Protocole OSPF + Protocole de Diffusion

Exercice : protocole de routage OSPF

Dans cet exercice, nous nous intéressons au calcul de la table de routage. Supposons que le nœud F du réseau a la vision suivante du réseau :

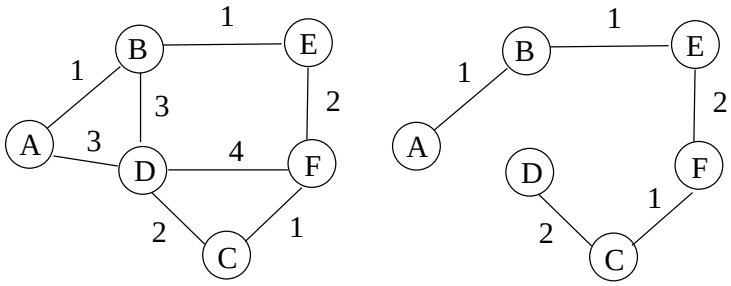
liaison	distance
A vers B	1
A vers D	3
B vers D	3
B vers E	1

liaison	distance
C vers F	1
C vers D	2
D vers A	3
D vers B	3
D vers F	4

liaison	distance
E vers B	1
E vers F	2
F vers C	1
F vers D	4
F vers E	2

1. Dessinez le réseau à partir des informations ci-dessus.

   Correction :



2. Ecrivez la table de routage du noeud F.

   Correction :

   – route : = un plus court chemin dans le réseau
   – construction d'un arbre de plus court chemin en utilisant l'algorithme de Dijkstra.

   Description des variables.

   – `cap(k) correspond si l' algo a déjà décidé si l' arête est déjà dans l' arbre.`
   - `d(i) distance entre le noeud racine` s `de l'arbre.`

   Description de l' algorithme.

   `Procedure initialisation :`
   $\forall v$, v `sommets de` G, `distance(`v`) :=` $infinity$
   $\forall v$, v `sommets de` G, $pere(v)$ `:=` nil
   $d(s) = 0$

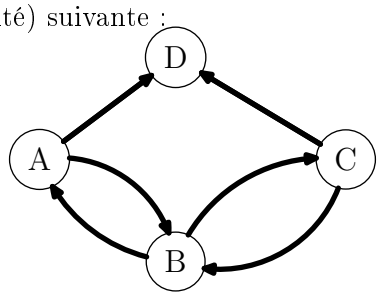
   `Procedure centrale`
   `Pour` $i = 1, \ldots, n-1$ `faire`
   `Pour chaque arc (u,v) de` G `faire`
   `si` $d(v) > d(u) + cap(u,v)$ `alors`
   $d(v) \leftarrow d(u) + cap(u,v)$
   $pere(v) = v$

destination	liaison
A	F vers E
B	F vers E
C	F vers C
D	F vers C
E	F vers E
F	locale

Exercice : protocole OSPF et ses différentes métriques.

Le protocole de routage OSPF calcule les routes de plus cout chemins en fonction d'une fonction de coût (ou de distance) sur les arêtes. Cette fonction peut être par exemple, le coût des liens, la fiabilité des liens, le temps de transmission des liens. Dans cet exercice, nous considérons que la fonction de coût d'un lien correspond à la charge de traffic qui le traverse.

Considérons la topologie (orienté) suivante :

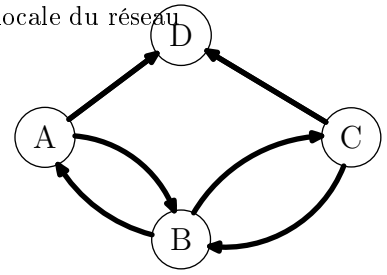


Au top 0, aucun traffic existe dans le réseau. Au top 1, la situation suivante est considérée :

1. la quantité de traffic allant de A vers D et celle allant de C vers D correspond à 1 unité.
2. la quantité de traffic allant de B vers D et celle allant de C vers D correspond à e unité avec $e < 1$

1. Déterminer les routes pour aller vers D de chaque noeud du réseau au top 0. *(si deux routes ont de même coût, alors la route utilisant le nombre de liens le plus petit sera favorisée).*
   Correction :

   vue locale du réseau

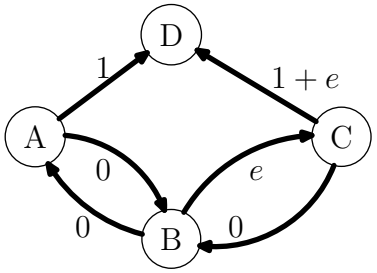
   

   - la route pour aller de A vers D est AD
   - la route pour aller de C vers D est CD
   - la route pour aller de B vers D est BCD

2. Au top 1, le traffic circule dans le réseau. Les coûts des arêtes changent. Chaque noeud du réseau s'appercoit de ces modifications et adapte sa table de routage en fonction.

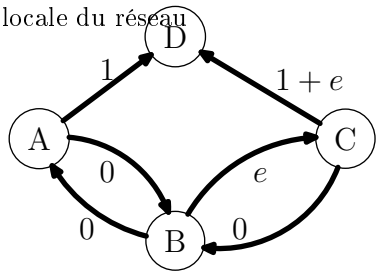
   (a) Quelles sont les nouveaux coûts des arêtes ?
   Correction :

- le coût de l'arc $A \to D$ est 1
- le coût de l'arc $B \to C$ est e
- le coût de l'arc $C \to D$ est $1+e$
- le coût des autres arcs est 0



(b) Déterminer les routes pour aller vers D de chaque noeud du réseau.
Correction :

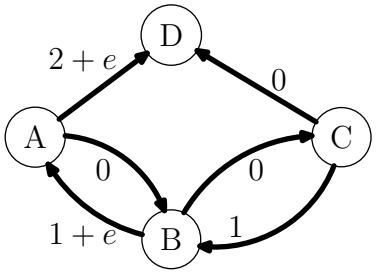
vue locale du réseau



- la route pour aller de A vers D est AD
- la route pour aller de C vers D est $CBAD$
- la route pour aller de B vers D est BAD

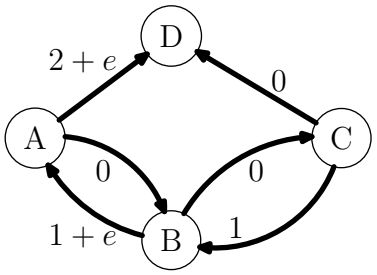
3. Au top 2, chaque noeud s'appercoit que les coûts des arêtes changent. Ils recalculent ainsi leur table de routage.

(a) Quelles sont les nouveaux coûts des arêtes ?
Correction :



- le coût de l'arc $A \to D$ est $2+e$
- le coût de l'arc $B \to A$ est $1+e$
- le coût de l'arc $B \to C$ est 1
- le coût des autres arcs est 0

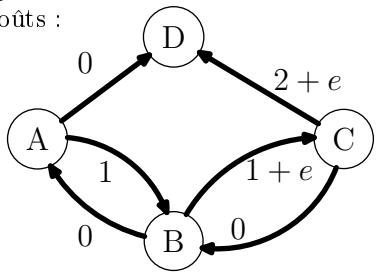
(b) Déterminer les routes pour aller vers D de chaque noeud du réseau.
Correction :



- la route pour aller de A vers D est $ABCD$
- la route pour aller de C vers D est CD
- la route pour aller de B vers D est BCD

4. et ainsi de suite. Qu'en déduisez-vous ?
Correction :

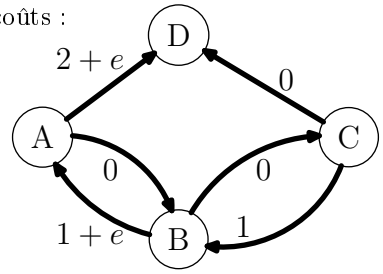
configuration 1 : description
Les coûts :



Alors les nouvelles routes sont les suivantes :
- la route pour aller de A vers D est AD
- la route pour aller de C vers D est $CBAD$
- la route pour aller de B vers D est BAD

configuration 2 : description

Les coûts :



Alors les nouvelles routes sont les suivantes :
- la route pour aller de A vers D est $ABCD$
- la route pour aller de C vers D est CD
- la route pour aller de B vers D est BCD

Le système oscille entre la configuration 1 et la configuration 2. Les routes sont instables!!! ce qui n'est pas bien pour des protocoles de routage car plus les routes sont instables, plus les paquets ont des chances de se perdre ou de circuler longtemps dans le réseau. Prendre la charge de traffic sur un lien comme métrique est une mauvaise idée.

Diffusions dans les systèmes distribués

La diffusion est l'opération qui consiste pour un site donné à envoyer un même message à tous les autres sites d'un système. Nous considérons un système constitué de n sites $P_0, P_1, \ldots, P_{n-1}$ tous interconnectés.

Exercice : Diffusion asynchrone en cas de panne sur les liens.

1. Décrire un algorithme de diffusion à partir de P_0 vers les autres sites (en supposant aucune panne). Calculer le nombre de communications nécessaires pour réaliser cette diffusion.

Correction : La procedure suivante est decrite pour le noeud p_0 initiateur.

```
Procedure diffuser(M)
  Envoyer(<M>) a $P_1,\dots,P_{n-1}$ dans cet ordre
```

Correction : $n-1$ communications.

Nous voulons construire un algorithme de diffusion qui fonctionne même en cas de pannes de certains sites. Nous appelons panne d'un site l'arrêt soudain d'un site. On suppose qu'un site une fois en panne reste en panne et ne fait plus rien.

2 Proposer un algorithme de diffusion à partir de P_0 vers les autres sites qui tolère une panne d'un des sites (on suppose que si P_0 est en le site en panne, il a au moins le temps d'envoyer un message). Calculer le nombre de communications nécessaires pour réaliser cette diffusion.

Correction : Soit $S_0 = \{p_0, p_1\}$

La procedure suivante est decrite pour le noeud p_0 initiateur.

```
Procedure diffuser(M)
  Envoyer(<M>) a $P_1,\dots,P_{n-1}$ dans cet ordre;
  Pour tout (q tel que  q dans V - S0)  faire
       Envoyer(<M>) a q
  Accepter(M) ;
```

La procedure suivante est decrite pour tout noeud $q_k \neq -p_0$.

```
Procedure diffuser(M)
  Si (q_k \in S_0) alors
   Si (k < 1) alors
     Envoyer(<M>) a $P_{k+1},\dots,P_{t}$ dans cet ordre;
```

```
  Pour tout (q tel que  q dans V - S0)  faire
       Envoyer(<M>) a q
  accepter(M) ;
```

Correction : $n-1+n-2 = 2n-3$ communications

3 Généraliser cet algorithme qui tolère jusqu'à t sites en panne.

Correction :

Soit $S_0 = \{p_0, p_1, \ldots, p_t\}$

La procedure suivante est decrite pour le noeud p_0 initiateur.

```
Procedure diffuser(M)
  Envoyer(<M>) a $P_1,\dots,P_{n-1}$ dans cet ordre;
  Pour tout (q tel que  q dans V - S0)  faire
       Envoyer(<M>) a q
  Accepter(M) ;
```

La procedure suivante est decrite pour tout noeud $q_k \neq -p_0$.

```
Procedure diffuser(M)
  Si (q_k \in S_0) alors
   Si (k < t) alors
     Envoyer(<M>) a $P_{k+1},\dots,P_{t}$ dans cet ordre;
  Pour tout (q tel que  q dans V - S0)  faire
       Envoyer(<M>) a q
  accepter(M) ;
```

Principe du tout ou rien.

- Si p tombe en panne. Si p a le temps d'envoyer un message, tous les autres sites le recevront sinon personnes ne le recoit.
- si c'est un site q tombe en panne tous les sites vont recevoir le message.

Correction :

- Le site envoie $n-1$ messages.
- Chaque site q_i envoie $t-i+n-(t+1) = n-i-1$ messages.
- $n-1+\sum_{t=1}^{t}(n-i-1) = (t+1)(n-1-t/2)$ messages

Exercice : Diffusion ne respectant pas l'ordre FIFO des messages

On suppose maintenant que les sites sont tous fiables, mais que le réseau de communications peut ne pas respecter l'ordre FIFO. On suppose maintenant que P_0 diffuse plusieurs messages vers les autres sites. Proposer une méthode pour garantir que ces sites traitent les messages en respectant l'ordre d'envoi par P_0.

Correction :

Description des variables

- `num_envoi(p) le numéro d'envoi`
- `seq(i) signifie que le noeud` i `a reçu le message de numéro` $1, \ldots, seq(i)-1$.

Code du site p

```
Procédure Init
```
$num_envoi(p) := 0$;

```
Procédure diffuser (M)
num_envoi(p) := num_envoi(p) + 1 ;
Pour tout noeud q dans V différent de p faire
    Envoyer(<M, num_envoi(p) >) à q ;
```

Code du site $u \neq p$

```
Procédure Init
seq(u) := 1 ;


Lors de la réception de <M, num_envoi(p) > de p
Stocker (M) ;
Attendre (num_envoi(p)= seq(u)) ;
Délivrer(M) ;
seq(u) := seq(u) + 1 ;
Détruire(M) ;
```

Inconvénient du protocole

- un site i peut avoir à stocker beaucoup de messages avant de pouvoir les délivrer et de pouvoir les détruire (par exemple si un des messages est très lent par rapport aux suivants et arrive bien après eux).
- Le numéro de séquence des messages croît au delà de toute limite raisonnable si p diffuse beaucoup de messages. C'est un problème de taille de messages.
- Ici on utilise explicitement le fait que le réseau ne perd pas de message. Dans le cas contraire,le protocole peut être bloqué et ne plus délivrer de messages.

Pour résoudre les deux premiers points on peut mettre en place un système `d'acquittements dans une fenêtre de taille t fixée`. Dans ce système, le site p fait au plus t diffusions de suite avant de recevoir des acquittements. Chaque fois qu'un site i a pu délivrer un message, il envoie un acquittement à p, comprenant le numéro du message acquitté. Lorsque p a reçu les acquittements de tous les sites, pour les derniers messages envoyés non encore acquittés, il peut continuer à diffuser. Dans ce protocole, les numéros de séquence des messages diffusés et les numéros de séquence en réception sont construits modulo t, en commençant à 0. Du coup, les numéros de séquence ne dépassent pas une certaine valeur et le deuxième problème est résolu. Chaque site n'aura à stocker qu'au plus t messages avant de les détruire et si t est suffisamment petit, le premier problème est résolu.