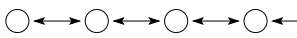
Routage compact dans les arbres

1 Indépendance des noms et facteur d'étirement

On considère une chaîne de machines comme représentée si-dessous :

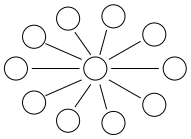


Question 1.1. *En supposant que l'on peut choisir la numérotation des machines, quelle taille de tables de routage et quelle facteur d'étirement peut-on facilement obtenir ?*

On considère désormais l'indépendance des noms : la numérotation des machines est fixée, et n'est pas nécessairement la plus adaptée au routage.

Question 1.2. *Donner un algorithme permettant d'obtenir une taille de table de routage en $O(1)$ et un facteur d'étirement $s \leqslant 8$. Peut-on améliorer ce facteur d'étirement ?*

On considère une topologie en étoile comme sur la figure suivante :



Question 1.3. *Pour avoir un facteur d'étirement de $s < 3$, quelle doit être la taille de table de routage chez le nœud central ?*

Question 1.4. *On cherche à distribuer cette "grosse" table de routage, proposez un algorihme utilisant $\sqrt{n}$ nœuds "relais", tel que la taille maximale d'une table en un nœud donné est $O(\sqrt{n}\log(n))$ et que le facteur d'étirement est $s \leqslant 3$.*

Question 1.5. *Est-il possible de réduire encore la taille de la table de routage en chaque nœud ?*

2 Un schéma de routage pour les arbres très compact

Dans cet exercice, on cherche à améliorer le schéma de routage dans les arbres de Thorup et Zwick étudié en cours. Pour réduire la taille des entêtes des messages, on utilise des subtilités de codage. On change également légèrement la définition des sommets lourds : chaque sommet v qui n'est pas une feuille possède exactement un fils lourd, qui est son fils possédant le plus de descendants (s_v est le nombre de descendants de v, v compris). On note v' le fils lourd de v, et les fils légers sont numérotés par poids décroissant $v_0, v_1, ..., v_{d-1}$ (à savoir $s_{v'} \geqslant s_{v_0} \geqslant ... \geqslant s_{v_{d-1}}$).

On désire réduire la taille de stockage de L_v, qui est la table contenant les numéros de port q des arêtes menant aux sommets légers dans le chemin de la racine r vers v. l_v est le nombre de sommets légers au dessus de v, et $L_v = (q_0, ..., q_{l_v-1})$

Question 2.1. *Définir une affectation des numéros de ports d'un sommet v vers ses fils, de manière à pouvoir majorer le produit des $(q_i + 2)$ par n.*

Plutôt que d'utiliser un mot de taille $\log(n)$ par numéro de port comme dans la version vue en cours, on représente maintenant chaque numéro de port q avec $\lfloor \log(q) \rfloor + 1$ bits, ou un unique bit si $q = 0$, et on concatène ces chaînes de bits.

Question 2.2. *Illustrer sur un exemple de L_v et définir un masque qui permet de séparer les différentes chaînes après concaténation.*

Question 2.3. *Majorer la taille de L_v et du masque M_v. On admettra que*

$$\alpha = \max_{q=1,2,...+\infty} \frac{\lfloor \log(q) \rfloor + 1}{\log(q+2)} \approx 1.20412$$

Question 2.4. *Définir les entêtes des paquets et calculer leur taille globale.*

La dernière étape pour compléter l'algorithme consiste à programmer l'extraction du numéro de port de manière efficace.

Question 2.5. *Proposer une formulation C pour compléter la formule donnée dans le cas du routage de base vu en cours. Il s'agit de remplacer* `L[l]` *dans la formule donnant le numéro de port dans un routage d'un paquet v en un noeud u (on enlève les indices par lisibilité) :*

```
((v>=u && v<h) ?   L[l]   : P[v>=h && v<=f])
```

Ce routage permet donc un décodage très efficace et des entêtes utilisant au plus $c \log_2(n)$ bits (c étant la petite constante obtenue en question 2.3), ce qui devrait être largement supporté, pour des réseaux ayant plus de 2^{37} noeuds pour des protocoles utilisant des adresses IP à 128 bits.

Il est cependant possible de faire encore mieux, mais cela oblige à raffiner un peu le raisonnement.

3 Routage encore plus compact !

Soit T un arbre. On numérote maintenant les numéros de port de façon *canonique* : soit v un sommet de T, de père v' et de fils $v_1, v_2, ..., v_d$. Les fils sont rangés par poids décroissants, à savoir $s_{v_1} \geqslant s_{v_2} \geqslant ... \geqslant s_{v_d}$ (s_u est le nombre de descendants de u, u compris). Alors $port(v, v') = 0$ et $port(v, v_i) = i$ pour $1 \leqslant i \leqslant d$ (en cas d'égalité dans le nombre de descendants, on peut avoir plusieurs numérotations canoniques).

Le principal résultat est le théorème suivant :

Théorème 1. *Il est possible en temps linéaire d'allouer à chaque sommet $v \in T$ une étiquette de taille $(1+o(1))log_2(n)$ bits, notée $etiq(v)$, telle que la connaissance de $etiq(u)$ et $etiq(v)$ permette de déterminer, en temps constant, le numéro de port à emprunter au noeud u pour aller à v sur le plus court chemin de u à v dans T.*

Ce résultat permet d'améliorer encore la complexité du schéma de routage dans les arbres. On utilise un paramètre $b = \lceil \sqrt{\log_2(n)} \rceil$, et un noeud est maintenant lourd si $s_v \geqslant n/b$, léger sinon. Soit T_h le sous-arbre de T comprenant tous les sommets lourds.

Question 3.1. *Que pouvez vous dire sur les fils des noeuds légers ? Montrer que T_h possède au plus b feuilles.*

On partitionne T_h en supprimant les arêtes allant d'un noeud v vers ses fils lorsque v possède plus d'un fils lourd. Les chemins obtenus, appelés *chemins lourds*, peuvent être constitués d'un unique sommet.

Question 3.2. *Majorer le nombre de chemins lourds.*

L'étiquette d'un sommet v est constituée d'un identifiant, qui comprend plusieurs informations sur le chemin lourd sur lequel se trouve v, et d'une mini table de routage de taille $o(\log(n))$. Les détails de l'algorithme sont disponibles dans l'article de Thorup et Zwick intitulé *Compact routing schemes*, et téléchargeable sur la page web du cours :

http://graal.ens-lyon.fr/~abenoit/reso05/