

Le routage

But

Nous allons étudier quelques-uns des problèmes qui se posent lorsque l'on veut acheminer des informations via un réseau de communication.

1 Introduction

Le routage de paquets d'information d'un point A à un point B via un réseau de communication reposant sur des liaisons point à point est une des fonctionnalités de la couche Réseau. Il s'agit de mettre à disposition un service qui permette l'acheminement de paquets dans le réseau sans avoir à spécifier de façon explicite le chemin à emprunter. Indépendamment du type de service offert (avec ou sans connexion), nous nous intéressons ici aux problèmes posés par la constitution des tables de routages nécessaires à l'acheminement des paquets et à l'impact sur le fonctionnement du réseau des réponses apportées à ces problèmes.

Par la suite, le réseau de communication sera modélisé par un graphe non-orienté et connexe dont les arcs seront étiquetés avec une fonction de coût c . Les algorithmes de routage ont tous un dénominateur commun : il s'agit pour aller d'un point A à un point B , de minimiser le coût total du chemin (la somme des coûts des arcs du chemin). Par abus de langage, on parlera de *plus court chemin*.

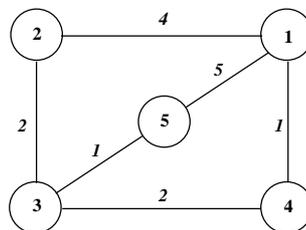


FIG. 1 – un réseau de communication

Exemple Dans le réseau de la figure 1, $c_{15} = c_{51} = 5, c_{14} = c_{41} = 1, \dots$. Le plus court chemin pour aller de 5 à 1 est 5,3,4,1.

On attribue à chaque site i une table de routage $routage_i[1..n]$. Dans un premier temps, on considère qu'une table de routage est composée de n entrées (n désignant le nombre de sites) indiquant, pour chaque destination, lequel des voisins (du site i sur lequel se trouve la table) est situé sur le plus court chemin menant à cette destination. Par exemple, en considérant la figure 1, $routage_5[1]$ est 3.

Exemple Les tables de routage pour la figure 1 sont les suivantes:

	1	2	3	4	5
$Routage_1$	×	2	4	4	4
$Routage_2$	1	×	3	3	3
$Routage_3$	4	2	×	4	5
$Routage_4$	1	3	3	×	3
$Routage_5$	3	3	3	3	×

2 Le routage centralisé et le routage local

Une première solution à envisager pour le calcul des tables est de désigner un site comme *centre de contrôle de routage (CCR)*. Chaque site envoie périodiquement vers le CCR des informations relatives à l'état de ses canaux (encombrement, nouveaux voisins, panne, etc.). Au moyen de ces informations, le CCR calcule une vision approchée de l'état du réseau puis les chemins optimaux en résultant et redistribue à chaque site sa nouvelle table de routage.

Question 1 *Quelle est la conséquence de cette approche sur les liens de communication proches du CCR? Que se passe-t'il si l'état du réseau varie fréquemment (par exemple: modification fréquente des fonctions de coût en fonction du trafic)? Que pouvez vous dire de la robustesse de ce type d'algorithme?*

Question 2 *Discuter de la panne possible du CCR, quels problèmes cela pose t'il? Citez un cas de panne posant les mêmes problèmes.*

Le *routage local* laisse à chaque site, sur la foi d'informations purement locales (état de ses canaux en entrée et en sortie, de sa mémoire, ..), la décision de router un paquet vers tel ou tel autre site.

Question 3 *Donnez un exemple simple d'algorithme de routage local, ainsi que ses caractéristiques. Comparez avec la solution précédente.*

3 Le routage distribué

L'idée est de réaliser une technique de routage combinant les avantages des deux techniques précédentes en limitant les inconvénients. Pour ce faire un algorithme distribué est utilisé qui est exécuté sur chaque site et permet d'échanger des connaissances locales pour arriver à une connaissance approchée de l'état du réseau à un instant donné.

L'algorithme que nous allons étudier repose sur la propriété suivante :

$$\forall i, j : dist_i[j] = \min_{k \in voisins_i} \{c_{ik} + dist_k[j]\}$$

Où $dist_i[j]$ désigne la valeur du meilleur chemin de i à j , c_{ik} le coût associé à l'arc (i,k) et $voisins_i$ l'ensemble des nœuds voisins du nœud i dans le graphe.

Si un site i connaît les valeurs de c_{ik} et de $dist_k[j]$ pour chacun des sites k dans l'ensemble de ses voisins alors il peut déterminer quel est le voisin situé sur le plus court chemin de i à j (autrement dit la valeur de $routage_i[j]$). Sur cette base, on peut bâtir l'algorithme suivant qui fonctionne en trois étapes :

1. Au départ, tout site i positionne la valeur $dist_i[j]$ à c_{ij} si $j \in voisins_i$, il la positionne à ∞ sinon.
2. Chaque site i envoie la valeur de son tableau $dist_i$ à chacun des k tels que $k \in voisins_i$.
3. Lors de la réception d'un tableau $dist_k$ envoyé par un $k \in voisins_i$, le site i effectue le traitement suivant:

change := faux;

pour tout j **tel que** $c_{ik} + dist_k[j] < dist_i[j]$ **faire**

$dist_i[j] := c_{ik} + dist_k[j]$; $routage_i[j] := k$;

change := vrai;

fin pour;

si *change* **alors** effectuer étape 2 **fin si**;

Question 4 Dérouler l'algorithme sur le graphe de la figure 1, en combien de temps les tables deviennent stables ?

Question 5 Une panne survient. La liaison entre le site 3 et le site 4 est interrompue. Quel est le problème rencontré ? On propose alors de réexécuter l'algorithme. Discutez de cette solution.

Question 6 On considère maintenant que l'algorithme n'est pas fini et qu'il suffit de modifier les tables de distances: $dist_3[4] = \infty$ et $dist_4[3] = \infty$. Puis, de simplement réexpédier les nouvelles tables ainsi formées aux voisins. Considérez le cas d'un graphe simple (un réseau $R = (\{A,B,C\}, \{(A,1,B), (B,1,C), (A,1,C)\})$), qu'en pensez vous ?

Question 7 On suppose que tout site peut détecter l'arrivée d'un nouveau voisin (et donc d'une nouvelle liaison) via une primitive adéquate, modifier l'algorithme pour qu'il tienne compte à la fois des ruptures de liaison et de l'arrivée de nouveaux voisins.

4 Hiérarchisation des tables

Il s'agit ici d'étudier un autre problème lié au routage dans les réseaux: le volume des tables de routage. Dans toute cette partie, pour simplifier les raisonnements on considérera des tables de routages simples comme celles décrites dans le premier exemple. Tout en sachant que les tables de routage ont besoin de structures de données beaucoup plus complexes.

Question 8 Dans un réseau de n sites quelle est, au pire, la taille d'une table sur un site ?

Pour éviter que les tables ne soient trop grosses, il est possible de hiérarchiser les tables. Il s'agit de structurer le réseau en domaines. Dans le réseau de la figure 2, si le site A du domaine 1 (noté 1_A) veut envoyer un message aux sites $2_A, 2_B, 2_C$ ou 2_D il devra d'abord envoyer sa requête au site 1_B qui est en charge de la liaison entre le domaine 1 et le domaine 2. On voit ici que le site 1_A n'a à connaître que

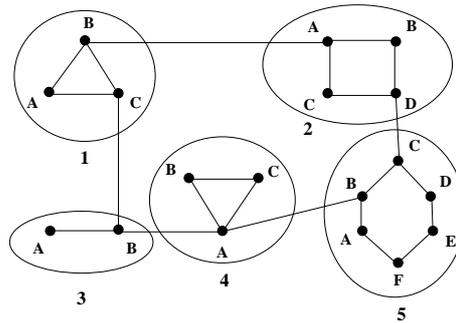


FIG. 2 – Structuration en domaines d'un réseau

le site en charge de la liaison (1,2) pour tous les sites 2_* , d'où une économie au niveau des entrées des tables de routage.

Question 9 En considérant le réseau de la figure 2, donnez les entrées de la table de routage du site 1_A (avec les coûts).

Question 10 Le routage est-il optimal ?

Question 11 Étant donné un réseau composé de n sites, quel est le nombre m de niveaux hiérarchiques que doivent avoir les tables de routages pour être de taille minimale ? On supposera que tous les niveaux ont k entrées. Application numérique: $n = 65536$.