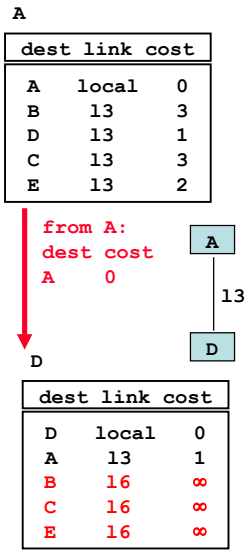
Exemple 3: Conclusions

- Les coûts de C, B, E augmentent indéfiniment: “Count to Infinity”
  - Les vrais coûts sont infinis
- Convergence vers un état stable si on fixe
  - ∞ = grand nombre
  - Par exemple, dans RIP, on a ∞ = 16
- “Split Horizon”: horizon coupé
  - Heuristique pour éviter ce phénomène
  - Si A route les paquets vers X via B, il n'annonce pas cette route à B

Exemple 3: avec l'horizon coupé



Exemple 3: avec l'horizon coupé



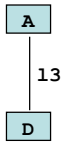
Exemple 3: avec l'horizon coupé

A

dest	link	cost
A	local	0
B	13	∞
D	13	1
C	13	∞
E	13	∞

from D:

dest	cost
D	0
B,C,E	∞



D

dest	link	cost
D	local	0
A	13	1
B	16	∞
C	16	∞
E	16	∞

L'horizon coupé supprime l'augmentation progressive vers ∞

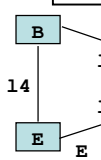
L'horizon coupé ne marche pas toujours

B

dest	link	cost
B	local	0
A	14	∞
C	12	1
E	14	1
D	14	∞

from E:

dest	cost
E	0
A	∞
C	1
D	∞



E

dest	link	cost
E	local	0
A	16	∞
B	14	1
D	16	∞
C	15	1

C

dest	link	cost
C	local	0
A	15	3
B	12	1
D	15	2
E	15	1

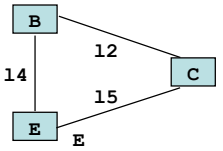
L'horizon coupé ne marche pas toujours

B

dest	link	cost
B	local	0
A	12	4
C	12	1
E	14	1
D	12	3

from C:

dest	cost
A	3
D	2
E	1



C

dest	link	cost
C	local	0
A	15	3
B	12	1
D	15	2
E	15	1

E

dest	link	cost
E	local	0
A	16	∞
B	14	1
D	16	∞
C	15	1

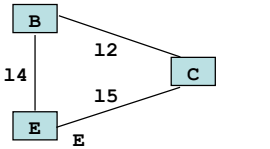
from C:

dest	cost
B	1

L'horizon coupé ne marche pas toujours

B

dest	link	cost
B	local	0
A	12	4
C	12	1
E	14	1
D	12	3



C

dest	link	cost
C	local	0
A	15	3
B	12	1
D	15	2
E	15	1

from B:

dest	cost
A	4
B	0
C	1
D	3

E

dest	link	cost
E	local	0
A	14	5
B	14	1
D	14	4
C	15	1

Conclusion

- La convergence vers un état stable peut être lente après des modifications de l'état du réseau.

- Pour éviter le comptage vers l'infini, il faut par exemple fixer une distance maximum.