Routage dans les réseaux (k, r)-cellulaires

Youssou Dieng
LaBRI
Université de Bordeaux
dieng@labri.fr

Cyril Gavoille
LaBRI
Université de Bordeaux
gavoille@labri.fr

***Résumé*—Fraigniaud *et al.* [1] et indépendamment Thorup *et al.* [2] ont montré que tout arbre à n nœuds possède un schéma de routage de plus courts chemins utilisant des adresses, des en-têtes et des tables de routage de $O(\log n)$ bits. Il est toujours ouvert de savoir si ce résultat optimal peut être étendu ne serait-ce qu'aux réseaux de largeur arborescente deux dont la meilleure borne connue, due à Peleg [3] est $O(\log^2 n)$ bits.**

Dans cet article, nous étendons ce résultat optimal aux réseaux dits (k, r)-*cellulaires*, c'est-à-dire aux réseaux ne contenant pas $K_{2,r}$ comme mineur et dont les composantes 2-connexes peuvent être rendues planaire-extérieures par la suppression de k sommets. Par exemple, les réseaux (2,4)-cellulaires contiennent les arbres, les graphes planaire-extérieurs et plus généralement tous les graphes sans mineur $K_{2,4}$.

***Index Terms*—routage compact, graphe sans mineur, graphe planaire, graphe planaire-extérieur.**

I. Introduction

La communication point-à-point dans les réseaux d'ordinateurs se fait grâce à une fonction de routage. Cette fonction de routage peut être simple pour des réseaux de taille réduite. Cependant dans les réseaux de grande taille il devient important de réduire la quantité d'informations que chaque nœud doit mettre à disposition de la fonction de routage pour permettre la communication. De même il devient capital d'acheminer les messages par des chemins aussi courts que possible.

Les composants d'un *schéma de routage* sont les tables de routage, les adresses, les en-têtes et un algorithme de routage. La *fonction de routage* prend en paramètre l'en-tête de message arrivant et, en fonction de la table de routage du sommet courant, elle détermine le numéro de port par lequel le message doit être retransmis. L'en-tête de message est généré par le sommet émetteur et reste souvent fixe tout au long du trajet. Elle est générée à partir de sa table de routage et de l'adresse de destination. Dans beaucoup de schémas, l'en-tête se résume à l'adresse de destination.

L'enjeu est donc d'optimiser non seulement la longueur des routes mais aussi l'espace requis par les tables de routage, la taille des en-têtes et des adresses de même que la latence dans chaque routeur.

De nombreux travaux ont été consacrés à la conception de schéma de routage comportant un compromis taille des tables / longueur des routes en passant. Dans le cas d'une topologie quelconque, les tables sont de taille $n^{\Theta(1/s)}$ où s est le *facteur d'étirement*, le ratio maximum sur toutes les paires de nœuds (x, y) entre la longueur de la route et la distance entre x et y. (Voir par exemple [2], [4]–[7] et [8]–[10] pour une introduction au routage compact dans l'Internet). Dans toutes ces solutions, le passage à l'échelle s'avère coûteux lorsque s est très proche de 1, c'est-à-dire lorsque les routes sont presque des plus courts chemins, la taille des tables étant alors polynomiale en la taille du réseau.

En passant par l'étude structurelle fine de la topologie sous-jacente du réseau, il est possible d'obtenir dans certains cas des schémas de plus courts chemins avec des tables de taille poly-*logarithmique* en la taille du réseau. C'est le cas des topologies arborescentes où il a été montré que des tables de $O(\log n)$ bits sont possibles [1], [2]. Notons que ce résultat est à la base des schémas plus généraux de [2] et de [4], [5]. Il est également connu que les réseaux de largeur arborescente au plus t (une généralisation des arbres) possèdent un routage de plus courts chemins avec des tables, adresses et en-têtes de $O(t \log^2 n)$ bits [3]. Pour $t = 1$ (c'est-à-dire le cas des arbres), cette borne n'est pas optimale, et il est toujours ouvert de savoir si $\Theta(\log n)$ bits suffisent pour $t = 2$.

Dans cet article nous proposons d'étendre le résultat des arbres à des réseaux beaucoup plus complexes, les réseaux (k, r)-cellulaires, définis, entre autre, à partir de structure interdite. Comme nous allons le voir les réseaux (k, r)-cellulaires, sans contenir tous les réseaux de largeur arborescente ≤ 2, contiennent les réseaux planaire-extérieurs et même certains réseaux de largeur arborescente r.

II. Préliminaires

Le schéma de routage que nous proposons est de plus courts chemins et ne modifie pas les en-têtes de message qui seront toujours égales à l'adresse de destination. Les réseaux sont modélisés par des graphes simples non orientés et connexes.

Un *mineur* de G est un graphe H obtenu à partir d'un sous graphe de G en contractant des arêtes. Un graphe G est dit sans mineur H si ce dernier n'est pas un mineur de G. On note $K_{p,q}$ le graphe biparti complet avec des parts de taille p et q.

Un graphe est *planaire-extérieur* s'il admet un dessin sur le plan sans croisement d'arêtes où tous les sommets sont sur le bord de la face extérieure. Il est bien connu qu'un graphe planaire-extérieur ne possède pas de mineur $K_{2,3}$.

Un graphe est *k-apex planaire-extérieur* s'il est possible de le rendre planaire-extérieur en supprimant k sommets. Les graphes planaire-extérieurs sont donc 0-apex planaire-extérieurs.

Un graphe est (k,r)-*cellulaire* s'il est sans mineur $K_{2,r}$ et que ses composantes 2-connexes sont k-apex planaire-extérieures, une composante 2-connexe étant un sous-graphe maximal sans point d'articulation.

Les arbres sont sans mineur $K_{2,2}$, et sont donc $(0,2)$-cellulaires. Les planaire-extérieurs sont sans mineur $K_{2,3}$, et donc $(0,3)$-cellulaires. De manière plus intéressante, les graphes sans $K_{2,3}$ sont $(1,3)$-cellulaires[1], et très récemment Dieng *et al.* [11] ont montré que les graphes sans $K_{2,4}$ sont $(2,4)$-cellulaires. On peut montrer que les graphes sans mineur $K_{2,4}$ sont certes de largeur arborescente au plus 4, mais peuvent contenir des structures complexes non-planaires comme K_5 ou $K_{3,3}$. Remarquons enfin que les cliques à $r+1$ sommets (de largeur arborescente r) sont $(r-2,r)$-cellulaires.

La structure (k,r)-cellulaire d'un graphe peut être déterminée en temps polynomial lorsque k et r sont bornés. Il suit que notre schéma de routage peut être construit en temps polynomial.

Pour tout arbre T enraciné en x, on note T_x le sous arbre de T enraciné en x induit par x et ses descendants.

III. L'ALGORITHME DE ROUTAGE

Soit G un réseau (k,r)-cellulaire à n sommets. On suppose que les arêtes ont des poids uniformes. Dans la version étendue de cet article, nous considérons des valuations arbitraires.

Théorème 1: Tout graphe (k,r)-cellulaire possède un schéma de routage de plus courts chemins avec des tables de $O(kr\log n)$ bits, des adresses et des en-têtes de $O(k\log n)$ bits. Le temps de routage est de $O(\log k+\log r)$.

La suite de cet article consistera à montrer ce théorème.

Soit T un arbre de plus courts chemins dans G enraciné en un sommet arbitraire r_T de G.

Pour tout sommet x de G, soit *ROUTE-*$T(x)$ une table de routage de x vers tous les descendants T_x de x dans T.

Pour tout sommet x de G, soit G_x la composante 2-connexe choisie par x de telle sorte que chaque sommet x appartienne à une unique composante 2-connexe. Si x n'est pas un séparateur de G, alors G_x est la composante 2-connexe de G qui le contient ; il n'y en a qu'une. Sinon, si $x=r_T$, alors x choisit une composante 2-connexe arbitraire parmi toutes les composantes 2-connexes qui le partagent.

Sinon, G_x est une composante 2-connexe contenant x et qui est plus proche de la racine r_T.

De ce fait, tout sommet x de G choisit d'appartenir à une unique composante 2-connexe G_x. Il faut aussi souligner que seul x connaît le choix de la composante 2-connexe G_x ; les autres sommets appartenant à des composantes 2-connexes partageant x considèrent x comme étant dans la leur.

[1]Car le seul graphe 2-connexe sans $K_{2,3}$ qui ne soit pas planaire-extérieur est le graphe complet à quatre sommets qui devient planaire-extérieur lorsqu'on supprime un sommet.

Soit *ROUTE0-*$G_x(x)$ une table de routage permettant de router par de plus courts chemins dans G_x vers les destinations de G_x non descendant de x dans T. Ainsi, pour toute destination $y\in G_x$, *ROUTE0-*$G_x(x)[y]$ donne le numéro de port de l'arête menant vers y si y n'est pas descendant de x dans T.

Pour tout sommet $x\in G$ et G_x sa composante 2-connexe, posons *ROUTE-OUT*(x) une table de routage de plus courts chemins de x permettant de router vers toute destination $z\notin G_x$, non descendant de x dans T. Ainsi, pour toute destination $y\notin G_x$, *ROUTE-OUT*$(x)[y]$ donne le numéro de port de l'arête menant vers y si y n'est pas descendant de x dans T.

Pour tout sommet d'articulation x et pour toute destination z tels que x est un séparateur de G_z et G_x (avec $G_z\neq G_x$) stocker dans un champ spécial, *P*(z) de l'adresse de z, l'information permettant à x de router par de plus courts chemins vers z de telle sorte que *P*(z) donne le numéro de port menant vers z à partir de x.

L'algorithme de routage est le suivant :

ROUTE : de $x\in G$ vers $y\in G$:
1. si $y\in T_x$, retourner *ROUTE-*$T(x)[y]$, sinon
2. si $y\in G_x$, retourner *ROUTE0-*$G_x(x)[y]$, sinon
3. si $y\in$*ROUTE-OUT*(x), retourner *ROUTE-OUT*$(x)[y]$, sinon
4. retourner *P*(y)

Lemme 1: L'algorithme de routage *ROUTE* utilise des plus courts chemins dans G.

Preuve. Si $y\in T_x$, alors *ROUTE-*$T(x)[y]$ donne numéro de port correspondant à une arête (x,w) telle que $y\in T_w$. Vu que par construction T est un arbre couvrant G de plus courts chemins, alors (x,w) est une arête d'un plus courts chemins entre x et y dans G.

Sinon, si $y\in G_x$, alors *ROUTE0-*$G_x(x)[y]$ donne le numéro de port correspondant à une arête (x,w) sur un plus courts chemins entre x et y dans G_x car, par construction *ROUTE0-*$G_x(x)$ est une table de routage de plus courts chemins dans G_x. Ce chemin est un plus courts chemins dans G.[2]

Sinon, si $y\in$*ROUTE-OUT*(x), alors y et x n'ont pas choisi la même composante 2-connexe et il n'y a aucune composante 2-connexe qui contient à la fois x et y. Par construction *ROUTE-OUT*$(x)[y]$ donne le numéro de port correspondant à la première arête sur un plus courts chemins entre x et y dans G.

Sinon, alors aucune des trois conditions précédentes n'est vérifiée et x et y partagent une composante 2-connexe et x est un séparateur qui n'a pas choisi d'appartenir à la composante de y. Dans ce cas c'est y qui contient l'information dans le

[2]Car tout chemin sans boucle entre x et y qui utilise un sommet $z\in G_z\neq G_x$ pour $z\notin G_x$ utilise forcément deux sommets d'articulation de G_x, les seuls sommets qui permettent de sortir et de rentrer dans G_x ; $G_x\cup G_z$ est donc une composante 2-connexe ce qui est absurde car par hypothèse G_x est maximal.

champ spécial *P*(y) de son adresse et *P*(y) donne le numéro de port permettant à x de router vers y par un plus court chemin.

Ce qui termine la preuve du lemme 1.

□

IV. Implémentation

D'après [1] et [2], *ROUTE-T*(x) s'implémente avec $O(\log n)$ bits.

Par ailleurs, d'après le résultats ci-après *ROUTE-*G_x s'implémente avec au plus $O(kr \log n)$ bits.

Lemme 2: Tout graphe (k, r)-cellulaire, 2-connexe à n sommets possède un schéma de routage de plus courts chemins avec des tables de $O(kr \log n)$ bits, des adresses et des en-têtes de $O(k \log n)$ bits. Le temps de routage est de $O(\log k + \log r)$.

Preuve. Soit G un réseau (k, r)-cellulaire, 2-connexe à n sommets.

Soit X l'ensemble des sommets tel que $H = G \setminus X$ soit planaire-extérieur. Pour tout sommet x de G, soit *APEX*(x) une table de routage construite à partir d'un arbre T_x de plus courts chemins couvrant G, enraciné en x dans laquelle seuls les nœuds descendants d'au moins un apex sont préservés. Ainsi, pour toute destination y, *APEX*(x)[y] donne le numéro de port de l'arête menant vers y si dans l'arbre T_x choisi par x, le chemin utilise au moins un apex.

Soit T_H un arbre couvrant H, de plus courts chemins. Notons que T_H n'est pas forcément un arbre de plus court chemins dans G. Pour tout sommet x de H, soit *TABLEH*(x) une table de routage pour x de plus courts chemins dans H construite à partir de T_H et d'une table de routage spéciale permettant de router depuis x vers les destinations non accessibles dans T_H par de plus courts chemins. Ainsi, pour toute destination $y \in H$, *TABLEH*(x)[y] donne le numéro de port de l'arête menant vers y selon un plus courts chemins dans H.

Si x est un nœud apex, alors on prend T_x un arbre de plus court chemins enraciné en x, couvrant G. Chaque destination $y \in G$ stocke dans son adresse une table nommée *ROUTER-*T_x de taille k dans laquelle l'entrée i contient le numéro de port qui permet d'aller en y à partir du nœud apex numéro i. Donc pour toute destination y, *ROUTER-*T_x(y) donne le numéro de port de l'arête menant vers y. L'algorithme de routage est le suivant :

ROUTE0 : de $x \in G$ vers $y \in G$:
1. si $y \in$*APEX*(x), retourner *APEX*(x)[y], sinon
2. si x est non-apex, retourner *TABLEH*(x)[y], sinon
3. retourner *ROUTER-*T_x(y)

Lemme 3: ROUTE0 est un algorithme de routage de plus courts chemins.

Preuve. Si y utilise un apex, alors y appartient à un segment qui est codé dans la table de routage *APEX*(x) et *APEX*(x)[y] donne le numéro de port correspondant à l'arête qui relie x au sous arbre qui contient y. Vu que cette arête appartient à un arbre de plus court chemins dans G, alors on route sur un plus courts chemins.

Sinon, alors si x lui même n'est pas un apex, alors le plus courts chemins dans H est aussi un plus courts chemins dans G. Le chemin est déterminé par un schéma de routage de plus courts chemins dans H.

Sinon, alors le sommet x est un apex. Dans ce cas, le numéro de port permettant à x de router vers y est dans la table *ROUTER-*T_x(y) qui se trouve dans l'adresse de y. Étant donné que cette table est construite à partir d'un arbre de plus courts chemins dans G, ce port correspond à une arête sur un plus courts chemins dans G ; ce qui termine la preuve du lemme 3. □

Le schéma de routage est fortement dépendant de la numérotation des nœuds du réseau. Considérons la fonction de numérotation des nœuds de G qui attribue un numéro unique compris entre 1 et n (avec n le nombre de nœud de G) à chaque nœud du réseau de la manière suivante :

1) Numéroter en premier les nœuds apex dans un ordre quelconque
2) Numéroter consécutivement le reste des sommets de G par des entiers de $k + 1$ à n selon un sens de parcours bordant la face extérieure de H.

Nous allons maintenant mettre en place une fonction de codage des tables de routage et des adresses. Cette fonction a pour but de déterminer et de stocker en chaque nœud du réseau les informations utiles à la fonction de routage, qui doit trouver à partir de tout nœud x, le chemin par lequel elle doit faire transiter tout message pour une destination donnée du réseau.

Pour chaque sommet x de H, on notera $add_H(x)$ l'adresse du sommet x dans le routage de plus courts chemins dans H. La taille de $add_H(x)$ est $O(\log n)$ bits, d'après le lemme 4.

L'adresse du sommet x est composée de $(i, ROUTER\text{-}T_x, add_H(x))$, où i est l'indice de x et *ROUTER-*T_x un tableau de taille k dont l'entrée j contient le numéro de port pour venir en x en partant du nœud apex d'indice j. Ainsi l'adresse de chaque sommet est d'au plus $O(k \log n)$ bits.

La taille de la table de routage de x est donnée par les tables $APEX(x)$ et $TABLEH(x)$. La table $TABLEH(x)$ s'implémente comme dans le lemme suivant :

Lemme 4: [12] Tout graphe planaire-extérieur H, possède un schéma de routage de plus courts chemins avec des tables, des en-têtes et des adresses de $O(\log n)$ bits. Le temps de routage est constant.

La table $APEX(x)$ s'implémente de la manière suivante : pour chaque nœud $x \in G$, posons T_x un arbre de plus courts chemins, couvrant G enraciné en x. Pour chaque port p utilisé par un apex u en x, compacter au maximum possible sous

forme de segments[3] d'entiers consécutifs l'ensemble des destinations couvertes par le sous arbre de T_u. Ainsi les segments couverts par T_u sont incompressibles. L'entrée numéro p de la table *APEX*(x) contient l'ensemble des segments utilisant au moins un $apex$ et passant par le port p. La fonction de routage pourra donc déterminer le port p utilisé par y si y est descendant d'au moins un apex.

Étant donné que le coût pour stocker un segment est $2 \log n$ bits, alors déterminer la taille de *APEX*(x) revient à déterminer le nombre de segments en x qui utilisent au moins un apex.

Lemme 5: Pour tout sommet $x \in H$, le nombre de segments en x qui utilisent au moins un apex est au plus $4kr$.

Preuve. En effet, pour tout nœud apex $u \in X$, posons S_u l'ensemble des segment utilisant u et qui sont adjacents à u. Un segment s est adjacent à u si s contient au moins un sommet adjacent à u dans T_x.

Pour mieux dénombrer le nombre de segments de S_u, nous allons le partitionner en deux groupes.

Soit S_u^1 l'ensemble des segments de S_u à partir desquels, on peut atteindre un nœud non couvert par T_u dans H sans être obligé de traverser un autre segment de S_u.

Soit S_u^2 l'ensemble des segments de S_u à partir desquels on est contraint de traverser au moins un autre segment S_u pour atteindre un sommet non couvert par T_u dans H.

Lemme 6: Pour tout nœud x et tout nœud apex u le nombre de segments en x qui sont adjacents à u dans T_x est au plus $2r$. En d'autres termes on a :

1) $|S_u^1| < r$.
2) $|S_u^2| < r$.

Preuve.

1) En effet, supposons que pour un apex u, on a $r \leq |S_u^1|$. A partir de chacun des segments de S_u^1, on peut atteindre dans H un sommet non couvert par T_u sans traverser un autre segment de S_u^1. On en déduit donc dans G un ensemble $P = \rho_1, \rho_2, \ldots, \rho_r$ d'au moins r chemins disjoints de longueurs au moins 2 qui relient u à un ensemble de sommets non couverts par T_u.
   En posant U la composante connexe constituée par $G \setminus T_u$ on a $P = \rho_1, \rho_2, \ldots, \rho_r$ un ensemble d'au moins r chemins de longueur au moins 2 entre u et U ; ce qui induit dans H un mineur $K_{2,r}$. Ceci est bien évidemment absurde par hypothèse. Donc pour tout apex u, on a $|S_u^1| < r$.
2) Supposons que pour un apex u, on a $|S_u^2| < r$.
   On va montrer qu'à partir de chaque segment de S_u^2, on peut atteindre dans H un nœud non couvert par T_u sans traverser un autre segment de S_u^2.
   En effet, soit s_1 un segment de S_u^2 tel que tout chemin dans H reliant s_1 à un sommet $z \notin T_u$ utilise un autre segment s_2 de S_u^2. Alors tout chemin entre s_1 et z est couvert par T_u. A plus forte raison le chemin P entre s_1 et z, contenant s_2 et induit par le bord de la face extérieure du plongement planaire-exterieur de H est aussi couvert par T_u. Par conséquent, $s_1 \cup P \cup s_2$ constitue un seul segment de T_u car consécutifs sur le bord de la face exterieure de H. Ce qui contredit l'hypothèse que les segments de T_u sont incompressibles.
   Par conséquent, à partir de s_1, on peut atteindre un sommet non couvert par T_u sans traverser un segment de S_u^2. En posant $\rho_1, \rho_2, \ldots, \rho_r$, r chemins dont chacun relie un des segments de S_u^2 à un sommet de $G \setminus T_u$ et en considérant la composante connexe $U = T_x \setminus T_u$, alors les chemins $\rho_1, \rho_2, \ldots, \rho_r$ combinés avec les arêtes reliant chaque segment de S_u^2 à u on obtient r chemins disjoints de longueur au moins 2 entre u et U. Nous pouvons donc constituer facilement un mineur $K_{2,r}$ et cela contredirait l'hypothèse. Par conséquent, S_u^2 contient moins de r segments.

Ce qui termine la preuve du lemme 6. □

Donc pour un nœud apex u, le nombre total de segments de S_u est au plus $2r$. Étant donné qu'on a au plus k nœuds apex dans X, alors le nombre total de segments qui utilisent un apex et qui sont adjacent à cet apex est $\sum_{i=1}^{k} |S_{u_i}| < 2kr$.

Calculons maintenant le nombre de segments de T_u non adjacents à u. Soit S l'ensemble des segments de T_u non adjacents à u dans T_u et posons $s_1 \in S$.

Le segment s_1 est connecté à un segment parent s_1' dans T_u par une arête $e \in H$, car sinon s_1 serait adjacent à u dans T_u et contredirait donc l'hypothèse. Les segments s_1 et s_1' ne sont pas consécutifs sur la face extérieure car sinon $s_1 \cup s_1'$ constituerait un seul segment dans T_u. L'arête e aussi n'est pas sur le bord de la face extérieure de H car sinon s_1 et s_1' seraient consécutifs. L'arête e entre s_1 et s_1' partage donc le plan en deux zones dont chaque zone contient un segment non couvert par T_u. En effet, si une des deux zones z était totalement couverte par T_u, alors on aurait $s_1 \cup s_1' \cup z$ qui constituerait un seul segment dans T_u et contredirait le fait que dans T_u les segments sont incompressibles.

Soient z_r la zone qui contient la racine x de l'arbre T_x et $s_3 \notin z_r$ un segment non couvert par T_u. Tout chemin dans G entre la racine x et s_3 soit intersecte e, soit utilise un nœud apex $u' \neq u$[4]. Ce qui veux donc dire que la zone z qui contient s_3 contient aussi un segment s_3' adjacent dans T_x à un apex u' et $u' \neq u$. On en déduit qu'à chaque élément de S, correspond un segment adjacent dans T_x à un apex $u' \neq u$.

Montrons maintenant que deux segments différents de S ne peuvent correspondre à un même segment s_0 de $S_{u'}$ pour un nœud apex $u' \neq u$. En effet, supposons que, 2 segments $s_2, s_4 \in S$ ont le même segment $s_0 \in S_{u'}$ pour un nœud apex $u' \neq u$. Soient s_1 le segment parent de s_2 dans T_u et s_3 le

[3]Un segment est un ensemble de sommets consécutifs obtenu en parcourant les sommets le long de la face extérieure.

[4]Car $s_3 \notin T_u$ par hypothèse.

segment parent de s_4 dans T_u. Vu que la numérotation des nœuds se fait en tournant suivant la face externe du plongement de H, on peut supposer sans perdre de généralité que les nœuds de s_4 sont numérotés en premier, ensuite ceux de s_2 puis ceux de s_0. Si les nœuds numérotés juste après s_4 et juste avant s_2 étaient tous couverts par T_u, alors ils auraient formé avec s_2 et s_4 un seul segment dans T_u, ce qui contredirait l'hypothèse. Il y a donc au moins un segment s'_0 non couvert par T_u tel que s'_0 est numéroté après s_4 et avant s_2.

Par ailleurs, l'arête qui relie s_4 et s_3 dans T_u partage le plan en deux zones dont l'une contient x.

Si x et s'_0 sont dans deux zones différentes, alors tout chemin dans T_x qui relie x à s'_0 utilise un apex $u'' \neq u$ car sinon, $s'_0 \cup s_4 \cup s_2$ constituerait un seul segment dans T_u, ce qui est absurde par hypothèse. En plus, comme $s_0 \neq s'_0$, nous avons montré qu'à chaque couple de segment distincts de S, on peut lui associer un couple de segments distincts non couverts par T_u et dont chacun est adjacents dans T_x à X.

Sinon si x et s'_0 sont dans la même zone, alors la zone z_2 qui ne contient pas x ne peut être accessible à x que si on passe par un apex u''. De plus cette apex $u'' \neq u$ car sinon $z_2 \cup s_4 \cup s_3$ formeraient un seul segment dans T_u ; ce qui est absurde car par hypothèse les segments de T_u sont incompressibles.

On en déduit donc que le nombre de segments dans S est au plus égal au nombre de segments adjacent à X dans T_x c'est à dire au plus $2kr$.

Nous avons donc montré que le nombre de segments en x est au plus $4kr$; ce qui termine la preuve du lemme 5. □

Par conséquent la table *APEX*(x) s'implémente avec au plus $O(kr \log n)$ bits.

Ce qui veut donc dire que cet algorithme utilise des tables de routage de $O(kr \log n)$ bits et des adresses de tailles $O(k \log n)$ bits.

Lemme 7: Le temps de routage est de $O(\log k + \log r)$.

Preuve.

1) Décider si y est un apex est très simple, c'est juste vérifier si l'indice de y est inférieur ou égal à k. C'est une opération qui se fait en temps constant. La table *APEX*(x) contient au plus $4kr$ segments et vérifier l'appartenance de y à un segment se fait en temps constant. Donc vérifier si y appartient à un segment de *APEX*(x) se fait en temps $O(\log kr)$ en utilisant une recherche dichotomique en supposant que les intervalles sont disjoints et ont étés préalablement triés.
2) D'après le Lemme 4, trouver le port de sortie dans *TABLEH*(x) se fait en temps constant.
3) *ROUTER-*T_x est une table à accès directe. Donc lire la valeur de l'entrée i tel que i est l'indice de x se fait en temps constant.

Nous avons donc montré que le temps de routage est en $O(\log k + \log r)$; ce qui termine la preuve du lemme 7.

□

Ce qui termine la preuve du lemme 2. □

Appliquons la fonction de numérotation expliquée dans l'implémentation du lemme 2 à chaque composante 2-connexe de G_x et modifions là de telle sorte que si les sommets d'une composante 2-connexe sont numérotés par des entiers de 1 à j, alors les sommets de la composante 2-connexe suivante seront numérotés à partir de $j+1$ et les sommets déjà numérotés ne seront pas renumérotés si on les rencontre plusieurs fois pendant la numérotation. La numérotation commence par la composante 2-connexe G_{r_T}, en suite traite les autres composantes les un après les autres suivant un parcours en profondeur de l'arbre des composantes 2-connexe.

Ainsi, pour savoir si y est dans la même composante 2-connexe que x, il suffit seulement que x stocke l'intervalle qui contient l'ensemble des sommets de la composante 2-connexe G_x ; $2 \log n$ bits suffisent.

Lemme 8: La table de routage *ROUTE-OUT*(x) s'implémente avec au plus $O(kr \log n)$ bits.

Preuve. En effet, *ROUTE-OUT*(x) contient l'ensemble des destinations non descendantes de x dans T et qui ne sont pas dans la même composante 2-connexe que x. Vu que les sommets appartenant à une composante 2-connexe sont numérotés consécutivement, alors l'ensemble des sommets d'un composante 2-connexe constitue un segment. Ainsi donc, l'ensemble des destinations qui utilisent un sommet d'articulation $\alpha \in G_x$ forme un segment. On peut donc dire que par chaque sommet d'articulation $\alpha \in G_x$ passe au plus un segment. Le port qui permet de router vers le sommet α sert aussi pour router vers ce segment. Vu qu'on sait router vers n'importe quel sommet $y \in G_x$ avec au plus kr segments en x, alors on sait router vers tout sommet d'articulation de G_x avec kr segments. En ajoutant au plus un segment à chaque port utilisé par un articulation, on ajoute l'information utile pour router vers les destinations de *ROUTE-OUT*(x). Si chaque segment en x contient au plus un sommet d'articulation, alors *ROUTE-OUT*(x) contient au plus autant de segment que de sommet d'articulation pour G_x c'est à dire au plus kr segments. Si deux sommets d'articulation de G_x sont dans un même segment en x, alors ces deux sommets d'articulation utilisent le même port. Ainsi, les sommets des deux composantes 2-connexes connectées à chacun d'eux sont numérotés consécutivement, composante après composante. Ainsi, l'union des étiquettes des sommets de ces composantes 2-connexes constitue un seul segment et utilise le même porte. Ceci contredirait l'hypothèse que les segments de *ROUTE-OUT*(x) sont incompressibles ; ce qui est absurde. Cette information permet donc à x de router vers les destinations utilisant un sommet d'articulation de G_x. Cependant, que pouvons nous dire des destinations qui utilisent non pas des sommets d'articulations de G_x mais des sommets d'articulations d'une composante 2-connexe $G_z \neq G_x$ et $x \in G_z$? Il faut que x soit capable de décider entre plusieurs sommets d'articulations de

G_z pour les destinations qui les utilisent. Cette question est à priori difficile à résoudre car x peut être partagé par un nombre non borné de composantes 2-connexes dont chacune contient un nombre non borné de sommets d'articulations.

Il faut remarquer que si x à choisi G_x, c'est parce que dans l'arbre des composantes 2-connexes, G_x est plus proche de G_{r_T} parmi toutes les composantes 2-connexes partageant x. Par conséquent, l'arbre T passe forcément par G_x pour atteindre G_z. Vu que x est le seul sommet de passage entre G_x et G_z, alors x est un ancêtre de tout sommet de G_z et à plus forte raison les destinations qui utilisent des sommets d'articulations de G_z. Ainsi donc les destinations qui utilise un sommet séparateur de G_z sont dans T_x et donc sont dans la table *ROUTE-*$T(x)$.

Par conséquent, le nombre de segments à ajouter dans la table de routage de x pour lui permettre de router vers les destinations utilisant des sommets d'articulations ne peut pas dépasser le nombre de segments nécessaires à x pour router dans G_x ; c'est à dire $4kr$. Nous avons donc montré que $O(kr \log n)$ bits suffisent pour implémenter *ROUTE-OUT*(x). Ceci termine la preuve du lemme 8.

□

En utilisant une recherche dichotomique sur la table *ROUTE-OUT*(x) supposée triée au préalable, nous somme capable de dire si une destination y est dans *ROUTE-OUT*(x) en temps $O(\log k + \log r)$.

En outre l'adresse de y contient un unique champ correspondant au numéro de port à utiliser si x est un sommet d'articulation et y un sommet appartenant à une composante 2-connexe contenant x et non choisie par x.

Tous ces résultats, combinés constituent la preuve du théorème 1.

En utilisant le résultats de [11] montrant que tout graphe 2-connexe sans mineur $K_{2,4}$ est 2-apex planaire-extérieur, donc $(2,4)$-cellulaire, on en déduit :

Théorème 2: Tout graphe $(2,4)$-cellulaire admet un schéma de routage de plus court chemins avec des tables, des en-têtes et des adresses de $O(\log n)$ bits. Le temps de routage est constant.

V. Conclusion

Nous avons étendu le schéma optimal de routage dans les arbres aux réseaux (k,r)-*cellulaires* non valués. Notre schéma de routage utilise des adresses fixes de $O(k \log n)$ bits et des tables pré-calculées de $O(kr \log n)$ bits par sommet. Le temps de routage en un sommet est $O(\log k + \log r)$.

Notons que ce résultat est fortement dépendant du schéma de routage dans les réseaux planaire-extérieurs et les valuations des arêtes sont supposées être uniformes. Notre schéma peut être adapté aux graphes de valuations arbitraires moyennant un algorithme et une implémentation plus sophistiqués pour le routage dans les graphes planaire-extérieurs (car le schéma de routage dans les graphes planaire-extérieurs ne marche que pour des valuations uniformes).

Une autre amélioration de ce résultats serait de diminuer la taille des adresses. Avec un algorithme et une implémentation plus sophistiqués dans les graphes planaire-extérieurs, nous montrons que les adresses peuvent être réduites à seulement $O(\log r \log n)$ bits.

Références

[1] P. Fraigniaud and C. Gavoille, "Routing in trees," in 28^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, F. Orejas, P. G. Spirakis, and J. v. Leeuwen, Eds., vol. 2076 of Lecture Notes in Computer Science. Springer, Jul. 2001, pp. 757–772.

[2] M. Thorup and U. Zwick, "Compact routing schemes," in 13^{th} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. ACM Press, Jul. 2001, pp. 1–10.

[3] D. Peleg, "Proximity-preserving labeling schemes," *Journal of Graph Theory*, vol. 33, no. 3, pp. 167–176, 2000.

[4] I. Abraham, C. Gavoille, and D. Malkhi, "On space-stretch trade-offs : Upper bounds," in 18^{th} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. ACM Press, jul 2006, pp. 207–216.

[5] I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, and M. Thorup, "Compact name-independent routing with minimum stretch," *ACM Transactions on Algorithms*, vol. 3, no. 4, p. Article 37, Jun. 2008.

[6] L. J. Cowen, "Compact routing with minimum stretch," *Journal of Algorithms*, vol. 38, no. 1, pp. 170–183, 2001.

[7] D. Peleg and E. Upfal, "A trade-off between space and efficiency for routing tables," *Journal of the ACM*, vol. 36, no. 3, pp. 510–530, Jul. 1989.

[8] M. Dom, "Compact routing," in *Algorithms for Sensor and Ad Hoc Networks*, vol. 4621 of Lecture Notes in Computer Science. Springer, 2007, pp. 187–202.

[9] D. Krioukov, K. Claffy, K. Fall, and A. Brady, "On compact routing for the internet," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 43–52, Jul. 2007.

[10] C. Gavoille, "Routing in distributed networks : Overview and open problems," *ACM SIGACT News - Distributed Computing Column*, vol. 32, no. 1, pp. 36–52, Mar. 2001.

[11] Y. Dieng and C. Gavoille, "La structure des graphes sans mineur $K_{2,4}$," in $10^{èmes}$ *Journées Graphes et Algorithmes*, Nov. 2008.

[12] ——, "Routage dans les graphes cellulaires," in $9^{èmes}$ *Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (AlgoTel)*, May 2007, pp. 91–94.