Routage basé sur ancre dans les réseaux à large échelle auto-organisables

A. C. Viana[1,2], M. D. de Amorim[1], S. Fdida[1] et J. F. de Rezende[2]

[1] *LIP6 – UPMC – 8, rue du Capitaine Scott – 75015 – Paris – France*
`{viana,amorim,sf}@rp.lip6.fr`

[2] *COPPE/EE - UFRJ – P.O. Box 68504 – 21945-970 – Rio de Janeiro – RJ – Brasil*
`rezende@gta.ufrj.br`

RÉSUMÉ. *Le découplage entre l'identifiant et l'adresse topologique d'une entité permet aux systèmes communicants de faire face, d'une manière simple, au facteur d'échelle. Cet article présente Tribe, une stratégie de routage indirect pour les réseaux spontanés exigeant l'auto-organisation des nœuds. Dans le modèle de service indirect de Tribe, la communication est basée sur des points de rendez-vous, où le routage est exécuté en deux phases : la recherche de la localisation d'une destination et la communication directe avec cette destination. Dans cette approche, l'adresse topologique courante d'un nœud est stockée dans un autre nœud qui joue le rôle d'une ancre. Les ancres sont responsables pour traduire l'identifiant d'un nœud en une adresse dépendante de la topologie. Tribe est une solution originale, robuste, autonome et capable de gérer les départs et les arrivées dynamiques de nœuds.*

ABSTRACT. *Decoupling the permanent identifier of a node from the node's topology-dependent address is a promising approach toward completely scalable self-organizing networks. This paper proposes Tribe, an indirect routing strategy for wireless self-organizing networks. The Tribe indirect service model is instantiated as a rendezvous-based communication abstraction, where routing is performed in two phases: search for a node's address based on the node's identifier and the direct communication with this node. Any node in Tribe can play the role of an anchor node. Anchor nodes are responsible for storing the location information of other nodes in the topology, and for translating a node's identifier into its topology-dependent address. The indirect routing strategy used by Tribe allows the network to support dynamic joins and leaves and to scale to arbitrarily large self-organizing topologies.*

MOTS-CLÉS : *Routage, auto-organisation, tables de hachage distribuées.*

KEYWORDS: *Routing, self-organization, distributed hash tables.*

1. Introduction

L'évolution de l'Internet vers une architecture mobile et sans fil exige une nouvelle stratégie de routage dans les réseaux dynamiques à large échelle . Dans ce scénario, la mobilité de nœuds invalide l'information de routage. Cette information ne peut plus être incluse dans l'adresse qui identifie le nœud dans le réseau. Ainsi, la mise à jour fréquente de l'adressage suggère la proposition d'une infrastructure de réseau où l'identifiant permanent du nœud est découplé de son adresse dépendante de la topologie.

Dans ce papier, nous détaillons le protocole Tribe, une stratégie de routage indirecte dans les réseaux spontanés sans fil exigeant l'auto-organisation des nœuds. La communication dans Tribe est basée sur une abstraction de point de rendez-vous. Dans ce modèle, le routage est exécuté en deux phases : la recherche de la localisation de la destination, basée sur son identifiant, et la communication directe avec cette destination, basée sur son adresse topologique. Une fonction de hachage traduit l'identifiant permanent de la destination en une adresse virtuelle valable dans un espace d'adressage virtuel. Les sources utilisent l'adresse virtuelle de la destination afin d'identifier le nœud responsable pour stocker l'adresse topologique de la destination.

Tribe distribue l'information de localisation des nœuds sur toute la topologie et identifie les nœuds qui jouent le rôle de point de rendez-vous et qui sont responsables pour stocker et maintenir cette information à jour – les nœuds *ancres*. N'importe quel nœud dans la topologie de Tribe peut jouer le rôle d'un nœud ancre.

Des projets comme Terminodes [HUB 01, BLA 01] et Grid [Gri , CHA 02] proposent une distribution d'information similaire à celle de Tribe, mais se servent du positionnement géographique des nœuds pour la création de la topologie et le routage des messages. Tribe est complètement indépendant de n'importe quelle information géographique.

Tribe établit une représentation virtuelle du réseau qui reflète la localisation relative des nœuds par rapport à son voisinage. Les nœuds voisins dans la topologie virtuelle sont aussi des voisins dans le réseau physique. Nous utilisons le terme "virtuel" pour indiquer que le système d'adressage employé correspond à la traduction d'une adresse dans l'espace d'adressage réel – l'identifiant universel – en une adresse dans l'espace d'adressage virtuel – l'adresse dépendante de la topologie.

Dans Tribe, l'arrivée et le départ d'un nœud n'affectent pas la structure globale de la topologie. Le mécanisme d'attribution d'adresses est le résultat de la coopération d'un petit groupe de nœuds. De plus, en connaissant l'identifiant universel d'une destination, le routage est exécuté de manière déterministe. Chaque nœud dans Tribe ne stocke que des informations sur lui-même, sur ses voisins et sur les nœuds sous sa responsabilité. Ces informations sont suffisantes pour la mise en œuvre de la procédure de routage, laquelle est effectuée saut à saut. Chaque nœud achemine les messages vers au voisin le plus proche possible de la destination. Ainsi, chaque nœud maintient une table de routage avec $O(k)$ lignes (où k est le nombre de voisins du nœud) et un coût de communication de routage en $O(1)$. Le nombre de voisins et, par conséquent la surcharge de signalisation dans le réseau, dépendent seulement de la zone de couverture d'un nœud et est indépendant du nombre total de nœuds dans le système.

Tribe a été spécialement conçu pour les réseaux à faible mobilité. Dans les scénarios à très forte mobilité, l'acheminement géographique [FIN 87, KAR 00, LI 00, WOO 01] et les protocoles ad hoc traditionnels (AODV [PER 02] et DSR [JOH 02] par exemple) seraient mieux adaptés. Dans Tribe, la mobilité provient des nœuds qui rejoignent ou quittent volontairement le réseau. Dans cette situation, il est fondamental de maintenir la stabilité globale du système et la cohérence de la procédure de routage. Ceci est assuré par un mécanisme d'attribution d'adresses qui garantit le passage à échelle d'une façon complètement adaptative.

La suite de cet article détaille le protocole Tribe. Dans la Section 2 nous décrirons une vue d'ensemble de l'architecture. Le modèle indirect de service, la communication basée sur l'abstraction de point de rendez-vous et le fonctionnement de Tribe seront également présentés. Les propriétés de l'infrastructure réseau, les procédures et le fonctionnement du protocole sont également discutés. La mobilité des nœuds sera traitée dans la Section 3 tandis que les travaux qui y sont liés seront discutés dans la Section 4. Finalement, la Section 6 conclura cet article.

2. Tribe : généralités

Dans cette section nous présentons les principales caractéristiques de Tribe. Nous présentons d'abord le modèle de service et l'abstraction de la communication, puis la conception du protocole.

2.1. *Modèle de service indirect*

Le routage indirect est au cœur du protocole Tribe. Une procédure de routage est indirecte lorsqu'elle est exécutée en deux phases. Ceci permet au réseau de découpler les informations sur la localisation d'un nœud de la localisation elle-même.

Les avantages de cette technique sont multiples. Les nœuds ne sont pas obligés de stocker des informations de routage de tous les autres nœuds du système. Chaque nœud est responsable uniquement pour une partie des informations de la localisation des nœuds existants. Ces informations sont complètement distribuées parmi les éléments du réseau – le passage à l'échelle est donc assuré. Un autre avantage de cette technique est son faible coût de maintenance. Chaque nœud maintient sa localisation exacte mise à jour uniquement chez son nœud ancre, et uniquement chez celui-ci. Si le nœud se déplace, il doit informer sa nouvelle localisation à son nœud ancre.[1]

1. Il est clair qu'il faut également que l'identité d'un nœud soit connue par ses voisins. De toute façon, cet échange se fait naturellement (environnement sans fil) et ne compromet pas le fonctionnement global du réseau.

2.2. *Communication basée sur des points de rendez-vous*

La communication entre deux nœuds est basée sur l'abstraction de point de rendez-vous. Les nœuds nommés *nœuds ancres* jouent le rôle de points de rendez-vous et sont responsables du stockage de l'information de localisation d'autres nœuds. La procédure de routage est donc le résultat de la convolution de deux autres fonctions : la localisation de la destination et la communication.

Un nœud x possède trois identifiants. Le premier, appelé identifiant universel U_x, représente l'identité du nœud. Cet identifiant est connu de tous les nœuds du système (au moins des nœuds susceptibles de communiquer avec x) et est complètement indépendant d'un adressage au niveau réseau. L'identifiant universel peut être un nom, un numéro voir même une adresse IP.

Le deuxième identifiant est l'adresse virtuelle V_x, obtenue à partir de la traduction de U_x en une valeur dans un espace d'adressage $\mathcal{V}$ connu. La fonction de traduction est également rendue publique et, de cette manière, connue de tous les nœuds. Le rôle de cet identifiant est d'acheminer une requête de localisation vers le nœud ancre du nœud recherché. Le dernier identifiant est l'adresse relative du nœud dans le réseau, E_x. Cette adresse est dépendante de sa localisation et est obtenue par l'intermédiaire du voisinage temporaire du nœud. L'utilisation de l'adressage relatif est le seul moyen d'acheminer des données. Un point important à remarquer est que l'adresse relative change si le nœud se déplace, mais les identifiants universels et virtuels sont fixes. La figure 1 illustre le fonctionnement de base du processus de routage indirect.

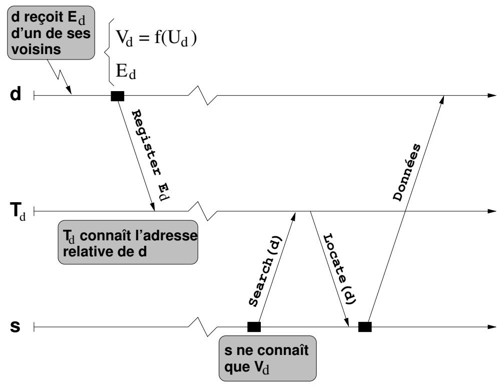


Figure 1. *Principe du routage indirect.*

Soit s le nœud voulant établir une communication avec le nœud d. Tout au début, s ne sait pas comment atteindre d car la localisation de d est inconnue. Le nœud s doit donc contacter le nœud qui connaît cette localisation, c'est-à-dire, le nœud ancre de d. Appelons ce nœud T_d. T_d connaît la localisation de d simplement parce que d la lui avait informée. Autrement dit, T_d est le point de rendez-vous entre d et n'importe quel autre nœud voulant contacter d. La particularité de cette approche est que les points de rendez-vous sont distribués et dépendants uniquement de l'identifiant de chaque nœud. A priori, ils sont inconnus des sources. Après avoir reçu la demande de s, le nœud T_d répond en envoyant un message contenant la localisation actuelle de d. La communication entre s et d peut maintenant être établie.

2.3. *Fonctionnement de Tribe*

Tribe établit une infrastructure réseau qui reflète la localisation relative des nœuds par rapport à leur voisinage. Il spécifie également un protocole distribué qui permet aux nœuds de contrôler des régions indépendantes appartenant à un espace virtuel. Les nœuds qui sont physiquement proches dans le réseau contrôlent également des régions proches dans la topologie virtuelle de Tribe. À tout moment, chaque nœud est responsable pour une région $\mathcal{R} = [R^{\ominus}, R^{\oplus}]$ de l'espace d'adressage virtuel $\mathcal{V}$. Nous définissons $\mathcal{V}$ comme l'ensemble des entiers compris entre 0 et $(2^m - 1)$. La borne inférieure de la région de contrôle $R^{\ominus}$ identifie d'une façon unique le nœud dans la topologie. Cette borne est l'adresse relative E du nœud. Ainsi, $E = R^{\ominus}$ décrit la position relative du nœud selon son voisinage.

2.3.1. *Arrivée d'un nœud*

À l'arrivée d'un nœud, celui-ci met en marche un algorithme de *bootstrap* pour identifier ses voisins. Comme l'environnement en question est du type sans fil, les voisins d'un nœud sont tous ceux qui se situent dans sa zone de couverture. L'un de ses voisins lui donnera une partie de sa propre région de contrôle. La région de contrôle reçue servira à deux buts : l'identification du nœud et le routage. Soit $t = t_0^i$ l'instant quand le nœud i reçoit la région de contrôle $\mathcal{R}_i(t_0^i)$. Ainsi, $\mathcal{R}_i(t_0^i) = [R_i^{\ominus}(t_0^i), R_i^{\oplus}(t_0^i)] \subseteq [0, 2^m)$. Cette région est nommée la *région de contrôle initiale*[2] du nœud i. La borne inférieure de la région de contrôle reçue, $R_i^{\ominus}(t_0^i)$ est l'adresse relative E_i du nœud i et l'identifie d'une façon unique dans la topologie.

L'exécution du algorithme de *bootstrap* donne comme résultats l'ensemble de voisins, $\mathbf{A}_i$, et de la liste de voisins $\mathbf{L}_i$. La *liste de voisins* $\mathbf{L}_i$ du nœud i est définie comme étant $\mathbf{L}_i = \{[E_{a_1}, \mathcal{R}_{a_1}, \mathcal{R}_{a_1}(t_0^{a_1})], \ldots, [E_{a_k}, \mathcal{R}_{a_k}, \mathcal{R}_{a_k}(t_0^{a_k})]\}$, où $\mathcal{R}_{a_j}$ est la région sous la responsabilité du nœud a_j et $\mathcal{R}_{a_j}(t_0^{a_j})$ est la région de contrôle initiale de a_j. La liste de voisinage est employée pour déterminer lequel des voisins partagera sa région de contrôle.

Si le nombre d'éléments dans $\mathbf{A}_i$ est égal ou plus grand que deux, le système doit appliquer une méthode pour décider lequel des voisins partagera sa région avec le nouvel arrivant. En premier lieu, le nœud qui vient d'arriver recevra la région du

2. Pour une question de simplicité de notation, on utilise $[X, Y)$ au lieu de $[X, Y - 1]$.

voisin qui possède la plus grande région de contrôle. Ce dernier sera son *voisin père*. En second lieu, la région attribuée sera la moitié supérieure de la région du voisin père. Ainsi, le voisin père p_i est choisi de façon à ce que $S(\mathcal{R}_{p_i}) = \max\{S(\mathcal{R}_{a_j})\}, \quad p_i \in \mathbf{A}_i, \forall a_j \in \mathbf{A}_i$, où $S(\mathcal{R}_{a_j})$ est la taille de la région de a_j. En une forme générale, $S(\mathcal{R}_z) = R_z^{\oplus} - R_z^{\ominus}$.

Si $\mathbf{A}_i = \{\}$, le nœud qui vient d'arriver est aussi l'unique nœud dans le réseau et recevra la région de contrôle $\mathcal{R}_{a_j}(t_0^{a_j}) = [0, 2^m)$.

2.3.2. *Enregistrement d'un nœud*

Après avoir reçu sa région de contrôle et son adresse relative, le nouvel arrivant (le nœud n) doit s'enregistrer auprès de son *nœud ancre*, T_n. Il est important de noter encore une fois que n ne connaît pas la vraie identité de T_n – les nœuds ancres sont découverts à la demande. Par ailleurs, le fait qu'un nœud n'ait pas besoin de connaître son nœud ancre est le principe fondamental qui fait de cette approche une architecture auto-organisable et autonome. Le nœud calcule, à partir de son identifiant universel, son adresse virtuelle. Le principe est équivalent à celui de la procédure de localisation d'un nœud. D'accord le procédure de routage (Section 2.5), un message contenant son adresse *relative* sera acheminé saut à saut jusqu'à ce qu'ils trouvent le nœud dont la région de contrôle contient l'adresse virtuelle de n.

Lorsqu'un nœud désire communiquer avec n, il doit d'abord demander à T_n quelle est la localisation actuelle de n. Le principal avantage de cette technique est que T_n est une entité logique, c'est-à-dire, le nœud qui possède la région contenant l'adresse virtuelle de n. T_n peut changer dans le temps sans que n ou celui qui veut communiquer avec n soit importuné. Il suffit que, lors d'un changement de nœud ancre, un autre nœud s'occupe de la région de contrôle abandonnée. Ainsi, le nœud s calcule, à partir du identifiant universel du nœud n, l'adresse virtuelle vers le nœud dont la région de contrôle contient V_n. En recevant la demande de s, T_n répond en envoyant la localisations actuelle de n, E_n.

2.4. *Gestion des régions dans Tribe*

Nous définissons tout d'abord quelques propriétés liées à la manipulation des régions de contrôle. L'*ensemble des fils* $\mathbf{C}_i = \{c_1, \ldots, c_k\}$ est l'ensemble de k nœuds qui ont comme *voisin père* le nœud i, c'est-à-dire, $p_{c_k} = i, \forall c_k \in \mathbf{C}_i$. Ainsi, pour le voisin père du nouvel arrivant n (p_n) l'ensemble des fils de p_n est représenté par $\mathbf{C}_{p_n}$.

À l'arrivée de p_n ($t = t_0^{p_n}$), son ensemble des fils est $\mathbf{C}_{p_n}(t_0^{p_n}) = \{\}$. Pour $t = \tau > t_0^{p_n}$, l'ensemble des fils sera $\mathbf{C}_{p_n}(\tau) = \{c_1, \ldots, c_k\}$. Nous considérons que les éléments de $\mathbf{C}_{p_n}$ sont organisés de telle façon à ce que c_i est un nœud arrivé dans le réseau avant le nœud c_j, $\forall i < j$, c'est-à-dire $t_0^{c_i} < t_0^{c_j}$. Ainsi :

1. $R_{c_1}^{\ominus} > \ldots > R_{c_{i-1}}^{\ominus} > R_{c_i}^{\ominus} > R_{c_{i+1}}^{\ominus} > \ldots > R_{c_k}^{\ominus}$.
2. $\mathcal{R}_{c_i}(\tau) \subset \mathcal{R}_{p_n}(t_0^{p_n}), \quad \forall c_i \in \mathbf{C}_{p_n}(\tau)$.

3. $R^{\oplus}_{p_n}(\tau) = \min\{R^{\ominus}_{c_i}(\tau)\} - 1, \forall c_i \in \mathbf{C}_{p_n}(\tau)$.

Soit $t_0^{c_i} > t_0^n$ l'instant d'arrivée du nœud c_i. Nous avons donc :

4. $\sum_{\forall c_i \in \mathbf{C}_{p_n}} \mathcal{R}_{c_i}(t_0^{c_i}) \subset \mathcal{R}_{p_n}(t_0^{p_n})$.

Nous utilisons également la notion de *descendants* d'un nœud i en référence à tous les nœuds dont l'adresse relative se trouve dans la région de contrôle initiale de i. Ainsi, nous définissons $\mathbf{D}_i = \{d_1, d_2, \ldots, d_k\}$ comme étant l'ensemble des k descendants de i tels que $\mathcal{R}_{d_i}(\tau) \subset \mathcal{R}_i(t_0^i)$. De la même manière, nous avons $\mathbf{C}_i \subseteq \mathbf{D}_i$.

2.5. *Procédure de routage*

Un nœud n, après son enregistrement, connaît les adresses relatives de tous ses voisins et leurs régions de contrôle initiales (selon $\mathbf{L}_n$). Ces informations sont suffisantes pour router un message et composent les tables de routage des nœuds. Il convient de rappeler que n maintient également l'information sur sa région de contrôle initiale, $\mathcal{R}_n(t_0^n)$.

Supposons que n reçoit un message de s et qu'il doit le transmettre à d. Afin d'acheminer le message, le nœud n vérifie d'abord si l'adresse de la destination correspond à un de ses descendants dans l'ensemble $\mathbf{D}_n$. Si la condition $E_d \in \mathcal{R}_n(t_0^n)$ est respectée, le nœud n conclut que le nœud d est un de ses descendants, c'est-à-dire, $\mathcal{R}_d(\tau) \subset \mathcal{R}_n(t_0^n)$, $t_0^n < \tau$. Ensuite, n achemine le message à son fils c_i de sorte que $E_{c_i} = \max\{E_{c_k}\}$, $E_{c_k} \leq E_d$ et $c_k \in \mathbf{C}_n(\tau)$. Néanmoins, si $E_d \notin \mathcal{R}_n(t_0^n)$, le nœud n envoie le message au nœud p_n.

D'un point de vue pratique, un nœud peut avoir plus de voisins immédiats que son père ou ses fils. Il n'y a aucun "lien familial" entre eux, mais ces liens, inhérents aux environnements sans fil, peuvent constituer des "raccourcis" au moment d'acheminer des messages. Nous définissons comme l'*ensemble de nœuds restants* $\mathbf{S}_i = \{s_1, \ldots, s_k\}$, l'ensemble de k nœuds qui sont des voisins immédiats du nœud i mais avec lesquels i n'a aucun lien familial. Ainsi, $\mathbf{S}_i = \{\mathbf{A}_i - (\mathbf{C}_i \cup \{p_i\})\}$.

Soit $\mathbf{S}_n$ l'ensemble de nœuds restants. Supposons que n doit envoyer un message à la destination d. Si $E_d \notin \mathcal{R}_n(t_0^n)$, donc le nœud n doit envoyer le message à son voisin père ou à l'un des nœuds restants $s_j \in \mathbf{S}_n$. Soit $\mathbf{U}_n = \mathbf{S}_n \cup \{p_n\}$.

Le nœud n vérifie si l'adresse de la destination correspond à un de ses descendants. Dans ce cas, n achemine le message à l'un de ses voisins immédiats u_k dont $E_d \in \mathcal{R}_{u_k}(t_0^{u_k})$, pour $u_k \in \mathbf{U}_n$. Sinon, le message sera acheminé à p_n.

Selon ces conditions et selon le mécanisme d'attribution de régions, un nœud d peut se retrouver dans l'une des situations suivantes :

– Il est un descendant du nœud u_k, c'est-à-dire, $\mathcal{R}_d(\tau) \subset \mathcal{R}_{u_k}(t_0^{u_k})$, $t_0^n < \tau$.

– Il est un héritier du nœud n ou appartient à une autre famille.

Le nombre de voisins immédiats et, par conséquent, la surcharge de signalisation dans le réseau ne dépendent que de la zone de couverture de chaque nœud et sont indépendants du nombre total de nœuds dans le système. Par ailleurs, une quantité limitée d'informations suffit pour acheminer des messages dans l'espace d'adressage relatif. Tribe limite à $O(k)$ la quantité de lignes de la table de routage dans chaque nœud, où k est le nombre de voisins immédiats d'un nœud. Les message avec une certaine adresse V ou E sont acheminés saut à saut jusqu'à ce qu'ils trouvent le nœud dont la région de contrôle contient l'adresse recherchée.

2.6. *Exemple*

Afin de mieux comprendre les procédures exécutées par le protocole Tribe, nous prendrons l'exemple de la figure 2. Cette figure représente une topologie dont l'espace d'adressage est de 128 bits, $\mathcal{V} = [0, 2^{128} - 1]$, et dont les lignes pleines représentent les liens père-fils. La figure 2(a) montre également l'arrivée du nœud n et sa zone de couverture. Après l'exécution de l'algorithme de *bootstrap*, n identifie comme voisins immédiats les nœuds a_j et a_i ayant comme résultat la liste de voisins $\mathbf{L}_n = \{(2^{103}, [2^{103}, 2^{110}), [2^{103}, 2^{110})), (2^{110}, [2^{110}, 2^{128}), [2^{110}, 2^{128}))\}$. Dans ce cas, comme $S(\mathcal{R}_{a_j}(\tau)) > S(\mathcal{R}_{a_i}(\tau))$, le nœud a_j partage sa région avec n et devient p_n.

Le nœud a_j informe ses voisins immédiats sur sa nouvelle région de contrôle et le nœud n s'enregistre auprès de son nœud ancre, T_n. En supposant que $V_n = 2^{100}$, son nœud ancre sera le nœud $E_{T_n} = 2^{96}$ car sa région de contrôle contient V_n.

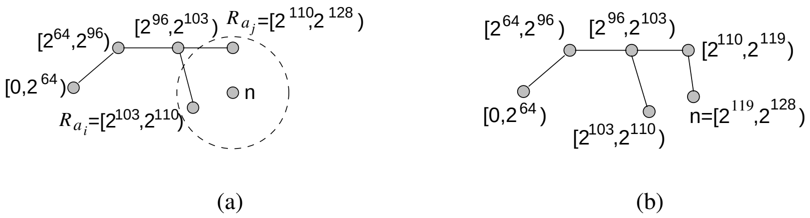


Figure 2. *Exemple d'une topologie Tribe.*

3. Mobilité

Tribe est également capable de traiter la mobilité des nœuds qui rejoignent ou quittent le réseau. Lors du départ d'un nœud, le système doit garantir que la région de contrôle abandonnée soit prise en charge par un nœud restant. Normalement, chaque nœud devrait contrôler une seule région. Néanmoins, à cause du dynamisme de la topologie, ceci est difficile à obtenir. Souvent, la région de contrôle du nœud partant et celle de son voisin père ne sont pas contiguës. Comme nous allons voir dans cette section, la discontinuité des régions peut être la cause d'un faux acheminement de messages. Pour ces mêmes raisons, certains nœuds peuvent devenir inaccessibles.

Le but de cette partie est de montrer les propriétés des topologies qui utilisent Tribe et comment ces propriétés sont maintenues avec la mobilité des nœuds.

3.1. *Continuité de la topologie*

La stabilité du système, et par conséquent celle du protocole de routage, est assurée par la continuité des régions à l'intérieur de la topologie. Une partie de la topologie est dite *continue* si et seulement si la *loi de la continuité des régions* est respectée au sein de cette aire.

Définition 1 *(Loi de la continuité des régions) Un groupe de nœuds forment une région continue si les messages échangés entre ces nœuds sont correctement délivrés au travers de l'opération normale du protocole Tribe.*

Afin de mieux analyser les situations de discontinuité, considérons que n quitte sa position dans le réseau. Nous appelons *orphelins*, représentés par $\mathbf{O}_n = \{o_1, \ldots, o_k\}$, l'ensemble de k enfants du nœud n après le départ de n. D'autre part, $\mathbf{O}_n \subset \mathbf{A}_n$ et $\mathbf{O}_n = \mathbf{C}_n$.

Rappelons qu'un nœud achemine les messages au voisin immédiat dont la région de contrôle s'approche le plus de la région de contrôle qui contient l'adresse de la destination. Supposons que n n'a pas été le dernier fils de p_n à rejoindre le réseau. Ainsi, $\{\exists c_j \in \mathbf{C}_{p_n} \mid R^{\ominus}_{c_j} < R^{\ominus}_{n}, c_j \neq n\}$.

Selon les propriétés de manipulation des régions définies dans la section 2.4, nous avons $R^{\ominus}_{c_1} > \ldots > R^{\ominus}_{c_{i-1}} > R^{\ominus}_{c_i} > R^{\ominus}_{c_{i+1}} > \ldots > R^{\ominus}_{c_k}$. Ainsi, c_i est un nœud qui a rejoint le réseau avant le nœud c_j, $\forall i < j$. Considérons que n est le $i$ème fils de p_n. De cette façon, pour $\mathbf{O}_n = \{o_1, \ldots, o_k\}$, nous avons :

$$R^{\ominus}_{c_{i-1}} > R^{\ominus}_{o_j} > R^{\ominus}_{c_{i+1}}, \forall o_i \in \mathbf{O}_n.$$

Soit $\hat{a}$ le dernier fils de n à joindre le réseau, c'est-à-dire, $\mathcal{R}_{\hat{a}}$ et $\mathcal{R}_n$ forment une région continue ($R^{\oplus}_{n} = R^{\ominus}_{\hat{a}} - 1$). Au départ de n, c_{i+1} est le fils de p_n dont l'adresse relative $E_{c_{i+1}}$ est en même temps inférieure et la plus proche possible de l'adresse relative $E_{\hat{a}}$. Si c_{i+1} n'est pas voisin immédiat de $\hat{a}$, c'est-à-dire, $c_{i+1} \notin \mathbf{A}_{\hat{a}}$, p_n acheminera à c_{i+1} tous messages adressés à $\forall o_i \in \mathbf{O}_n$.

La Fig. 3 montre un exemple de la situation décrite ci-dessus. Avant le départ de n, les messages adressés à $\hat{a}$ sont correctement acheminés (Fig. 3(a)). Dans cet exemple, le nœud $E_s = 2^{192}$ représente la source. L'adresse relative de la destination est $E_{\hat{a}} =$

2^{244}. La Fig. 3(b) montre la topologie après le départ de n. Lorsque p_n vérifie ses descendants afin d'acheminer un message adressé à $\hat{a}$, il a deux choix de route : c_i ($E_{c_i} = 2^{228}$) ou c_j ($E_{c_j} = 2^{232}$). Étant donné que c_j est, comparé à c_i, le nœud dont la région de contrôle s'approche le plus de l'adresse relative de la destination, c_j est choisit comme prochain saut. Néanmoins, il est clair que le nœud choisit aurait dû être c_i.

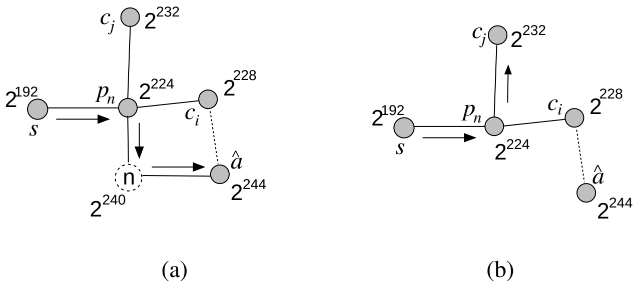


Figure 3. *Orphelins inaccessibles en raison de la discontinuité de la topologie.*

Nous proposons dans la suite un mécanisme de réattribution de régions qui garantit l'exécution correcte du protocole de routage après le départ d'un nœud. En permettant aux nœuds de contrôler plus d'un région d'une façon adaptative, l'opération globale du protocole est maintenue.

3.2. *Réattribution souple*

Une opération de réattribution de région est dite souple si le voisin père du nœud partant p_n est également voisin de tous les enfants de n. Dans ce cas, n rend sa région de contrôle à p_n. Autrement dit, une réattribution souple peut être exécutée si les conditions suivantes sont respectées :

Condition 1 $R^{\oplus}_{p_n} = R^{\ominus}_n - 1$.

Condition 2 $\mathbf{O}_n \subset \mathbf{A}_{p_n}$.

3.3. *Réattribution par clones*

Lorsque les conditions 1 et 2 ci-dessus ne sont pas respectées, un mécanisme supplémentaire doit être mis en place pour garantir la continuité des régions. Dans cet article, nous considérons le cas où la continuité entre p_n et les enfants de n est perdue mais les enfants du nœud partant gardent une région continue.

La solution proposée dans cette section est basée sur le concept de *clones* qui consiste à remplacer le nœud partant par une représentation virtuelle contenant un sous-ensemble des nœuds qui restent.

3.3.1. *La phase de clonage*

Rappelons que $\mathbf{O}_n = \{o_1, o_2, \ldots, o_k\}$, et que $\hat{a} = o_k$ est l'ensemble de k orphelins du nœud n et que $R^{\ominus}_{o_1} > R^{\ominus}_{o_2} > \ldots > R^{\ominus}_{\hat{a}}$. Cette séquence représente l'ordre d'arrivée (dont $\hat{a}$ est le dernier) des enfants de n.

Dans cet article, nous nous concentrons sur la garantie de continuité entre p_n et les orphelins de n. Ainsi, un chemin entre p_n et $\hat{a}$ satisfaisant la continuité des régions doit être assuré.

Après le départ de n, on doit garantir que les messages adressés à n'importe quel $d_i \in \mathbf{D}_n$ seront correctement délivrés. Rappelons que $\mathbf{C}_{p_n} = \{c_1, \ldots, c_{i-1}, c_i = n, c_{i+1}, \ldots, c_k\}$ est l'ensemble des enfants de p_n ordonnés par leurs adresses relatives, c'est-à-dire $R^{\ominus}_{c_1} < \ldots < R^{\ominus}_{c_{i-1}} < R^{\ominus}_{n} < R^{\ominus}_{c_{i+1}} < \ldots < R^{\ominus}_{c_k}$. Comme $\hat{a} \in \mathbf{C}_n$, l'inégalité $R^{\ominus}_{c_1} < \ldots < R^{\ominus}_{c_{i-1}} < R^{\ominus}_{\hat{a}} < R^{\ominus}_{d_1} < R^{\ominus}_{d_2} < \ldots < R^{\ominus}_{c_{i+1}} < \ldots < R^{\ominus}_{c_k}$ s'avère vraie. Il est donc facile de conclure que tous les messages adressés à $d_j \in \mathbf{D}_{\hat{a}}$ seront acheminés au travers de $\hat{a}$.

Le mécanisme pour garantir la continuité des régions entre p_n et $\hat{a}$ fonctionne de la façon suivante. Le nœud p_n envoie un message `Discover` adressé à $\hat{a}$ (p_n ne voit pas $\hat{a}$). À la réception de ce message, $\hat{a}$ répond en envoyant un message `Path` qui contient la liste $\mathbf{P}(\hat{a})$ des nœuds intermédiaires traversés par le message `Discover` : $\mathbf{P}(\hat{a}) = \{b_1, b_2, \ldots, b_M\}$, où M est le nombre de sauts entre p_n et $\hat{a}$, $b_1 = \hat{a}$, et $b_M = p_n$.

En inspectant $\mathbf{P}(\hat{a})$, le nœud p_n établie un chemin jusqu'au nœud $\hat{a}$ avec des *clones*. Les clones sont une représentation virtuelle de tous les nœuds dans le chemin entre p_n et $\hat{a}$. Les clones se comportent de façon à ce que le réseau opère comme si n était toujours présent. Chaque clone contrôle une fraction de la région de contrôle qui était sous la responsabilité de n.

Tout d'abord, le nœud p_n partage la région $\mathcal{R}_n$ en M parties. Ces régions sont attribuées aux nœuds entre la "source" (p_n) et la "destination" ($\hat{a}$). Soit c_i le clone associé au nœud b_{M-i+1}, $1 \le i \le M$, et $\mathcal{R}_i = \{R^{\ominus}_i, R^{\oplus}_i\}$ la région attribuée à ce clone. La région originelle est divisée comme suit :

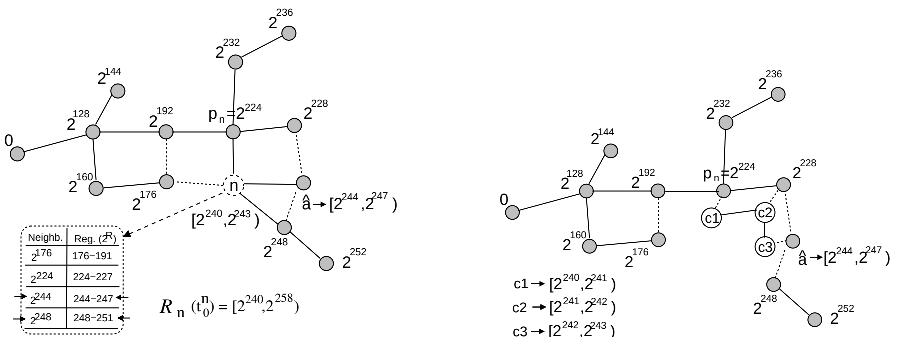
$$\left\{ \begin{array}{ll} R^{\ominus}_i = & R^{\ominus}_n + (i-1)\left\lceil \frac{R^{\oplus}_n - R^{\ominus}_n}{M} \right\rceil, \\ R^{\oplus}_i = & \min\left\{ R^{\oplus}_n, R^{\ominus}_i + \left\lceil \frac{R^{\oplus}_n - R^{\ominus}_n}{M} \right\rceil - 1 \right\}. \end{array} \right.$$

La base de données d'informations de localisation est également partagée entre les clones. Chaque clone sera responsable donc de la gestion de sa propre région de contrôle originelle et de celle qui lui a été attribuée en tant que clone.

La Fig. 4 montre un exemple de l'algorithme d'attribution des régions. Après le départ de n, p_n vérifie qu'il n'est pas un voisin des orphelins de n. Il envoie un message `Discover` à $\hat{a}$ et à a_j dont les adresses relatives sont $E_{\hat{a}} = 2^{244}$ et $E_{a_j} = 2^{248}$. Ces deux nœuds renvoient chacun un message `Path` : $\mathbf{P}_{\hat{a}} = \{2^{244}, 2^{228}, 2^{224}\}$ et $\mathbf{P}_{a_j} = \{2^{248}, 2^{244}, 2^{228}, 2^{224}\}$. Le nœud p_n analyse ces messages et conclut qu'il existe une connectivité entre $\hat{a}$ et a_j respectant la loi de la continuité des régions (a_j peut être atteint au travers de $\hat{a}$). De plus, un chemin entre p_n et $\hat{a}$ avec trois clones peut être établi.

Le nœud p_n divise $\mathcal{R}_n = [2^{240}, 2^{243}]$ en trois parties. Chaque close se verra attribuer une sous région, comme montre la Fig. 4(b).

Le départ du nœud n ne compromet pas l'opération du protocole Tribe et ne provoque aucune perte d'information de localisation. La Fig. 4(b) montre la distribution, en ordre croissante, des régions entre les clones. Il convient d'observer que la route entre p_n et $\hat{a}$ précédemment assurée par n est maintenant représentée par les clones.



(a) (b)

Figure 4. *Déplacement du nœud n dans la deuxième situation classique.*

3.3.2. *Routage avec des clones*

Le routage avec des clones se fait de la même façon que le routage de base. La seule différence consiste à vérifier si le nœud qui achemine le paquet doit l'envoyer vers un clone voisin ou un nœud voisin. Si le nœud est la source d'un chemin cloné (c'est-à-dire le premier clone du chemin), il doit décider s'il fonctionnera en mode nœud ou mode clone. Cette décision est prise à partir de l'adresse destination contenue dans le message de données à acheminer. Si le nœud correspond à la destination du chemin cloné, il éxecute la procédure de routage en mode normal (mode nœud). Par contre, si le nœud est un clone intermédiaire, il doit tout simplement envoyer le message au prochain clone du chemin. Un chemin cloné est équivalent à un tunnel.

3.3.3. *Annulation du clonage*

En plus du clonage, le mécanisme de réattribution de régions a aussi pour but de maintenir une région de contrôle par nœud. Ainsi, à chaque fois qu'un nœud rejoint le réseau dans une position qui avait été précédemment occupée par un autre nœud, le nœud père vérifie si le nouvel arrivant peut remplacer les clones qui remplacent l'ancien occupant. Si oui, le voisin père p_n assume qu'il est convenable de donner au nouveau nœud cette région de contrôle.

4. Travaux liés

La plupart des algorithmes de routage proposés pour les réseaux auto-organisable distribuent l'information de la topologie entre tous les nœuds dans le réseau. Des propositions pour les réseaux ad hoc sont des algorithmes proactifs et sont basés sur des propositions précédemment développées dans le cadre des réseaux filaires traditionnels, comme par exemple le protocole DSDV [PER 94], basé sur Bellman-Ford distribué (DBF). Dans DSDV, chaque nœud maintient une table de routage avec des indicateurs à tous les nœuds dans le réseau. Ceci est acceptable si la population des utilisateurs est petite. Néanmoins, à la mesure que le nombre de nœuds mobiles augmente, la surcharge de signalisation est également accrue. Ainsi, ces algorithmes sont peu résistants au facteur d'échelle.

D'autres propositions sont réactives et constituent la plupart des protocoles de routage actuellement évalués par le groupe de travail MANET [COR 99] de l'IETF. Deux des exemples sont AODV [PER 99, PER 02] et DSR [JOH 02]. Ces protocoles utilisent un mécanisme d'inondation de messages de requête pour la découverte des routes. Ils ne sont donc pas très performants en terme de charge de signalisation dans le cadre des réseaux à large échelle.

Contrairement à DSDV, AODV et DSR, chaque nœud dans Tribe ne stocke que des informations sur lui-même et sur ses voisins immédiats. Aucune procédure d'inondation n'est nécessaire au moment du routage. En plus, les nœuds acheminent les messages vers la destination en envoyant le message au voisin dont la région de contrôle s'approche le plus de la région de contrôle qui contient l'identifiant de la destination.

L'acheminement géographique [FIN 87, KAR 00, LI 00, WOO 01] est une proposition de routage qui peut assurer d'une façon assez performante le passage à l'échelle dans de réseau de grande échelle. Une approche pour la localisation géographique des nœuds est le service de localisation GLS [LI 00]. Le projet Grid [Gri , CHA 02] propose un protocole de routage résistant au facteur d'échelle qui emploie la combinaison de l'acheminement géographique avec GLS [LI 00]. GLS établit une base de données de localisation distribuée où chaque nœud choisit ses serveurs de localisation situés à des distances exponentiellement croissantes.

Autant l'acheminement géographique que GLS utilisent le système de positionnement global (GPS) pour déterminer la localisation géographique d'un nœud. Cette dépendance est impraticable dans beaucoup de réseaux. Tribe est complètement indépendant de n'importe quelle information géographique et la topologie est une représentation virtuelle du réseau physique où les nœuds sont identifiés par leur voisinage dans le réseau physique.

Le projet Terminodes [BLA 01] est également établi autour du concept d'acheminement géographique et propose une distribution de l'information de routage similaire à celle du protocole Tribe. Leur principale différence est que Terminodes utilise l'information géographique pour créer la topologie et pour effectuer le routage. Terminodes propose aussi un positionnement *GPS-free* pour les situations où le GPS n'est pas disponible. Cette proposition utilise des mesures de distance entre les nœuds pour établir un système de coordonnées afin de placer des nœuds dans le réseau. Dans Tribe, la position d'un nœud reflète sa localisation relative dans le réseau et il n'y a aucun besoin d'un système de positionnement géographique ou de prendre des mesures de distances.

Des architectures comme CAN [RAT 01], Chord [STO 03] et Pastry [ROW 01] emploient un concept similaire à la stratégie de routage indirecte proposée dans cet article. Néanmoins, différemment du protocole Tribe, elles sont traitées comme des infrastructures surcouche (*overlay*) et créent une topologie au niveau applicatif.

5. Travaux Futurs

Le protocole Tribe soulève un certain nombre de défis de conception, une partie d'eux sont présentées dans cet article tandis qu'un certain nombre d'autres sont laissées pour les travaux futurs.

Tribe n'a actuellement aucun mécanisme spécifique pour fusionner des réseaux Tribe form'es séparément. Nous avons étudié quelques heuristiques afin de traiter cette problématique, cependant, quelques questions fondamentales concernant la fusion des différents espaces topologiques sont un problème non résolu.

Par rapport la mobilité, cet article présent comment Tribe traite le problème de la discontinuité des régions de contrôle originée par des nœuds que se déplacent. La solution présentée n'est pas très satisfaisante, ce qui nous mène à proposer un autre espace topologique qui préservera la continuité de régions quand le déplacement d'un nœud.

Plusieurs facteurs peuvent être employés pour améliorer l'exécution de la procédure de routage de Tribe. Une proposition est d'employer un critère de partage qui permet de réduire le nombre moyen de sauts nécessaires au routage dans la topologie créée et qui maintient une bonne distribution des nœuds dans le réseau. Néanmoins, cet article présente une première analyse d'exécution du protocole de routage de Tribe. L'évaluation des caractéristiques comme l'hétérogénéité des nœuds et l'autre mécanisme de partage de régions sont des sujets aux travaux futurs.

Actuellement, des améliorations pour la gestion des pannes, sécurité et la robustesse sont étudiées et évaluées. Il est intéressant de dire que le nœud ancre semble être le bon endroit pour traiter la redondance dans Tribe afin d'augmenter la robustesse du protocole.

6. Conclusion

Le nomadisme implique le besoin de mises à jour fréquentes des adresses des utilisateurs ce qui nous mène à penser à dissocier l'identifiant permanent d'un nœud de son adresse dépendante de la topologie. Le protocole Tribe traite ce nouveau besoin et propose une stratégie indirecte résistante au facteur d'échelle pour les réseaux sans fil auto-organisables. L'adressage indirect joue un rôle fondamental en fournissant des solutions pour la mobilité. Tribe utilise une abstraction basée sur des nœuds ancres, où le routage est exécuté en deux phases : la localisation de la destination et la communication elle-même. Dans chaque phase, le système distingue l'identifiant de l'adresse dépendante de la topologie d'un nœud. Ceci permet à Tribe de gérer les départs et les arrivés dynamiques de nœuds.

En utilisant le concept de table de hachage distribuée, Tribe assure le passage à l'échelle et distribue la base de données d'information de localisation entre tous les nœuds du réseau. La topologie est une représentation virtuelle du réseau qui décrit la localisation relative des nœuds selon leur voisinage. En gérant les régions d'un espace d'adressage virtuel, les nœuds de Tribe utilisent, pour le routage, une quantité d'information limitée et un faible coût de maintenance. Pour chaque nœud, Tribe maintient une table de routage avec $O(k)$ lignes (où k est le nombre de voisins d'un nœud) et un coût de communication de routage en $O(1)$. Tribe assure la stabilité du système en gérant la continuité de la topologie lors du départ des nœuds grâce au mécanisme adaptatif de réattribution des régions de contrôle.

Les principales caractéristiques de Tribe sont la simplicité et l'indépendance d'un positionnement géographique. D'autres détails sur la technique de routage indirect sont présentés dans [VIA 03a, VIA 03b].

7. Bibliographie

[BLA 01] BLAZEVIC L., BUTTYAN L., S. CAPKUN S. G., HUBAUX J. P., BOUDEC J. Y. L., « Self-Organization in Mobile Ad-Hoc Networks : the Approach of Terminodes », *IEEE Computer Communications Magazine*, , 2001.

[CHA 02] CHAMBERS B. A., « The Grid Roofnet : a Rooftop Ad Hoc Wireless Network », Master's thesis, Deparment of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology - MIT, juin 2002.

[COR 99] CORSON S., BATSELL S., MACKER J., « Mobile ad hoc networking (MANET) : Routing protocol performance issues and evaluation considerations », *Internet RFC 2501*, , 1999.

[FIN 87] FINN G., « Routing and addressing problems in large metropolitan-scale internetworks », rapport, mars 1987, SI Res. Rep. ISI/RR-87-180, Univ. Southern California, Los Angeles.

[Gri] GRID PROJECT, `http ://www.pdos.lcs.mit.edu/grid/`.

[HUB 01] HUBAUX J. P., GROSS T., BOUDEC J. Y. L., VETTERLI M., « Towards self-organized mobile ad hoc networks : the Terminodes project », *IEEE Communications Magazine*, vol. 39, n° 1, 2001, p. 118–124.

[JOH 02] JOHNSON D. B., MALTZ D. A., HU Y.-C., JETCHEVA J. G., « The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR) », *Internet Draft*, , 2002, `draft-ietf-manet-dsr-07.txt`.

[KAR 00] KARP B., KUNG H. T., « GPSR : Greedy Perimeter Stateless Routing for Wireless Networks », *Proceedings of ACM MOBICOM'00*, août 2000.

[LI 00] LI J., JANNOTTI J., COUTO D. S. J. D., KARGER D. R., MORRIS R., « A Scalable Location Service for Geographic Ad Hoc Routing », *Proceedings of ACM MOBICOM'00*, août 2000.

[PER 94] PERKINS C. E., BHAGWAT P., « Highly dynamic destination sequenced distance-vector routing (DSDV) for mobile computers », *Proceedings of ACM SIGCOMM'94*, octobre 1994.

[PER 99] PERKINS C. E., ROYER E. M., « Ad Hoc on demand distance vector (AODV) routing », *Proceedings of IEEE Wrokshop Mobile Computing systems and Applications (WMCSA'99)*, février 1999.

[PER 02] PERKINS C. E., BELDING-ROYER E. M., DAS S. R., « Ad Hoc On Demand Distance Vector (AODV) Routing », *Internet Draft*, , 2002, `draft-ietf-manet-aodv-12.txt`.

[RAT 01] RATNASAMY S., FRANCIS P., HANDLEY M., KARP R., SHENKER S., « A Scalable Content-Addressable Network », *Proceedings of ACM SIGCOMM'01*, août 2001.

[ROW 01] ROWSTRON A., DRUSCHEL P., « Pastry : Scalable, distributed object location and routing for large-scale peer-to-peer systems », *Proceedings of IFIP/ACM Middleware'01*, novembre 2001.

[STO 03] STOICA I., MORRIS R., LIBEN-NOWELL D., KARGER D. R., KAASHOEK M. F., DABEK F., BALAKRISHNAN H., « Chord : A Scalable Peer-to-peer Lookup Service for Internet Applications », *IEEE/ACM Transactions on Networking*, vol. 11, n° 1, 2003, p. 17–32.

[VIA 03a] VIANA A. C., AMORIM M. D., FDIDA S., REZENDE J. F., « Indirect Routing using Distributed Location Information », *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PERCOM'03)*, Dallas-Fort Worth, Texas, mars 2003.

[VIA 03b] VIANA A. C., AMORIM M. D., FDIDA S., REZENDE J. F., « Routage pair-à-pair dans les réseaux spontanés à large échelle », *5éme Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (Algotel'03)*, Banyuls-sur-mer, France, mai 2003.

[WOO 01] WOO S.-C. M., SINGH S., « Scalable Routing Protocol for Ad Hoc Networks », *Wireless Networks*, vol. 7, n° 5, 2001, p. 513-529.