Université des Sciences et de la Technologie Houari Boumediene

Faculté d'électronique et informatique

Département d'informatique

Thèse de Magister

Le problème d'exclusion mutuelle dans les réseaux mobiles Ad Hoc

Présenté par :
Derhab Abdelouahid

Proposé par :
Dr. Nadjib Badache

Devant le jury composé de :

Madame Drias	Présidente
Monsieur Ahmed Nacer	Membre
Madame Alimazighi	Membre

REMERCIEMENT

Je tiens à remercier plus particulièrement mon directeur de thèse Dr. Nadjib Badache d'abord, pour m'avoir soutenu pendant toutes ces années et m'avoir encouragé à faire de la recherche scientifique, ensuite, pour m'avoir fait confiance en me proposant ce sujet. Je le remercie également pour sa disponibilité, son suivi et ses critiques constructives.

J'adresse également mes sincères remerciements à Madame Hassina Aliane chef du laboratoire des logiciels de base au CERIST pour ses encouragements et pour son aide logistique.

J'associe à ses remerciements toutes les personnes qui m'ont aidé à réaliser ce travail.

SOMMAIRE

Introduction

La communication entre des utilisateurs munis d'ordinateurs interconnectés à travers des liaisons sans fil devient de plus en plus populaire. Ceci est dû au progrès technologique récent dans le domaine des télécommunications qui a permis la manipulation de l'information à travers des unités de calcul portables comme les PDAs (Personal Digital Assistant) et les laptops, en formant un réseau mobile.

Il y'a deux catégories de réseaux mobiles sans fil :

Les réseaux avec infrastructure (*infrastructured*) qui utilisent le modèle de communication cellulaire (les réseaux GSM), ce modèle est composé de deux ensembles d'entités distinctes : un grand nombre d'hôtes mobiles et un nombre relativement faible de stations fixes. L'ensemble des stations fixes et les liaisons filaires entre elles constituent le réseau fixe (statique). Certains sites fixes appelés *stations de base* sont munis d'une interface de communication sans fil pour la communication directe avec des *unités mobiles* localisées dans une zone géographique limitée, appelée *cellule*.
Dans ce modèle, la mobilité des hôtes est représentée par la migration des hôtes mobiles entre les cellules.
Un des problèmes majeurs dans ce type de réseaux est le problème du Handoff, qui essaye de traiter le cas de migration d'un hôte mobile vers une autre cellule.

Les réseaux sans infrastructure (*infrastructureless*) qui sont les réseaux ad hoc, ne requièrent aucune infrastructure fixe. En d'autres termes, un réseau mobile ad hoc est un réseau temporaire formé d'un ensemble d'utilisateurs qui veulent communiquer entre eux sans aucune forme d'administration centralisée. La structure d'un réseau ad hoc est fortement dynamique, qui est la conséquence de la mobilité des hôtes.
Chaque unité mobile dans le réseau joue le rôle d'un routeur et/ou d'un hôte, et peut donc être chargée de transmettre des messages pour les autres unités mobiles.

Un réseau ad hoc possède un certain nombre de caractéristiques qui imposent de nouvelles exigences dans l'implémentation des algorithmes distribués. Les caractéristiques les plus importantes sont : Le changement fréquent de la topologie du réseau, la déconnexion fréquente d'un hôte, le partitionement possible du réseau et des ressources modestes (CPU, espace de stockage, source d'énergie et la bande passante).

Ce travail consiste à étudier le problème d'exclusion mutuelle dans les réseaux Ad Hoc ; il consiste à présenter les algorithmes qui existent et à proposer un algorithme qui résout ce problème et traite le problème du partitionnemnt et la panne des unités mobiles.

Ce document est composé de six chapitres :

Le premier chapitre est une introduction générale aux réseaux mobiles et principalement aux réseaux Ad Hoc.

Le deuxième chapitre est une introduction générale au problème d'exclusion mutuelle.

Le troisième chapitre présente les algorithmes d'exclusion mutuelle dans les réseaux mobiles Ad Hoc.

Le quatrième chapitre présente un nouvel algorithme d'exclusion mutuelle ainsi qu'une preuve qui vérifie les conditions de sûreté et de vivacité.

Le cinquième chapitre est une étude de complexité des algorithmes présentés dans les chapitres 3 et 4.

Le sixième chapitre est une comparaison de performances entre l'algorithme proposé dans le chapitre 4 et les algorithmes présentés dans le chapitre 3 en utilisant le simulateur GloMoSim.

Chapitre 1

Les environnements mobiles

1.1. Introduction

L'évolution rapide de la technologie dans le domaine de la communication sans fil, a permis à des usagers munis d'unités de calcul portables d'accéder à l'information indépendamment des facteurs : temps et lieu. Ces unités, qui communiquent entre elles à travers leurs interfaces sans fil [IMI 94], peuvent être de diverses configurations : avec ou sans disque, des capacités de sauvegarde et de traitement plus ou moins modestes et alimentées par des sources d'énergie autonomes (batteries). L'environnement de calcul résultant est appelé *environnement mobile* (ou nomade). Cet environnement n'astreint plus l'usager à une localisation fixe, mais lui permet une libre mobilité tout en assurant sa connexion avec le réseau [BAD 98].

Les environnements mobiles permettent une grande flexibilité d'emploi. En particulier, ils permettent la mise en réseau des sites dont le câblage serait trop onéreux à réaliser, voire même impossible (en présence d'une composante mobile).

L'environnement mobile offre beaucoup d'avantages par rapport à l'environnement habituel. Cependant de nouveaux problèmes peuvent apparaître (le problème de routage par exemple), causés par les nouvelles caractéristiques du système. Les solutions conçues pour les systèmes distribués avec uniquement des sites statiques, ne peuvent pas donc être utilisées directement dans un environnement mobile. De nouvelles solutions doivent être trouvées pour s'adapter aux limitations qui existent.

1.2. Caractéristiques d'un environnement mobile

1.2.1 La fiabilité des liaisons sans fil

Une bande passante limitée: La bande passante dans les médias de communication sans fil est moindre que celle dans les médias de communication filaire.
On trouve les réseaux locaux sans fil (*Wireless LANs*) qui atteignent un débit de 2 à 11 Mbps, et les réseaux sans fil dans des régions étendues (*Wireless WANs*) qui atteint un débit de 19,2 Kbps, exemple les réseaux CPDP (cellular Digital Data Packet).
Cette bande passante limitée affecte les performances des protocoles distribués et les applications qui ont besoin de transférer une grande quantité d'information.

Un taux d'erreur important : Tandis que les liaisons filaires dans les réseaux statiques offrent un taux d'erreur de transmission (Bit Error Rate) de l'ordre de 10^{-5} sur une ligne téléphonique et 10^{-7} à 10^{-8} sur un coaxial et 10^{-12} sur une fibre optique, les liaisons sans fil offrent un taux d'erreurs de transmission de l'ordre de 10^{-2} à 10^{-6}, en

plus de ça les liaisons sans fil sont très sensibles au sens de propagation des signaux, des *multifading*, et des interférences entre les signaux [KRI 96].

1.2.2. Modes de fonctionnement d'une unité mobile

Dans un système distribué classique, un site est soit connecté, soit totalement déconnecté du réseau. Par contre dans un environnement mobile, il peut y avoir plusieurs formes de liaison d'une unité mobile au reste du réseau, allant d'une déconnexion totale à une connexion faible lorsque la liaison de communication est de faible débit [PIT 93]. Pour économiser de l'énergie, une unité mobile peut fonctionner en *mode veille* (*doze mode*) ; dans ce cas la vitesse du processeur est réduite et aucun calcul n'est effectué. L'unité mobile est alors en attente de réception d'un message pour reprendre son fonctionnement normal. Alors que dans un environnement distribué classique, les déconnexions sont imprévisibles, dans un environnement mobile, la plupart des déconnexions peuvent être détectées et des protocoles peuvent être construits pour les prendre en charge.

Une unité mobile peut exécuter un protocole de déconnexion avant qu'elle ne se détache physiquement du réseau. Ce protocole permet d'assurer que l'unité a téléchargé suffisamment de données et les informations d'état nécessaires, lui permettant de continuer à fonctionner indépendamment des autres sites, et à des sites éventuels d'être informés de sa déconnexion. Une unité mobile peut aussi basculer à l'état « *partiellement connecté* » par l'exécution d'un protocole spécifique.
Durant ce mode de fonctionnement, toutes les communications avec le réseau doivent être limitées. Finalement, au cours de son fonctionnement et dans n'importe quel mode, une unité mobile peut sortir des limites de sa cellule courante pour entrer dans une nouvelle cellule. Un protocole dit protocole Handoff est exécuté et permet le transfert des informations d'état relatives au calcul mobile en cours à la station de base de la nouvelle cellule.

1.3. Les réseaux cellulaires

Le modèle de système intégrant des sites mobiles est composé de deux ensembles d'entités distinctes : les sites fixes d'un réseau de communication filaire classique (*wired network*) et les sites mobiles (*wireless network*) [IMI 94].
Certains sites fixes appelés stations support mobiles (Mobile Support Stations) ou *station de base (SB)* sont munis d'une interface de communication sans fil pour la communication directe avec des *unités mobiles* (Mobile Host) *UM* (voir figure 1.1), localisées dans une zone géographique limitée, appelée *cellule*.

A chaque station de base correspond une cellule à partir de laquelle des unités mobiles peuvent émettre et recevoir des messages. Alors que les sites fixes sont interconnectés entre eux à travers un réseau de communication filaire, généralement fiable et d'un débit élevé, les liaisons sans fil ont une bande passante limitée qui réduit sévèrement le volume des informations échangées [DUC 92].

Une unité mobile ne peut être à un instant donné, directement connectée qu'à une seule station de base. Elle peut communiquer avec les autres sites à travers la station de base à laquelle elle est directement rattachée.
Pour envoyer un message d'une unité mobile UM_1 à une autre unité mobile UM_2, UM1 envoie le message à sa station de base SB_1 à travers le réseau sans fil qui le transmet à la station de base de UM_2 appelée SB_2 qui à son tour le transmet à UM_2.

Pour qu'une unité mobile UM quitte sa cellule et entre dans une autre cellule, elle a besoin de fermer sa connexion courante par l'envoi d'un message *leave(r)* à sa station de base SB où r représente le numéro de séquence du dernier message reçu de SB, ensuite il établit une nouvelle connexion en envoyant un message *join(MH-id, previous MSS-id)* à sa nouvelle station de base.

Quand une unité mobile change de cellule, les stations de base des deux cellules exécutent une procédure appelée *handoff* [TRI 98].
Chaque station de base maintient une liste des identificateurs *Active_MH_List* et une structure de données des unités mobiles connectées à elle. Quand une unité mobile entre dans une nouvelle cellule, elle informe sa station de base précédente de l'identificateur de sa nouvelle station de base ou vice versa.

Une unité mobile peut se déconnecter de sa station de base sans quitter sa cellule par l'envoi d'un message *disconnect(r)*, quand la station de base reçoit le message *disconnect* d'une unité mobile UM, elle marque son état à « *déconnecté* » et mettra à jour une liste des unités mobiles déconnectées appelées *Disconnect_MH_List*. Si cette unité
mobile se reconnecte à une station de base, elle envoie un message *reconnect(MH-id, previous MSS-id)* à cette station de base. La nouvelle station de base informe la station de base précédente de la reconnexion de UM, ce qui permet à la station de base précédente d'exécuter sa propre procédure Handoff.
La différence entre quitter une cellule et déconnecter est que l'unité mobile qui quitte une cellule peut apparaître dans une autre cellule, mais quand elle se déconnecte, il n'y a pas une garantie que cette unité mobile va se reconnecter.
La gestion de localisation des unités mobile se fait en se basant sur deux niveaux de structure de données, constituées de : Home Location Register (HLR) et Visitor Location Register (VLR).

Chaque unité mobile est associée à un Home, qui garde trace de sa localisation courante. Plusieurs stations de base sont groupées et associées au VLR qui maintient les informations de localisation des unités mobiles dans les cellules servies par ces stations de base.

Durant l'exécution de la procédure handoff d'une unité mobile UM, la nouvelle station de base envoie une requête de localisation au VLR. Si UM se déplace dans une région gérée par le même VLR, il va seulement mettre à jour la localisation de cette UM, sinon VLR envoie un message de registre de localisation au HLR de UM pour

mettre à jour sa localisation, ensuite VLR envoie un message de registre de suppression à l'ancien VLR.

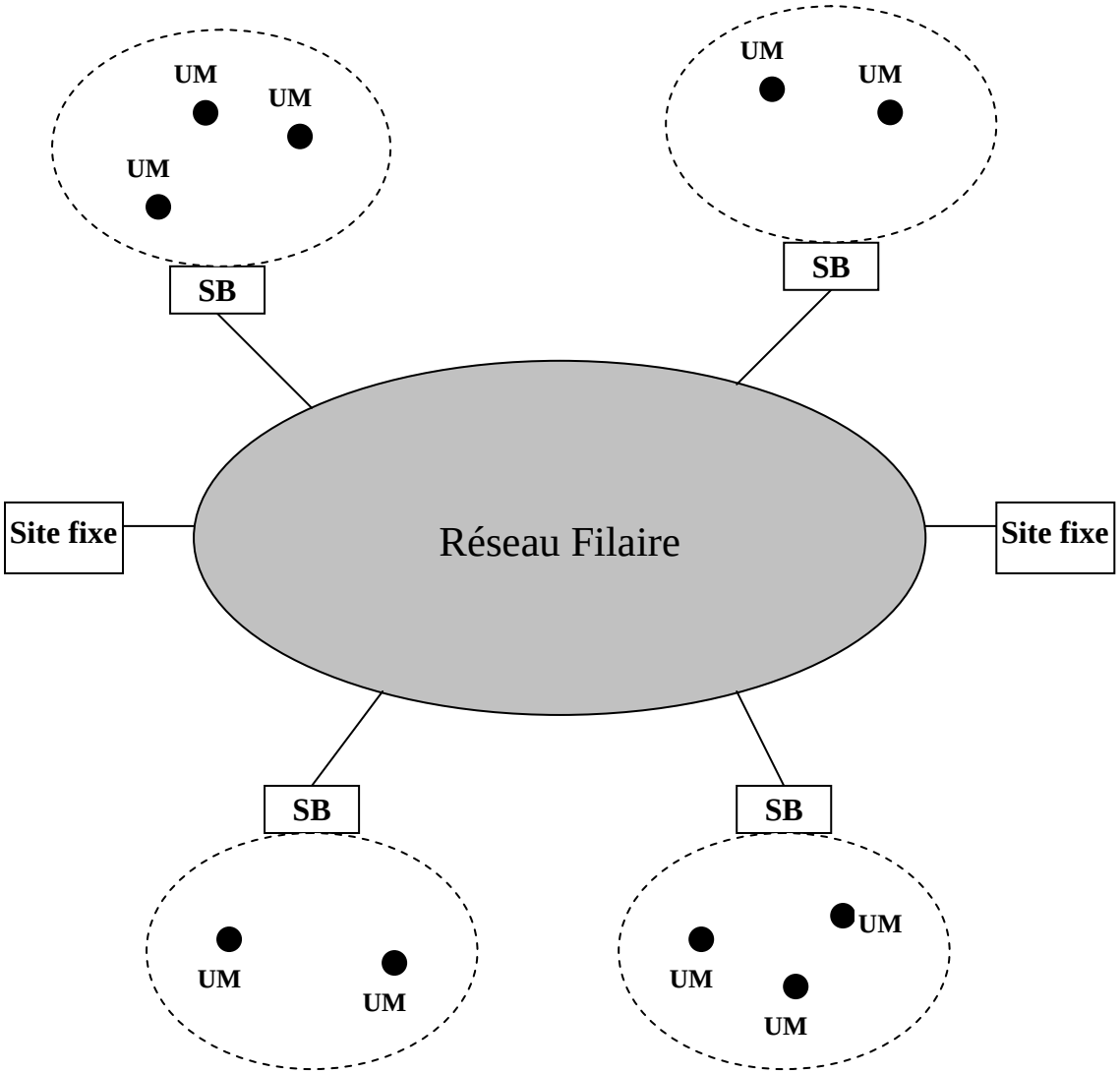


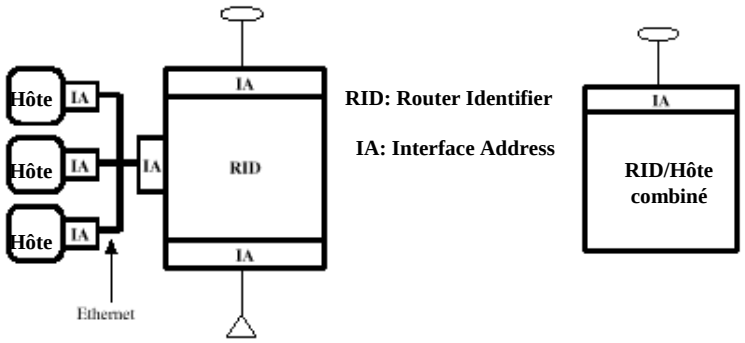
Figure 1.1 Le modèle des réseaux mobiles cellulaire

1.4. Les réseaux Ad Hoc

1.4.1. Introduction aux réseaux Ad Hoc

Un réseau ad hoc est une collection d'unités mobiles munies d'interfaces de communication sans fil formant un réseau temporaire sans recourir à aucune infrastructure fixe ou une administration centralisée [LAR 98].
Dans de tels environnements, les unités mobiles se comportent comme des hôtes et/ou des routeurs.

Un réseau mobile Ad Hoc ou Mobile Ad hoc NETwork (MANET) [MACa 98,MACb 98] est constitué de plates-formes mobiles, où chaque plate-forme est composée soit d'un routeur et de plusieurs hôtes, lié à des interfaces de communication filaire et sans fil (Figure 1.2.a), soit d'un élément qui joue le rôle d'un routeur et d'un hôte (exemple un laptop) lié à des interfaces de communication sans fil (Figure 1.2.b), l'entité routeur exécute un protocole de routage et l'hôte désigne simplement une adresse IP dans le sens traditionnel.



(a) **(b)**

Figure 1.2

Les nœuds des MANET sont équipés d'émetteurs et de récepteurs sans fil utilisant des antennes qui peuvent être omnidirectionnelles (broadcast), fortement directionnelles (point-à-point), ou une combinaison de ces deux types. A un instant donné, en fonction de la position des nœuds, de la configuration de leur émetteur-récepteur, des niveaux de puissance de transmission et d'interférence entre les canaux, il y a une connectivité sans fil qui existe entre les nœuds, sous forme de graphe multi-saut. Cette topologie ad hoc peut changer avec le temps en fonction du mouvement des nœuds ou de l'ajustement de leurs paramètres d'émission-réception.

Un MANET peut être isolé, mais il peut aussi avoir des passerelles ou des interfaces qui le relient à un réseau fixe. Dans son fonctionnement futur, on va le voir comme un réseau "final" rattaché à un réseau d'interconnexion. Un réseau final gère le trafic créé par ou destiné à des nœuds internes, mais ne permet pas à un trafic extérieur de transiter par lui.

1.4.2. Modèle d'un système de réseau Ad Hoc

Un réseau Ad Hoc peut être modélisé par un graphe non orienté $G_t=(V_t, E_t)$, où V_t représente l'ensemble des nœuds (les hôtes mobiles) et E_t représente l'ensemble des liens qui existent entre les nœuds (Figure 1.3).

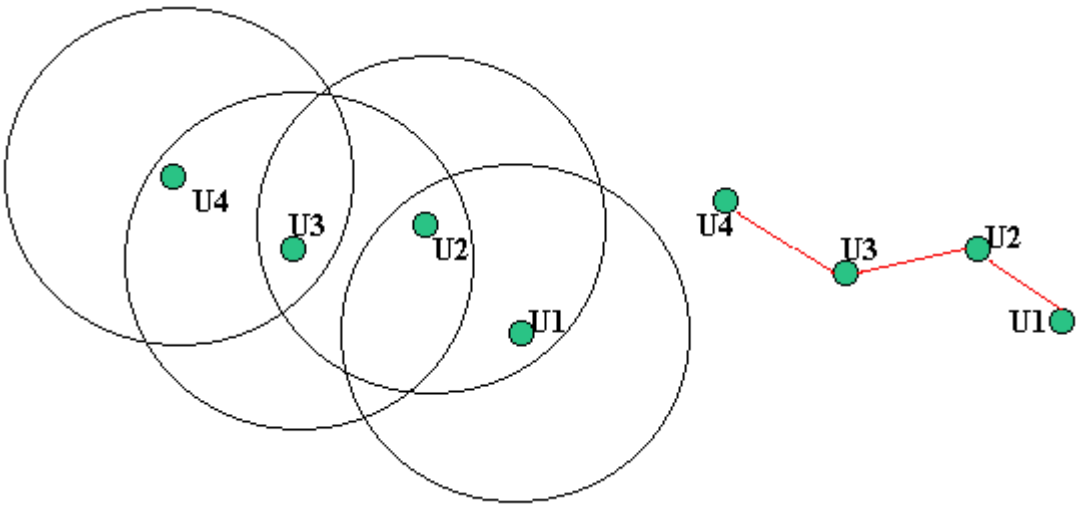


Figure 1.3

Si e=(u,v) $\in E_t$, cela veut dire que les nœuds u et v sont en mesure de se communiquer directement à l'instant t.
La mobilité des nœuds fait que la topologie du réseau peut changer fréquemment et à tout moment, ce qui entraîne de fréquentes déconnexions (Figure 1.4).

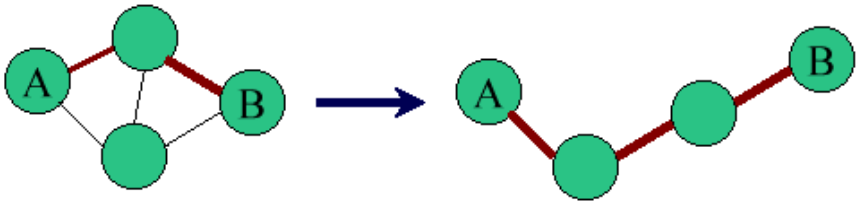


Figure 1.4

1.4.3. Les caractéristiques des réseaux Ad Hoc

Topologies dynamiques : Les unités mobiles se déplacent arbitrairement, ce qui fait que la topologie du réseau peut changer fréquemment et d'une manière rapide.

Une bande passante limitée : Une des caractéristiques importantes des réseaux basés sur la communication sans fil est l'utilisation d'un médium de communication partagé. Ce partage fait que la bande passante réservée à un hôte soit modeste.
Un des effets de ces capacités de liaison relativement faible est que la congestion sera généralement la norme plus que l'exception.

Comme le réseau mobile est une simple extension d'un réseau filaire, les utilisateurs du réseau Ad Hoc demanderont les même services. Cette demande ne cessera de croître avec l'augmentation des applications multimédia.

Des contraintes d'énergie : Les hôtes mobiles sont alimentés par des sources d'énergie autonomes comme les batteries ou les autres sources consommables.

Les composantes physiques : Les différentes unités mobiles du réseau Ad Hoc (palmtop, PDA, notebook) sont menues de CPU de vitesse limitée et de capacité de stockage modeste.

Sécurité physique limitée : Les réseaux mobiles sans fil sont généralement les plus sensibles aux menaces physiques et aux attaques que les réseaux filaires. Cela est du aux contraintes et limitations physiques qui font que le contrôle des données transférées doit être minimisé.
Les techniques existantes pour la sécurité des liaisons sont souvent appliquées aux réseaux sans fil pour réduire les risques d'attaques, mais plusieurs approches ont été mises pour sécuriser la communication dans les réseaux Ad Hoc [ZHO 99].

L'absence d'infrastructure : Un réseau ad hoc est caractérisé par l'absence d'infrastructure préexistant ou d'une administration centralisée.

1.4.4. Les applications des réseaux mobiles ad hoc

Les réseaux ad hoc sont utilisés dans toute application où le déploiement d'une infrastructure réseau filaire est trop contraignant, soit parce que difficile à mettre en place, soit parce que la durée d'installation du réseau ne justifie pas de câblage à demeure, parmi ces applications on trouve :

Application militaire : Les équipements militaires (troupes, véhicules) peuvent tirer avantages des réseaux Ad Hoc qui leur permettent de se déplacer et de communiquer entre eux dans les champs de bataille en maintenant les informations entre les soldats et les chefs des troupes [PER 00] .

Opérations de secours : Les opérations de secours se passent dans des régions désastreuses (incendie, inondation, tremblement de terre) où une infrastructure de réseau filaire ou station de base n'existe pas, mais une communication entre les équipes de secours est nécessaire [PER 00]. Dans ce cas une installation d'un réseau Ad Hoc peut résoudre ce problème.

Les conférences : Le réseau Ad Hoc est faisable pour un groupe de gens qui veulent créer un réseau temporaire, par exemple un réseau Ad Hoc qui relie les différentes unités portables (palmtop, PDA, notebook) peut être utilisé pour propager et partager les informations parmi les participants dans une conférence [FRO 00].

Utilisation privée : Les réseaux Ad Hoc peuvent être utilisés dans les ‘’*Home Networks*’’ ou les différents équipements échangent des informations (sons, vidéo, alarme), une des applications qui est dans ce contexte est un réseau de robots dans une maison qui font plusieurs travaux (nettoyage, assurance de la sécurité) [PER 00, FRO 00] .

Contrôle d’environnement : Des petits véhicules équipés de caméras et des détecteurs de son peuvent être utilisés dans une région déterminée afin de collecter un ensemble d’information et de l’envoyer à travers un réseau Ad Hoc à une station qui traite ces informations, par exemple on peut avec cette méthode prévoir la pollution de l’eau [ROY 99] [FRO 00] .

1.4.5. Problème de routage dans les réseaux Ad Hoc

Comme nous l’avons déjà vu, l'architecture d'un réseau mobile ad hoc est caractérisée par une absence d'infrastructure fixe préexistante, à l'inverse des réseaux de télécommunication classiques. Un réseau ad hoc doit s'organiser automatiquement de façon à être déployable rapidement et pouvoir s'adapter aux conditions de propagation, au trafic et aux différents mouvements pouvant intervenir au sein des unités mobiles.

Dans le but d'assurer la connectivité du réseau, malgré l'absence d'infrastructure fixe et la mobilité des stations, chaque nœud est susceptible d'être mis à contribution pour participer au routage et pour retransmettre les paquets d'un nœud qui n'est pas en mesure d'atteindre sa destination; tout nœud joue ainsi le rôle de station et de routeur.

Chaque nœud participe donc à un protocole de routage qui lui permet de découvrir les chemins existants, afin d'atteindre les autres nœuds du réseau. Le fait que la taille d'un réseau ad hoc peut être énorme, souligne que la gestion de routage de l'environnement doit être complètement différente des approches utilisées dans le routage classique. Le problème qui se pose dans le contexte des réseaux ad hoc est l'adaptation de la méthode d'acheminement utilisée avec le grand nombre d'unités existant dans un environnement caractérisé par de modestes capacités de calcul et de sauvegarde.

Plusieurs protocoles de routage ont été développés pour s’adapter à l’environnement mobile, on trouve les *protocoles pro-actifs*[PER 94, MUR 96, CHE 98, PEI 00, CHI 97, IWA 99] et les *protocoles réactifs*[JOH 96, PER 99, JIA 99, PAR 97, TOH 96, DUB 97]. Les protocoles pro-actifs établissent les routes à l'avance en se basant sur l'échange périodique des tables de routage. Cette approche est coûteuse en termes de ressources. Les protocoles réactifs cherchent les routes à la demande. L’inconvénient de cette approche est que le délai d’obtention d’une route peut être élevé. Dans [BAD 02], il a été montré que les protocoles réactifs sont plus adaptés aux caractéristiques des réseaux Ad Hoc que les protocoles proactifs.

1.5. Conclusion

Ce chapitre a été axé sur le concept des environnements mobiles et l'utilisation de la technologie de communication sans fil. L'évolution rapide qu'a connu la technologie sans fil récemment, a permis l'apparition de nouveaux systèmes de communication qui offrent plus d'avantages par rapport aux systèmes classiques. Les nouveaux systèmes n'astreignent plus l'usager à une localisation fixe, mais lui permettent une libre mobilité.

Nous avons présenté aussi le concept de réseau ad hoc qui est composé de calculateurs mobiles caractérisés par des ressources modestes (vitesse CPU, bande passante, source d'énergie, espace de stockage). Dans la pratique, les réseaux ad hoc connaissent aujourd'hui plusieurs applications telles que les applications militaires et les applications de secours et de façon générale, toutes les applications caractérisées par une absence d'infrastructure préexistante.

Le réseau Ad Hoc connaît une évolution continue pour intégrer de nouvelles fonctionnalités comme l'adaptation de l'adressage IP à la mobilité (Mobile IP), la sécurisation de la communication et l'intégration des extensions QOS [HAR 01].

Chapitre 2

Le problème d'exclusion mutuelle dans les réseaux Ad Hoc

2.1. Introduction

L'implémentation des algorithmes et des protocoles distribués suppose l'existence d'une architecture du réseau comprenant uniquement des sites fixes.
En absence de pannes de sites ou des liaisons de communication, la topologie du réseau reste statique, les algorithmes distribués supposent un modèle comprenant un ensemble de processeurs qui s'exécutent dans des sites fixes [BAD 94].

Les solutions pour les problèmes de synchronisation et de communication dans les systèmes distribués sont basées sur ce modèle de base qui peut être représenté par un graphe non orienté où les sommets représentent les unités de calcul, et les arêtes représentent les liaisons physiques entre ces unités. C'est ainsi que les algorithmes distribués, développés jusque-là, souvent basés sur une topologie logique de type arbre, graphe complet ou anneau, où chaque lien logique correspond à un chemin dans le réseau physique sous-jacent.

Dans un environnement mobile où les sites sont capables de se déplacer, la structure des liaisons physiques change souvent. L'utilisation de structure logique comme plate-forme de base pour la construction d'algorithmes distribués dans un environnement mobile devient alors trop lourde [BAD 98]

2.2. Le problème d'exclusion mutuelle

Le problème d'exclusion mutuelle est un problème de gestion de conflits d'accès à des ressources.
Les processus qui constituent les systèmes distribués peuvent demander la même ressource en même temps. Si la ressource a besoin d'un accès mutuellement exclusif, donc quelques régulations sont exigées pour y accéder, ceci est le problème d'exclusion mutuelle distribuée.

Plusieurs travaux sur le problème d'exclusion mutuelle traite le cas d'une seule ressource dans le système distribué, exemple : le contrôle d'accès à une base de données distribuée. Si le système contient K ressources, le problème deviendra comment attribuer ces K ressources aux processus, ce problème est appelé le problème de *K-exclusion mutuelle distribuée*.

2.3. Classification des algorithmes d'exclusion mutuelle

Les algorithmes distribués d'exclusion mutuelle qui existent dans la littérature sont divisés en deux catégories : ceux qui sont *basés sur les permissions* et ceux qui sont *basés sur les jetons*.

Dans la première catégorie, un processus qui veut accéder à la section critique doit demander et obtenir l'accès de tous ou d'un sous-ensemble de processus du système.

Dans la deuxième catégorie, le processus qui possède le jeton peut accéder à la section critique. Cette catégorie se décompose en deux familles d'algorithmes :

a) Algorithmes demandant un jeton : Un processus qui veut accéder à la section critique, diffuse un message de requête à tous les processus comme dans [SUZ 85]ou il envoie ce message directement au nœud qui possède le jeton comme dans [RAY 89] . Le processus qui possède le jeton insère la requête dans la file des requêtes, quand ce dernier quitte la section critique, il envoie le jeton au premier processus dans la file. Quand la file est vide, le processus qui possède le jeton, garde ce jeton et attend les requêtes qui viennent des autres processus.

b) Algorithmes basés sur un anneau logique : Un jeton circule continuellement dans l'anneau logique, un processus qui reçoit le jeton de son prédécesseur accède à la section critique s'il a besoin d'y accéder.

Les algorithmes distribués d'exclusion mutuelle basés sur les jetons génèrent moins de messages que ceux basés sur les permissions.

Les algorithmes d'exclusion mutuelle doivent garantir les conditions suivantes :
a) L'exclusion mutuelle : Au plus, il n'y a qu'un un seul processus dans sa section critique en un temps donné.
b) Déroulement (pas d'interblocage) : Aucun processus en dehors de sa section critique ne doit bloquer les autres processus.
c) Attente limitée (pas de famine) : Chaque processus qui demande d'accéder à la section critique, va obtenir l'accès en un temps limité.

2.4. L'exclusion mutuelle dans les réseaux statiques

Lamport [LAM 78] a proposé le premier algorithme d'exclusion mutuelle dans les systèmes répartis, cet algorithme est basé sur les permissions, où chaque processus a besoin d'obtenir la permission de tous les autres processus du système pour entrer dans sa section critique, ce qui nécessite un nombre de messages égale à $3\times(n-1)$, un autre algorithme basé sur les permissions proposé par Ricart et Agrawala [RIC 81] a réduit ce nombre à $2\times(n-1)$ messages.

Maekawa [MAE 85] a proposé un schéma dans lequel un processus n'a pas besoin d'obtenir la permission de tous les autres processus du système pour entrer dans sa section critique, ce schéma nécessite un nombre de messages égale à $3\times(\sqrt{n}-1)$ par entrée dans la section critique.

Dans la catégorie des algorithmes basés sur les jetons, Suzuki et Kassami [SUZ 85] ont proposé un algorithme qui requiert au plus n messages par entrée dans la section critique, ce nombre est réduit à $O(\log n)$ dans l'algorithme de Raymond [RAY 89] qui utilise une structure logique basée sur une arborescence enracinée.

2.5. L'exclusion mutuelle dans les réseaux cellulaires

Badrinath [BAD 94] présente un algorithme d'exclusion mutuelle qui s'adapte à un environnement composé de stations de base et des unités mobiles.
L'idée de cet algorithme est de faire un découplage entre la mobilité des sites et la construction des algorithmes pour des environnements mobiles. Pour cela, les algorithmes distribués sont structurés de manière à ce que la majeure partie des communications et du calcul induit par l'algorithme soit prise en charge par le réseau statique. Il suggère que la structure de données qui englobe l'état de l'algorithme réside dans les sites fixes, alors que le rôle des unités mobiles se résume à initier l'exécution de l'algorithme et à recevoir le résultat de l'exécution.
Une autre solution est présentée par Singhal[SIN 97] où l'exécution de l'algorithme se fait seulement entre les nœuds demandants l'accès à la section critique.

2.6. Le problème d'exclusion mutuelle dans les réseaux mobiles Ad Hoc

Le problème de l'exclusion mutuelle a été résolu dans les réseaux cellulaires en adoptant la transformation deux tiers de Badrinath[BAD 94] qui consiste à attribuer aux stations de base la plus grande partie de calcul et de communication durant l'exécution de l'algorithme.

Mais le problème de l'exclusion mutuelle dans les réseaux Ad Hoc est plus compliqué parce que ces derniers n'utilisent aucune infrastructure fixe, d'où la nécessité de concevoir des algorithmes qui s'adapte à ce nouvel environnement caractérisé par ses fréquentes déconnexions, changement de topologie et de partitionement, et qui prend en compte les caractéristiques physiques des unités mobiles (vitesse CPU, bande passante limité, espace de stockage, source d'énergie limitée). Ils doivent être aussi efficaces, performants et tolérants aux pannes.

2.7. Les applications d'exclusion mutuelle dans les réseaux Ad Hoc

On peut voir l'utilisation de l'exclusion mutuelle dans plusieurs applications :

Synchronisation dans une conférence : Cette application consiste à résoudre le problème d'attribution de la parole dans une conférence, exemple d'une application de téléconférence entre plusieurs conférenciers, où le droit de parler est une ressource critique partagée entre les membres de cette conférence [HOU 01].

Réplication de données : Pour des raisons de performance et de fiabilité, un même document peut avoir des copies multiples. Dans un tel contexte, un des problèmes majeurs est celui de la cohérence et du contrôle d'accès aux informations, il est nécessaire d'assurer la cohérence mutuelle des copies de documents répliqués sur plusieurs unités mobiles et de synchroniser l'ensemble des processus afin de garantir

qu'à tout moment, au plus un processus est en cours d'utilisation de la ressource critique, ce problème est appelé le problème de k-exclusion mutuelle [KAR 99].

Robotique: Considérant un système où un robot entre dans une salle, et bloque l'accès à cette salle, les autres robots ne peuvent entrer à moins que ce robot sorte et donne le droit d'entrer à un de ces robots en attente, ce cas est une forme physique de l'exclusion mutuelle [DEF 01].

Aéronautique : Considérant un aéroport dépourvu de tour de contrôle, l'ensemble des avions constitue un réseau Ad Hoc, et le problème consiste à allouer les pistes (ressources critiques) dans ce type de réseau, chaque piste ne peut être utilisée que par un avion à la fois.

2.8. Notions générales

Nous allons expliquer quelques notions utilisées dans les chapitres suivants :

DAG (Direct Acyclic Graph) : On dit qu'un graphe orienté est un DAG pour un sommet j, si à partir de n'importe quel autre sommet i≠j, il existe un chemin vers j.
La Figure 2.1 montre un graphe non orienté, un nœud destinataire (*D*) veut construire un DAG sur ce graphe, il assigne à sa taille la valeur $E_D=(0,0,D)$ et diffuse un message *Update*(E_D), un nœud i qui reçoit un message *Update*(α_j, β_j, *j)* de son voisin j, augmente sa taille pour être $E_i=(\alpha_j, \beta_j +1, i)$ et diffuse sa nouvelle taille à ses voisins.
Ce mécanisme permet de construire des liens orientés (liens sortants) des nœuds qui ont une taille plus grande vers les nœuds qui ont une taille plus petite.
Une taille $E_i=(\alpha_i, \beta_i, i)$ est inférieure à une taille $E_j=(\alpha_j, \beta_j, j)$ si $\beta_i < \beta_j \vee$ i <j, ceci permet de construire un ordre total sur les tailles des nœuds du graphe.
Par cette définition le nœud destinataire est le nœud qui a la plus petite taille, et qui n'a pas des liens sortants. Les tailles des nœuds restent inchangées jusqu'à ce que un nœud perde son dernier lien sortant ou si le jeton circule, dans ce cas une procédure d'inversement de liens est invoquée.

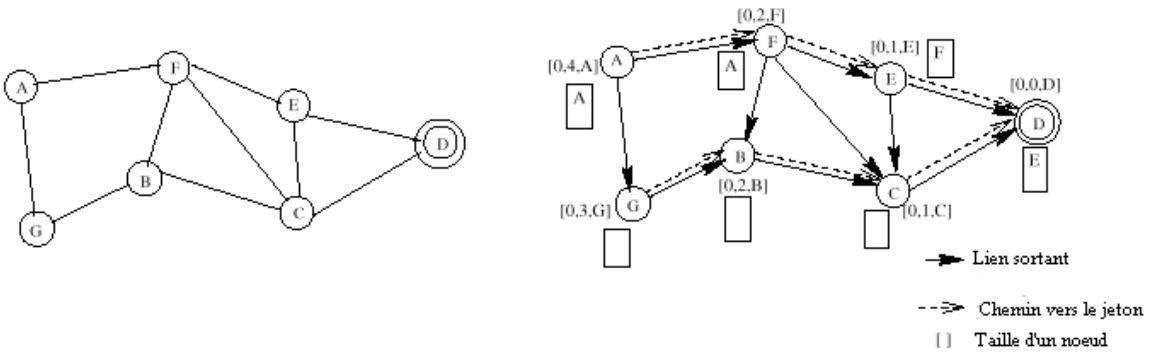


Figure 2.1

Inversement de liens : Une technique utilisée par un nœud qui perd son dernier lien sortant ou quand le jeton circule entre les nœuds.
Le but de cette technique est de garantir toujours l'existence d'un chemin vers le nœud qui possède le jeton.
La Figure 2.2 montre l'effet d'une défaillance d'un lien sur la taille des nœuds et l'orientation des liens.

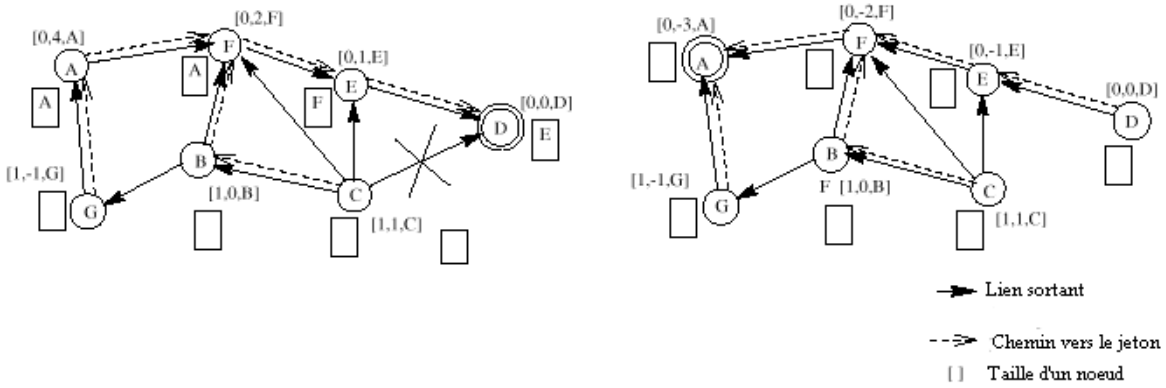


Figure 2.2

Procédure de recherche : Un nœud *i* qui veut localiser un autre nœud *j*, envoie un message de type *RREQ* sur tous ses liens entrants (Figure 2.3), un nœud *k* qui reçoit ce message vérifie si c'est lui le nœud recherché, dans ce cas, il envoie un message de réponse *RREP* vers i, sinon il envoie le message *RREQ* sur tous ses liens entrants.
Cette technique va permettre de trouver plusieurs chemins vers j, mais le premier message *RREQ* reçu par le nœud j va invoquer l'envoi d'un message de réponse (Figure 2.4), les autres messages seront ignorés.

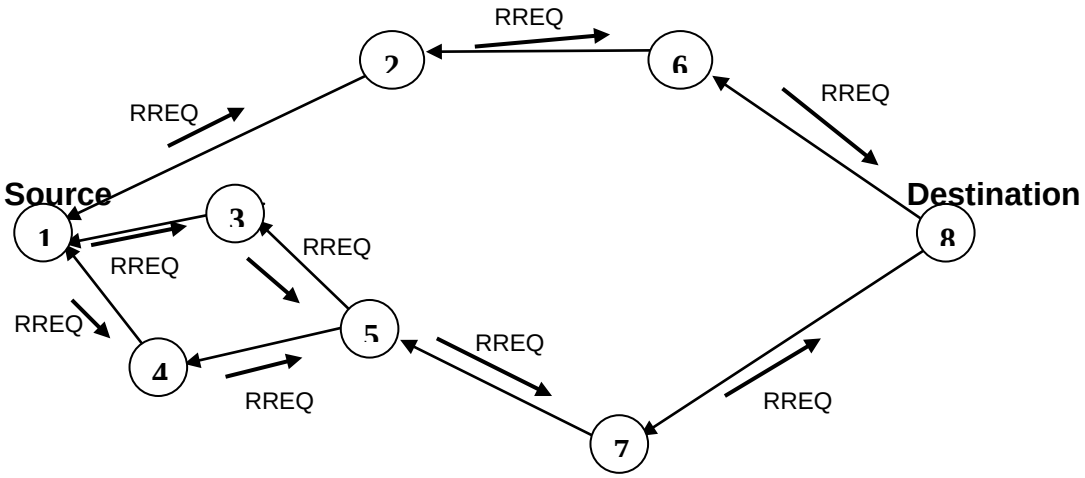


Figure 2.3

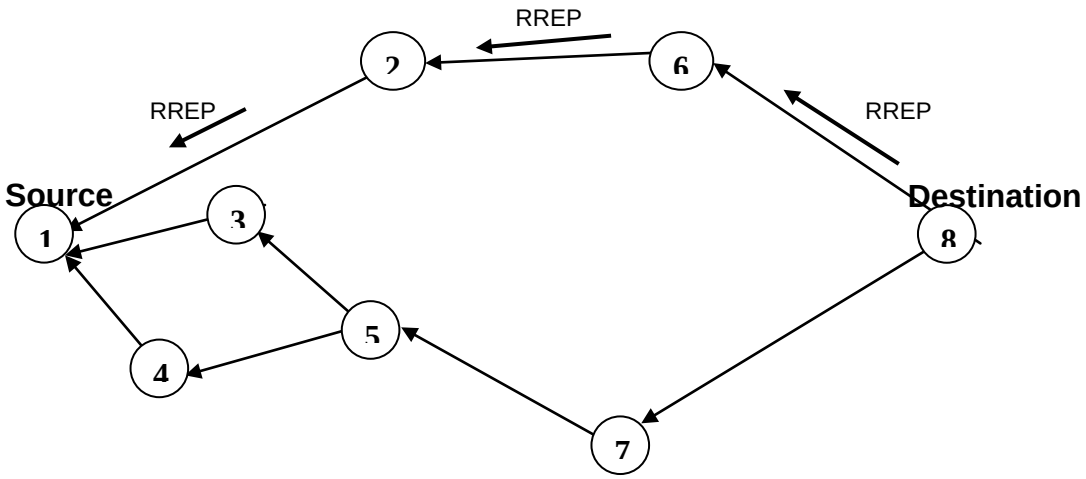


Figure 3.4

2.9. Propriétés des algorithmes d'exclusion mutuelle dans les réseaux Ad Hoc

Le type d'algorithme : Comme Les algorithmes basés sur les permissions consomment plus de messages et ne permettent à aucune unité mobile de se déconnecter pendant leur exécution, il s'avère que les algorithmes basés sur les jetons sont les mieux adaptés aux environnements mobiles Ad Hoc.

Les nœuds exécutant l'algorithme : L'exécution de l'algorithme doit se faire seulement entre les unités mobiles qui demandent d'accéder à leur section critique.

Le nombre d'unités mobiles : Comme le nombre d'unités mobiles dans un réseau Ad Hoc change continuellement, l'algorithme d'exclusion mutuelle ne doit pas supposer que ce nombre est constant.

Consommation d'énergie : Comme les sources d'énergie dans les unités mobiles sont modestes et leur durée de vie est limitée, l'algorithme d'exclusion mutuelle doit minimiser sa consommation d'énergie et cela en réduisant le nombre de messages qu'il génère.

Temps d'attente : Le temps d'attente d'un processus pour entrer dans la section critique doit être optimal.

2.10. Conclusion

Dans ce chapitre, nous avons présenté le problème d'exclusion mutuelle dans les réseaux Ad Hoc et les principales applications qu'on peut trouver.

La mobilité des hôtes et leurs caractéristiques ont induit de nouveaux problèmes qui n'ont pas été présents dans les systèmes répartis composés d'hôtes statiques, parmi ces problèmes on trouve :

1. Pour délivrer un message, l'hôte destinataire doit être d'abord localisé.
2. Les hôtes mobiles sont caractérisés par leurs ressources modestes (source d'énergie, vitesse de processeur, espace de stockage).
3. La déconnexion fréquente des sites mobiles du réseau.

Ces problèmes doivent être pris en compte lors de la conception d'algorithmes distribués dans un environnement mobile.

Dans les réseaux cellulaires, nous avons présenté une méthode de restructuration des algorithmes d'exclusion mutuelle conçus pour des réseaux statique afin de les adapter aux environnements mobiles, composés de stations de base et d'unités mobiles. Cette méthode a montré son efficacité, car les communications et les structures de données nécessaires à l'exécution de l'algorithme sont localisées au niveau de la partie statique du réseau, ce qui minimise la consommation de ressources au niveau des unités mobiles.

Dans les réseaux Ad Hoc, l'absence d'une infrastructure fixe va rendre la conception d'un algorithme d'exclusion mutuelle un problème compliqué, car il doit tenir compte du changement fréquent de la topologie et du partitionement du réseau, et comme les sources d'énergie sont limitées l'algorithme doit générer un nombre de messages optimal.

Chapitre 3

Les algorithmes d'exclusion mutuelle dans les réseaux mobile Ad Hoc

3.1. Introduction

Comme nous l'avons déjà vu, un réseau ad hoc est un ensemble de nœuds mobiles qui sont dynamiquement et arbitrairement éparpillés tels que l'interconnexion entre les nœuds peut changer à tout moment.

Les unités mobiles dans les réseaux Ad Hoc sont caractérisées par leurs fréquentes déconnexions et leurs sources d'énergie limitées, elles échangent des messages dans un environnement où les liens de communication sont non fiables, ces limitations vont imposer d'autres problèmes lors de la conception des algorithmes d'exclusion mutuelle, comme le traitement du partitionement du réseau, les déconnexions, la perte des messages, la consommation d'énergie…etc.

A travers ce chapitre, nous allons présenter les algorithmes d'exclusion mutuelle conçus pour un environnement Ad Hoc qui existe dans la littérature.

3.2. L' algorithme Reverse Link (RL)

Cet algorithme est basé sur les principes de l'algorithme de Raymond[RAY89]. Chaque nœud dans le réseau, hormis le nœud qui a le jeton (*nœud privilégié*) a un lien sortant dirigé vers le nœud privilégié. Ce schéma permet de construire un arbre enraciné (Direct Acyclic Graph) pour le nœud privilégié.

Considérons la première architecture [WAL97] où l'algorithme d'exclusion mutuelle est implémenté au-dessus du protocole de routage (Figure 3.1). Dans ce cas l'algorithme ne tient pas compte de la mobilité, cette tâche est assurée par le protocole de routage au niveau de la couche réseau.

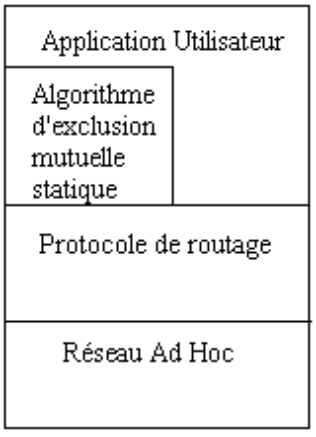


Figure 3.1

Un nœud qui veut entrer en section critique, envoie une requête sur les liens sortants jusqu'à ce qu'elle arrive au nœud privilégié.
A la réception d'une requête, chaque nœud va mettre dans sa file de requête l'identificateur du nœud qui a envoyé cette requête.
La figure 3.2 (a) illustre un arbre enraciné pour un système composé de neuf nœuds, si le nœud D envoie une requête, alors elle va traverser le chemin *D-A-C-F*.

Considérons le scénario suivant présenté dans la Figure 3.2 (b), où les nœuds *A* et *D* se déplacent de telle sorte que le lien physique entre *A* et *D* est rompu, et qu'un nouveau lien entre A et F soit formé.
Pour l'algorithme d'exclusion mutuelle le lien logique entre *A* et *C* existe, si *D* envoie une requête, le protocole de routage est invoqué pour trouver le nœud *C*, donc la requête va traverser le chemin *D-A-F-C-F* qui contient un cycle, ce qui augmente le coût en terme de nombre de messages et d'énergie consommée.

Cette solution ne convient plus à un environnement de réseaux ad hoc, où le mouvement des nœuds peut engendrer la création de nouveaux liens. De plus, les liens logiques dans cet algorithme ne correspondent pas aux liens physiques, ce qui fait que les messages peuvent traverser des chemins physiques qui contiennent des cycles.

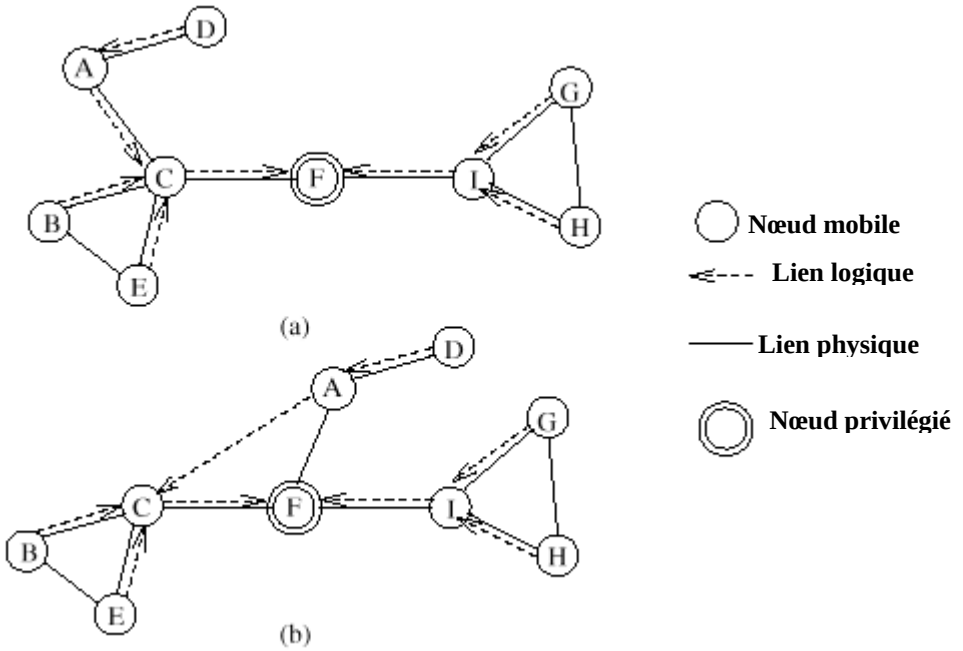


Figure 3.2

L'inconvénient présenté dans l'architecture de [WAL 97] fait que l'algorithme d'exclusion mutuelle a besoin d'être implémenté en utilisant une autre architecture.

Pour palier à cet inconvénient Walter et al. [WAL 98] implémente l'algorithme d'exclusion mutuelle au niveau de la couche réseau (Figure 3.3) où l'algorithme dans ce cas traite les problèmes de création et destruction de liens.

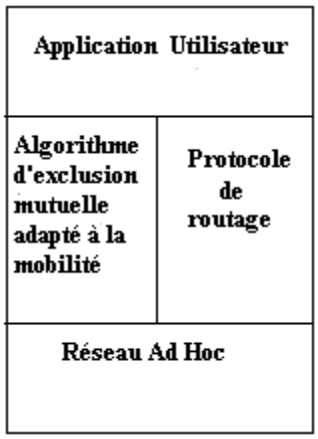


Figure 3.3

Le système contient n unités mobiles communicantes entre elles par envoi de messages. Chaque unité mobile exécute un processus d'application et un processus d'exclusion mutuelle.

Les processus au niveau d'un nœud *i* sont déclenchés par un ensemble d'évènements d'entrée présentés dans la figure 3.4.
$RequestCS_i$: Le processus d'application dans le nœud *i* demande d'accéder à la section critique et l'algorithme entre dans l'état *WAITING*.
$ReleaseCS_i$: Le processus d'application dans le nœud *i* sort de sa section critique et l'algorithme entre dans l'état *REMAINDER*.
$Recv_i(j,m)$: Le nœud *i* reçoit un message m du nœud *j*.
$LinkUp_i(l)$: Le nœud *i* reçoit une notification indiquant qu'un lien *l* est formé.
$LinkDown_i(l)$: Le nœud *i* reçoit une notification indiquant qu'un lien *l* est rompu.

La fonction de transition prend en entrée l'état du processus d'exclusion mutuelle et les évènements d'entrée pour produire des évènements de sortie et le nouvel état du processus. Les événements de sortie que le nœud *i* peut produire sont :
$EnterCS_i$: Le processus d'exclusion mutuelle dans le nœud i informe le processus d'application qu'il peut entrer dans l'état *CRITICAL*.
$Send_i(j,m)$: Le nœud *i* envoie le message m au nœud *j*.
Les évènements $RequestCS_i$, $EnterCS_i$ et $ReleaseCS_i$ sont appelés des évènements d'application quant au $Send_i$, $Recv_i$, $LinkUp_i$ et $LinkDown_i$ sont appelés des évènements réseaux.

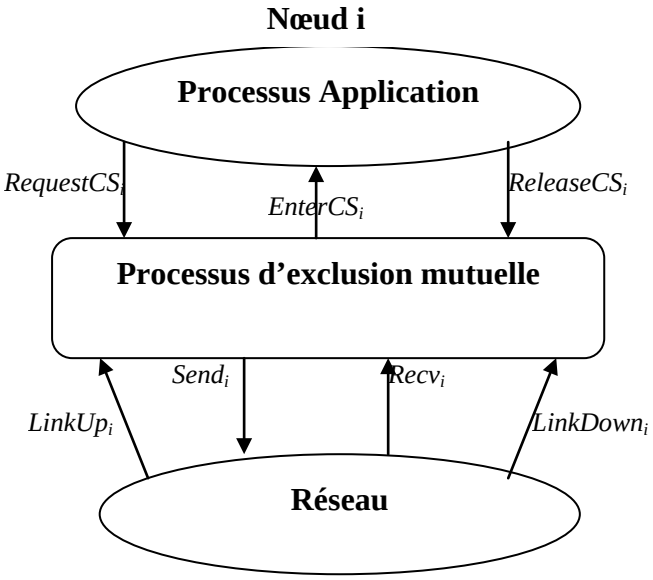


Figure 3.4

3.2.1. Hypothèses

1) Chaque nœud a un identificateur unique.
2) La défaillance de liens ne peut pas se produire.
3) Les liens de communication sont bidirectionnels et FIFO.
4) La couche de liaison assure que chaque nœud connaît l'ensemble de ses voisins en fournissant un indicateur de formation ou de défaillance de liens.
5) Le partitionement du réseau ne peut pas se produire.

3.2.2. Structure de données

Chaque nœud i maintient à son niveau les informations suivantes :
état : Indique l'état où se trouve le processus d'application, *WAITING*, *CRITICAL*, *REMAINDER*.
N_i : L'ensemble des voisins d'un nœud *i*.
$myheight_i$: Une taille d'un nœud i :triplet (α_i, β_i, i).
Height[j] : Un tableau qui contient les tailles des voisins.
$tokenHolder_i$: Un booléen qui vaut vrai si le nœud *i* possède le jeton et faux dans le cas contraire. Initialement $tokenHolder_i$ =vrai pour i=0 et faux pour $i \neq 0$.
$Next_i$: désigne le nœud voisin sur le chemin qui mène au nœud privilégié.
Q_i : file qui contient l'identificateur des nœuds voisins et de *i* qui ont demandé d'accéder en section critique.

3.2.3. Fonctionnement

Quand un nœud *i* veut entrer en section critique, il met son identificateur dans sa file Q_i et envoie un message *Request* au nœud désigné par $Next_i$.

Quand un nœud *i* reçoit *Request* d'un nœud *j*, il exécute la séquence suivante :
a) Il met l'identité de *j* dans la file.
b) Si le nœud *i* n'a qu'un seul élément dans Q_i, il envoie un message *Request* au nœud désigné par $Next_i$, sinon il attend de recevoir le jeton. Le message *Request* va suivre les liens orientés jusqu'à ce qu'il arrive au nœud privilégié.

Les figures 3.5(a) et 3.5 (b) montrent le cas où les nœuds 3 et 2 envoient leur message *Request* au nœud privilégié 0. Dans la figure 3.5(c), le nœud 1 envoie un message *Request* au nœud 3 dont la taille de sa file Q_3 est supérieure à 1(il n'a pas besoin d'envoyer un message *Request*) alors il attend de recevoir le jeton.

Quand un nœud *i* sort de sa section critique, il exécute la séquence suivante :
a) Il enlève le premier élément *j* de Q_i.
b) Il affecte à $Next_i$ le nœud *j*.
c) Il envoie le message *Token* à $Next_i$.
d) Si la file Q_i reste encore non vide alors *i* va envoyer *Request* à $Next_i$.
e) Il met son état à *REMAINDER*.

Quand *j* reçoit le message *Token*, il exécute la séquence suivante :
a) Il réduit sa taille pour qu'elle devienne inférieure à celle du nœud privilégié précédent (nœud *i*) comme suit :

$$\alpha_j = min\ \{\alpha_k \mid k \in N_j\} \qquad \beta_j = \begin{cases} min\ \{\beta_k \mid k \in N_j \wedge (\alpha_j = \alpha_k \wedge \beta_j \geq \beta_k)\} - 1 \\ \beta_j\ sinon \end{cases}$$

b) Il informe tous ses voisins sortants de sa nouvelle taille.
c) Il enlève le premier élément de Q_j (nœud k).
d) Si *j*= *k*, alors Il met son état à *CRITICAL* et entre dans sa section critique.
e) Si *j*≠ *k*, alors il va envoyer le message *Token* à *k*, si la file Q_j reste encore non vide alors *j* va envoyer *Request* à *k*.

La figure 3.5(c) montre le cas où le nœud 0 sort de sa section critique, il envoie un message *Token* au premier élément enlevé de la file Q_0 (le nœud 3). Q_0 reste encore non vide (le nœud a besoin encore de recevoir un message *Token*) alors il envoie un message *Request* au nœud 3. Le nœud 3 trouve que le premier élément de Q_3 est égale à 3, donc il entre dans sa section critique. La même chose est répétée pour les nœuds 1 et 2 dans les figures 3.5 (d) et 3.5(e).

Quand un nœud *i* détecte que le lien avec un nœud *j* est rompu, il vérifie si *j* est le dernier lien sortant, si ce n'est pas le cas alors *i* ne va prendre aucune action.
Si *j* est le dernier lien sortant, la procédure d'inversement de liens est appelée, cette procédure va mettre à jour les variables α_i et β_i comme suit :

$$\alpha_i = \min\ \{\alpha_i \mid k \in N_i\} + 1 \qquad \beta_i = \begin{cases} min\ \{\beta_j \mid j \in N_i \wedge (\alpha_i = \alpha_j)\} - 1 \\ \beta_i\ sinon \end{cases}$$

Si la file Q_j contient *i*, alors *i* sera supprimé de cette file.
Tous les nœuds voisins dont leur taille est inférieure à la taille de *i* seront supprimés de Q_i. si Q_i reste encore non vide alors il envoie un message *Request* au nouveau nœud désigné par $Next_i$.

La figure 3.6(b) illustre le cas où le lien entre les nœuds 3 et 0 est rompu. Le nœud 0 enlève 3 de sa file Q_0. Le nœud 3 perd son dernier lien sortant. Dans la figure 3.6(c), le nœud 3 met à jour sa taille et envoie un message *Request* à $Next_3$ (nœud 1). Dans les figure 3.6(d) et 3.6(e) Le message *Request* traverse le chemin (3,1,2) où le nœud 2 a déjà envoyé une requête et attend la réception d'un message *Token*.

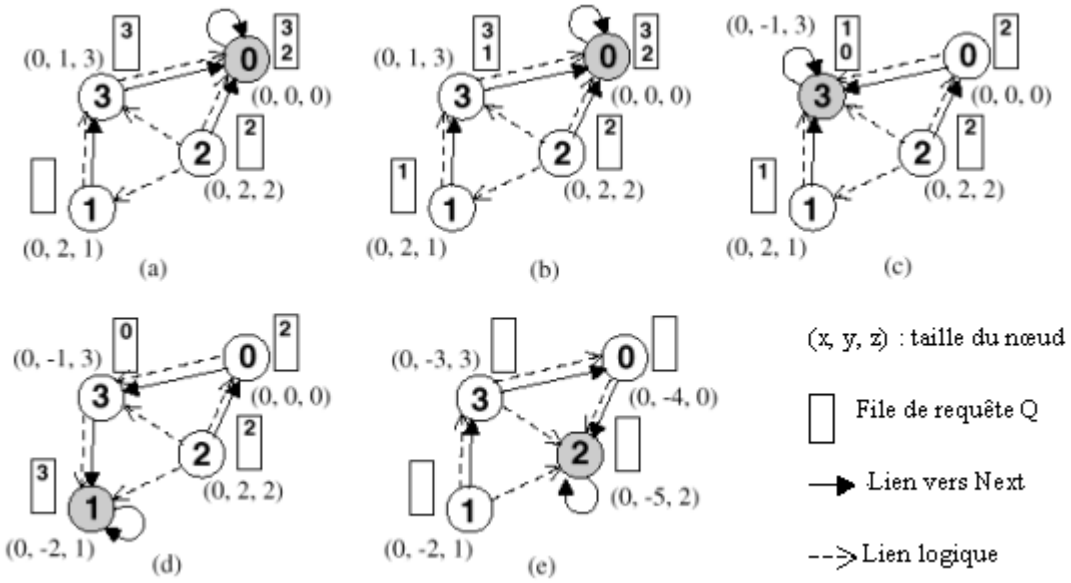


Figure 3.5

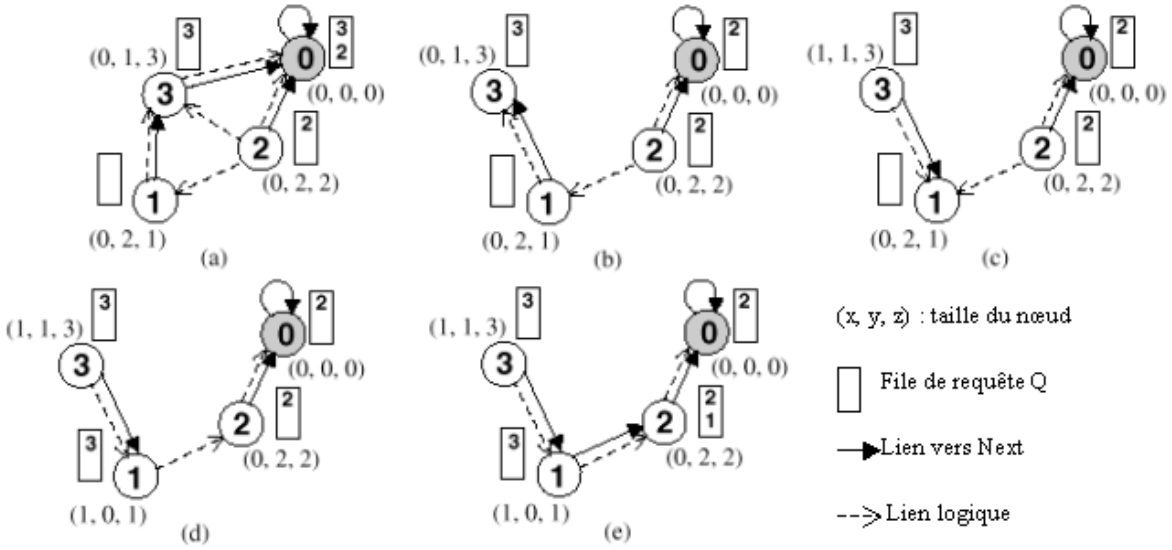


Figure 3.6

3.2.4. Les avantages de l'algorithme RL

1- La structure logique de la topologie du réseau correspond aux liens physiques en traitant les cas de destruction et formation de liens.
2- L'algorithme d'exclusion mutuelle ne dépend pas d'un protocole de routage spécifique.
3- Les nœuds ne contiennent des informations que sur leurs voisins directs.

3.2.4. Les inconvénients de l'algorithme RL

1- Quelques hypothèses ne sont pas réalistes comme le partitionement du graphe ne peut pas se produire.
2- L'algorithme prend en charge les cas de défaillance et formation de liens, ce qui représente un travail supplémentaire à l'algorithme et une charge supplémentaire en termes de communication et de consommation d'énergie.
3- Un nœud qui a envoyé une requête et perd son lien vers son voisin *Next*, renvoie la requête vers un autre voisin, ce qui augmente le temps d'attente pour un nœud pour avoir le jeton et peut produire une situation de famine si les changements de la topologie ne s'arrête pas.
4- Des nœuds qui n'ont pas demandé l'accès en section critique participe à l'exécution de l'algorithme.

3.3. L'algorithme de k-exclusion mutuelle (K Reverse Link) [WALa 01]

3.3. 1. Introduction

Le problème de k-exclusion mutuelle suppose l'existence de k ressources dans le système, partagées entre n processeurs, tel que : $1 \leq k \leq n$, ce qui implique que k processeurs peuvent être dans leur section critique en même temps.

Cet algorithme est une généralisation de l'algorithme de 1-exclusion mutuelle présenté dans le paragraphe (3.2). Il maintient k jetons dans le système ; quand k =1, le nœud qui a le plus faible identificateur va prendre le jeton et quand k > 1, il y' a k processeurs de 0 à k-1 que chacun d'eux va prendre un jeton.

3.3. 2. Hypothèses

1- Chaque nœud a un identificateur unique de 0 à n-1.
2- Les liens de communication sont bidirectionnels et FIFO.
3- La couche de liaison assure que chaque nœud connaît ses voisins.
4- Les nœuds qui possèdent un jeton ne tombent pas tous en panne.
5- Le partitionement du réseau est permis, tel que une portion du réseau doit contenir au moins un nœud qui possède un jeton et qui va continuer d'exécuter l'algorithme avec le sous-ensemble de jetons dont la partition.

3.3. 3. Fonctionnement de l'algorithme

Quand un nœud *i* fait une requête pour accéder à la section critique, *i* est mis dans la file Q_i et un message *Request* est envoyé à un voisin k ($Next_i$) qui a la plus faible taille. *i* sera mis dans Q_k et *k* envoie un autre message *Request* si la taille de Q_k égale a 1. Le message *Request* traverse les lien sortant jusqu'à ce qu'il atteint l'un des nœuds privilégiés.

Quand un nœud *i* reçoit le jeton, il enlève le premier élément de Q_i (exemple :*j*) si *j* égale à *i*, alors il donne la permission à son application, sinon il envoie le jeton au nœud *j*. Chaque fois qu'un nœud reçoit un jeton, il réduit sa taille par rapport au nœud privilégié précédent.

La figure 3.7(a) illustre la cas où les nœud 3 et 2 envoient leur requête aux nœud privilégié 0, et le nœud 4 envoie sa requête au nœud privilégié 1. Les figures 3.7 (b, c, d,e, f) montre la circulation du jeton dans le cas d'une topologie statique.

Quand un lien (*i, j*) est rompu, le nœud *j* enlève *i* de sa file Q_j et le nœud *i* envoie un autre message *Request* au nouveau nœud $Next_i$ comme il est illustré dans la figure 3.8 dans le cas de la destruction du lien (3,0).

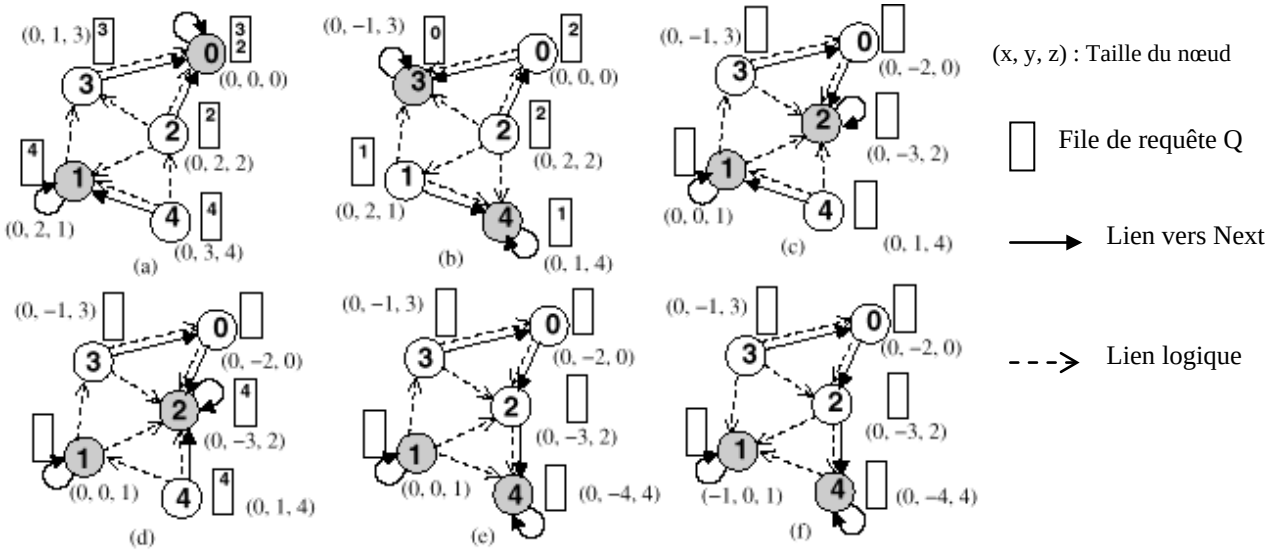


Figure 3.7

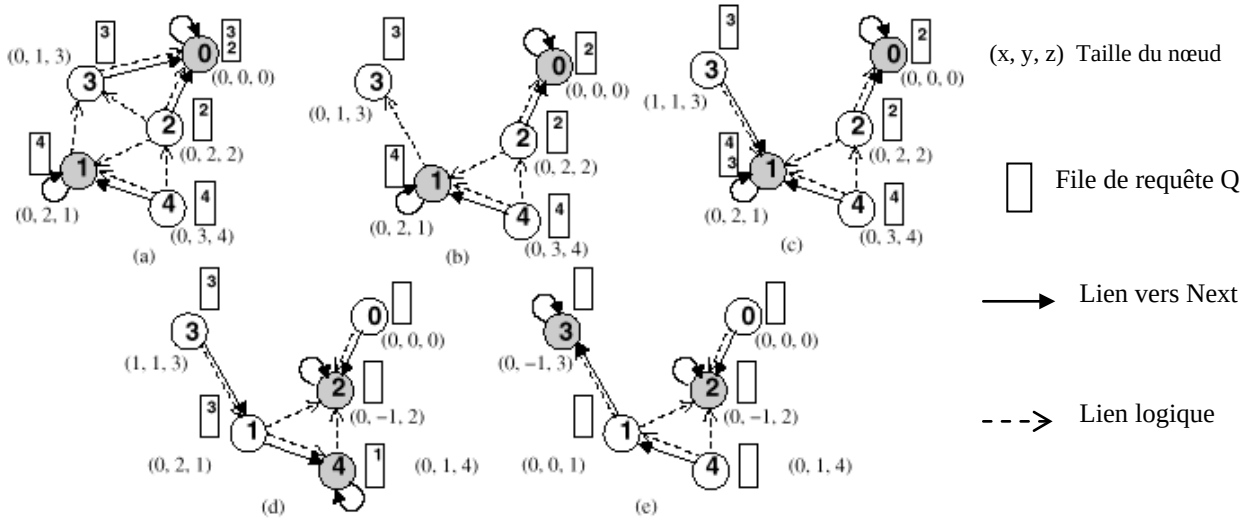


Figure 3.8

3.3.4. L'algorithme de KRL avec envoi du jeton [WALb 01]

La figure 3.9 montre l'exécution de l'algorithme dans lequel il y'a deux nœuds qui possèdent le jeton (3 et 5). Les nœuds 4 et 6 font leurs requêtes au nœud 5, le processus d'application du nœud 5 est dans la section critique et il a mis 4 et 6 dans sa file de requêtes Q_5.
Le processus d'application au niveau du nœud 3 quitte la section critique, le nœud 3 envoie le jeton que lorsque qu'il reçoit un message de requête, mais les nœuds 4 et 6 doivent attendre leur tour pour avoir le jeton utilisé par le nœud 5.

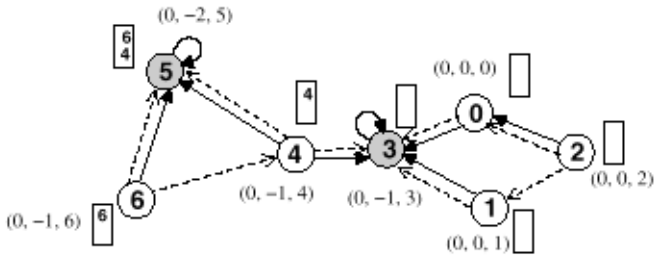


Figure 3.9

Pour remédier à ce problème, le nœud envoie le jeton à d'autres parties du réseau. Pour cela, il choisit le voisin qui la plus faible taille et envoie le jeton à ce voisin.
Le choix du plus faible voisin est dû au faible nombre de liens inversés. Les nœuds gardent une trace de leurs voisins qui ont envoyé le jeton et qui ont une file de requête vide en marquant le lien comme visité.

La figure 3.10 illustre ces modifications durant l'exécution de l'algorithme.
Dans la figure 3.10(a), le nœud 3 a un jeton, mais aucun voisin ne veut accéder en section critique.
La figure 3.10(b) montre l'état du système après que 3 quitté sa section critique, et le jeton est envoyé à travers 0, 2,1 et 3 (partie gauche du réseau), La lettre V sur chaque lien signifie que ce lien est marqué comme visité par le nœud émetteur et le nœud récepteur du jeton.
Quand le nœud 3 reçoit le jeton et il a une file de requête vide, il marque tous ses liens comme non visités et recommence le processus d'envoi du jeton.

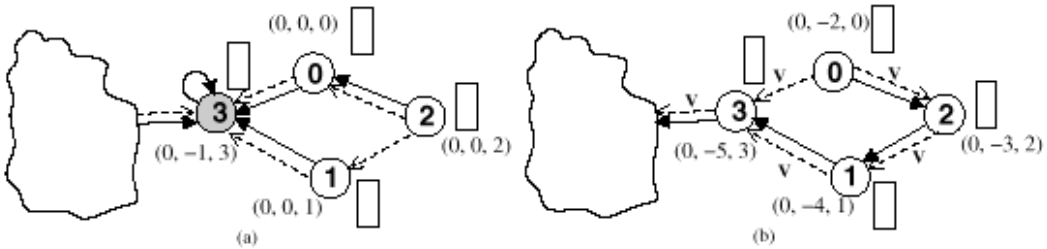


Figure 3.10

3.4. L'algorithme de Baldoni [BAL 02]

L'algorithme appartient à la famille des algorithmes demandant du jeton, il structure les processus en anneau logique.
L'anneau logique des processus est dynamique, chaque fois qu'un processus reçoit un jeton de son prédécesseur, il décide qui sera son successeur parmi les processus qui n'ont pas encore reçu le jeton durant un tour du jeton.
Cette décision est basée sur une politique donnée P. Dans cet algorithme, la politique P consiste à ordonner les processus P_i (ordre total) suivant leur nombre de sauts nécessaires pour atteindre P_i, chaque processus envoie le jeton au processus qui nécessite le moins de sauts.

3.4.1. Hypothèses

1- Les processus ne tombent pas en panne.
2- Les liens de communication sont fiables.
3- Le réseau n'est pas l'objet d'un partitionement permanent (si un partitionement se produit, alors chaque pair de processus pourra après un temps fini d'établir une communication).

3.4.2. Le fonctionnement de l'algorithme

Un processus qui souhaite accéder à la section critique envoie une requête au coordinateur et attend de recevoir le jeton, le coordinateur c_k (coordinateur d'un tour

k) insère la requête dans un ensemble nommé *pendingRequest (P)* ordonné selon un certain critère (le processus qui a l'identité la plus faible, FIFO, le processus le plus proche en terme de nombre de sauts,…etc).

Le coordinateur c_k envoie le jeton au premier processus dans la liste *pendingRequest(P)* qui sera le coordinateur c_{k+1}. Le jeton circule dans un anneau logique composé de (n-1) processus (le processus correspond au coordinateur précédent c_k est exclu de l'anneau). La structure de l'anneau logique sera construite à la volée.

Quand un processus p_i reçoit le jeton, il le transmete vers son successeur p_j qui remplit certains critères :

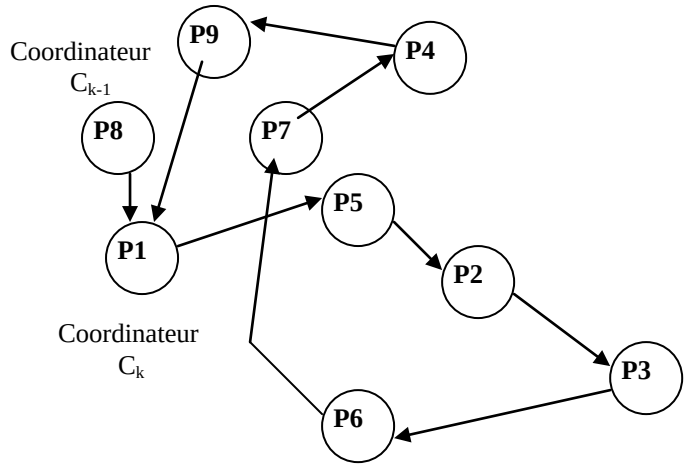
1- P_j n'a pas reçu le jeton pendant le tour courant et il est différent de c_k.
2- P_j est le plus proche en terme de nombre de sauts.

Quand le jeton passe par tous les processus, il sera envoyé au coordinateur c_{k+1}.

Si un processus est temporairement déconnecté, sa distance sera considérée infinie, si deux processus ont la même distance alors celui qui a l'identificateur le plus faible sera le prochain nœud privilégié.

Si le processus p_i n'a que des processus qui ont des distances infinies, p_i retransmet le jeton après un certain temps en supposant que p_j sera accessible et va recevoir le jeton.

La figure 3.11 décrit le fonctionnement de l'algorithme ou le coordinateur c_{k-1} envoie le jeton au coordinateur c_k qui fait que le jeton passe par tous les processus du réseau. Chaque processus p_i construit son ensemble $PossSucc_i$ qui contient les prochains nœuds privilégiés possible.



$PossSucc_1$={5,6,2,3,7,9,4}
$PossSucc_5$={2,3,7,9,6,4}
$PossSucc_2$={3,7,9,6,4}
$PossSucc_3$={6,7,9,4}
$PossSucc_6$={7,9,4}
$PossSucc_7$={4,9}
$PossSucc_4$={9}
$PossSucc_9$={}

Figure 3.11

3.4.3. Avantages

1- L'anneau est construit dynamiquement par l'envoi du jeton au processus qui demande le moindre effort en termes de ressource du réseau et de l'énergie.
2- La politique P présente une bonne heuristique, qui tient compte de la mobilité des nœuds avec la supposition que l'algorithme utilise les informations fournies par le protocole de routage.
3- La politique P peut être substitué sans qu'elle ait un impact sur le reste de l'algorithme.

3.4.4. Inconvénients

1- Pour une seule requête, le jeton doit traverser tout l'anneau logique, ce qui représente une charge supplémentaire en terme de communication et consommation d'énergie.
2- L'algorithme définit un ordre total sur tous les processus du système (chaque nœud connaît les information de routage de tous les nœuds du réseau), ce qui implique que le protocole de routage exécuté par les nœuds est proactif.
3- Des nœuds qui n'ont pas demandé l'accès en section critique participent à l'exécution de l'algorithme.

3.5. Conclusion

Dans ce chapitre nous avons présenté plusieurs algorithmes d'exclusion mutuelle qui ont été proposés pour résoudre le problème d'accès concurrent dans un réseau mobile Ad Hoc. Nous avons décrit leurs principales caractéristiques et fonctionnalités ainsi que leurs avantages et inconvénients.
Les algorithmes proposés sont classés en deux catégories. On trouve les algorithmes d'exclusion mutuelle qui règlent l'accès concurrent à une ressource, et les algorithmes d'exclusion mutuelle qui règlent l'accès à k exemplaires d'une même ressource.

Ces protocoles ne donnent pas de bonnes performances en termes de messages et d'énergie. Ils introduisent dans leur exécution d'autres nœuds qui n'ont pas demandé l'accès à la section critique.

Assurer l'exclusion mutuelle à une ressource dans un réseau Ad Hoc est un problème très complexe à cause du changement rapide de la topologie et le risque de panne de sites. La perte du jeton qui engendre l'exécution d'un protocole d'élection [HAT 99, MAL 00] pour regénérer un jeton.

La résolution du problème d'exclusion mutuelle dans un réseau Ad Hoc nécessite l'amélioration des techniques proposées afin de donner de meilleures performances et garantir les propriétés d'exclusion mutuelle et offrir une meilleure adaptation à la mobilité des unités mobile et de forcer l'exécution du protocole d'exclusion mutuelle seulement entre les nœuds qui demandent l'accès à la section critique.

Chapitre 4

Présentation d'un nouvel algorithme d'exclusion mutuelle dans un réseau mobile Ad Hoc

4.1. Introduction

Les algorithmes d'exclusion mutuelle dans un réseau Ad Hoc présentés dans le chapitre précédent ont quelques inconvénients, mais l'inconvénient commun entre ces algorithmes est la participation des nœuds qui n'ont pas demandé l'accès à la section critique dans l'exécution de ces algorithmes, ce qui représente une charge supplémentaire en terme de message et d'énergie pour les unités mobiles.

Dans ce chapitre, nous allons présenter un nouvel algorithme basé sur les jetons, il s'exécute seulement entre les nœuds qui désirent entrer dans la section critique ainsi que le nœud privilégié.

Cet algorithme s'exécute au-dessus de la couche réseau. Cette dernière utilise deux protocoles de routage, Chaque protocole est utilisé pour router un type de message donné. Pour délivrer un jeton au prochain nœud, le protocole de routage utilise la notion d'algorithme local et global, présentés dans [MAL 01].

Nous utilisons aussi une autre politique pour ordonner les requêtes. Cette politique permet de s'adapter aux changements de la topologie du réseau et de minimiser le nombre de messages de routage transmis pour transmettre le jeton.
Nous présentons aussi une méthode qui permet de reprendre l'exécution de l'algorithme d'exclusion mutuelle quand il y'a une perte du jeton.

4.2. Modèle du système

Nous considérons un système composé de n unités mobiles. Dans chaque unité mobile, il y'a un algorithme d'exclusion mutuelle qui s'exécute au-dessus de la couche réseau. La couche réseau contient plusieurs protocoles de routage, où chaque type de message est routé par un protocole de routage donné afin que l'algorithme d'exclusion mutuelle donne une consommation minimale en termes de nombre de sauts et de nombre de messages générés par le protocole de routage.

L'algorithme d'exclusion mutuelle utilise deux types de messages, un message de requête et un message jeton.
Dans la couche réseau, nous utilisons le protocole de routage TORA [PAR 97] pour router les messages de requête et un protocole de routage réactif de type source routing similaire à celui présenté dans [JOH 96] pour router le message jeton.

Nous pouvons modéliser l'interaction entre les couches par l'automate présenté dans la figure 4.1

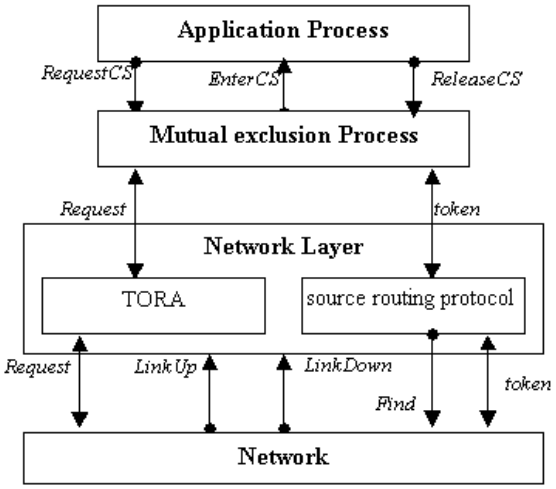


Figure 4.1 Interaction entre les couches du réseau dans une unité mobile

Les messages transmis entre les couches sont :

Les messages d'application
RequestCS : Le processus d'application demande d'entrer dans sa section critique.
ReleaseCS : Le processus d'application quitte sa section critique.

Les messages de l'algorithme d'exclusion mutuelle
EnterCS : Le processus d'exclusion mutuelle informe le processus d'application qu'il peut entrer dans sa section critique.
Request : Le processus d'exclusion mutuelle demande au protocole de routage TORA d'envoyer le message au nœud qui possède le jeton.
token : Le protocole de routage type source routing envoie le jeton au prochain nœud privilégié.

Les messages de la couche MAC
LinkUp : Informe le protocole de routage de la création d'un lien.
LinkDown : Informe le protocole de routage de la défaillance d'un lien.

4.3. Hypothèses de l'algorithme

1- Chaque nœud a un identificateur unique.
2- Les liens de communication sont fiables, bidirectionnels et FIFO.
3- La couche de liaison (MAC) assure que chaque nœud connaît ses voisins et informe le protocole de routage de la création ou la défaillance d'un lien.
4- Le protocole TCP est utilisé au niveau de la couche transport.
5- Un nœud ne peut pas rester dans la section critique indéfiniment.
6- Le partitionement du réseau est temporaire.

4.4. Structure du message jeton

Un champ *type* : représente le type du message.
Un champ *total* : représente le nombre de visites effectuées par le jeton aux nœuds du réseau.
Une file *Q* : représente la file de requête, elle contient l'identité de tous les nœuds qui sont en attente pour entrer en section critique.

La file *Q* est constituée d'un ensemble d'éléments Bag_k, k $\in$ N est le numéro de séquence de l'élément Bag_k, ces éléments sont ordonnés suivant la politique FIFO.
Bag_k est un ensemble, qui contient l'identité des nœuds qui ont demandé l'accès à la section critique et dont la requête est reçue par le même nœud.

Les éléments de Bag_k sont ordonnés suivant une politique FIFO ou autre que FIFO, par exemple : le nombre de sauts entre les nœuds de Bag_k et le nœud qui possède le jeton, ou le nœud le moins récemment visité par un jeton.
Cette politique garantit qu'aucune requête de Bag_{k+1} n'est satisfaite avant que toutes les requêtes de Bag_k ne soient satisfaites.

4.5. Les structures de données au niveau de chaque nœud i

Au niveau de l'algorithme d'exclusion mutuelle
$Requesting_i$: Un booléen qui indique si le nœud i a demandé l'accès à la section critique.
$tokenHolder_i$: Un booléen qui indique si le nœud i possède le jeton.
$token\text{-}copy_i$: une copie d'un jeton
$token\text{-}seq_i$: indique quand est ce que i a reçu le jeton pour la dernière fois.

Au niveau du protocole de routage TORA
$myHeight_i$: Un 5-uplet $(\tau_i , oid_i , r_i , \delta_i , i)$ représente la taille d'un nœud i dans le DAG.
height[j] : Tableau des tailles de ses voisins dans le DAG.
$Next_i$: Représente le prochain nœud dans la route qui mène au nœud privilégié.
N_i : L'ensemble des voisins d'un nœud i.

Au niveau du protocole de routage source routing
Table de routage qui contient les champs suivants :
Source : désigne l'identité du nœud source de la route.
NextToken : désigne l'identité du nœud destination de la route.
NextHop : désigne le prochain saut dans la route qui mène à la destination

4.6. Fonctionnement de l'algorithme

1) Initialisation du DAG

Un nœud donné commence la construction du DAG comme il est décrit dans [PAR 97]. La figure 4.2(a) décrit la construction du DAG pour la destination F.

2) Demande d'accès en section critique

Quand un processus d'application d'un nœud *i* désire entrer dans la section critique, il envoie un message *RequestCS* à son algorithme d'exclusion mutuelle, il met la variable *Requesting* à vrai et envoie un message de requête *Request* au protocole de routage TORA dans la couche réseau.

Le protocole TORA consulte sa taille, si sa taille est nulle c'est-à-dire que le graphe est partitionné, et la requête de i ne peut pas arriver au nœud privilégié, dans ce cas la requête va être mise en attente jusqu'à ce que la taille de *i* soit non nulle, dans l'autre cas (taille est non nulle), il envoie le message *Request* au nœud désigné par la variable $Next_i$.

Au niveau de la couche réseau, un nœud intermédiaire qui reçoit un message, vérifie son type, si c'est un message *Request*, alors il va le délivrer au protocole TORA qui va l'envoyer au prochain nœud sur la route qui mène au nœud privilégié.

Quand le message *Request* atteint le nœud désiré, il va être livré à l'algorithme d'exclusion mutuelle, si cette requête est la première qui arrive à ce nœud alors il va incrémenter *seq* et il va ajouter dans la file *Q* l'ensemble Bag_{seq} qui va inclure l'identité de la nouvelle requête et les autres requêtes qui vont arriver.

Dans la figure 4.2(b), les requêtes des nœuds D, C et G qui arrivent au nœud privilégié F sont mises dans l'ensemble Bag_1 et dans la figure 4.2(c) les requêtes des nœuds H et B qui arrivent au nœud privilégié G sont mises dans l'ensemble Bag_2.

3) Envoi du jeton

Quand un nœud i possédant le jeton n'est pas dans sa section critique et sa file de requête n'est pas vide, alors il va envoyer le jeton à l'un des nœuds qui sont dans Bag_k, tel que k est le numéro de séquence minimal de la file *Q* du jeton.

Pour choisir le nœud qui va recevoir le jeton, il va envoyer un message $token(Bag_k)$ au protocole de routage source routing, qui lui demande de trouver un nœud de Bag_k qui a le minimum de sauts entre lui et le nœud *i*.

Le protocole source routing vérifie s'il y' a des voisins dans l'ensemble Bag_k, alors il va choisir le voisin qui a la taille minimale pour lui envoyer le jeton, dans le cas

contraire, s'il n' y a pas des nœuds voisins dans l'ensemble Bag_k, il diffuse un message *Find(Bag_k)*.
Un nœud j qui reçoit ce message va vérifier si son identité est dans l'ensemble Bag_k, si c'est le cas alors il va enlever son identité de l'ensemble Bag_k et va envoyer un message *Reply* au nœud i.

Quand un nœud i reçoit le premier message *Reply(j)* d'un nœud j, (on suppose que le message *Reply(j)* est arrivé en premier parce qu'il a parcouru le moindre nombre de sauts), alors il va envoyer le jeton à j.
Si i ne reçoit aucun message *Reply* (c'est à dire qu'il y'a un partitionement et il doit attendre afin que le réseau se joigne à une autre partition), il met les requêtes de l'ensemble Bag_k dans l'ensemble Bag_{k+1} et envoie un message *find(Bag_{k+1})*.

Un nœud intermédiaire qui reçoit le jeton, décrémente sa taille en utilisant les formules suivantes :

$$(\tau_i , oid_i , r_i)=min \{(\tau_j , oid_j , r_j) \mid j\in N_i \}$$

$$(\delta_i , i)= min \{(\delta_j , j) \mid j\in N_i \text{ avec } (\tau_i , oid_i , r_i)= (\tau_j , oid_j , r_j)\}\text{-}1$$

Le nœud i envoie sa nouvelle taille à tous ses voisins, ensuite il consulte sa table de routage pour trouver le prochain saut qui mène au nœud j.
Quand le message jeton arrive à la destination désirée, le protocole de routage le livre à l'algorithme d'exclusion mutuelle, ce dernier va exécuter les séquences suivantes :
Il enlève son identité de la file *Q* du jeton.
Il garde une copie du jeton au niveau de l'algorithme d'exclusion mutuelle.
Il incrémente *token.total* et affecte sa valeur à $token\text{-}seq_i$.
L'algorithme d'exclusion mutuelle envoie un message *EnterCS* au processus d'application, et attend la réception d'un message *ReleaseCS*.

Dans la figure 4.2(c), le nœud privilégié F envoie un message *Find({D,C,G})* et attend la réception d'un message *Reply* tel que l'ensemble *{D,C,G}* désigne l'ensemble Bag_1, F reçoit 3 différent chemins : $path_1$*=F, H, G* , $path_2$*=F, E, B, D* et $path_3$*=F, E, B, A, C*. Le nœud privilégié choisit le chemin $path_1$ qui a le minimum nombre de sauts et envoie le jeton au nœud G. La figure 4.2(d) montre la circulation du jeton dans le réseau jusqu'à ce qu'il atteint le nœud D.

4) Création et défaillance de liens

Le traitement de création et défaillance de lien se fait au niveau de la couche réseau. Dans notre algorithme, le protocole de routage source routing ne réagit pas au changement de la topologie, parce qu'il n'utilise les routes qu'à la demande afin d'envoyer le message jeton, la réaction aux changements de la topologie se fait au niveau du protocole de routage TORA.

Quand un nœud i détecte que le lien vers j est rompu, il enlève j de l'ensemble V_i il vérifie si ce lien est le dernier lien sortant, si ce n'est pas le cas alors i ne va prendre aucune action.
Si j est le dernier lien sortant, la procédure d'inversement de liens (*Partial Reversal procedure*)est appelée, cette procédure met à jour les variables α_i et β_i comme suit :

$$(\tau_i , oid_i , r_i)=max \{(\tau_j , oid_j , r_j) \mid j\in N_i \}$$

$$(\delta_i , i)= min \{(\delta_j , j) \mid j\in N_i \text{ avec } (\tau_i , oid_i , r_i)= (\tau_j , oid_j , r_j)\}\text{-}1$$

Le nœud i envoie sa nouvelle taille à tous ses voisins vers lesquels il a des liens sortants. Quand un nœud j reçoit la nouvelle taille de i, cela peut faire que j perd son dernier lien sortant vers le nœud qui tient le jeton ($myheight_i < myheight_j$), dans ce cas j exécute la procédure d'inversement de liens.

Quand un nœud i détecte la création d'un lien vers un nœud j, alors il ajoute j dans l'ensemble N_i, si sa taille est nulle (jointure de deux partitions) alors il va vérifier s'il a une requête, alors il va l'envoyer au nœud j afin d'arriver au nœud privilégié et propage la construction du DAG aux autres nœuds dont leur taille est nulle.

La figure 4.2(e) et 4.2(f) montre le cas où un partitionement se produit dans le réseau, le nœud privilégié H ne peut pas localiser B, il conclut que B appartient à une autre partition, donc B sera enlevé de Bag_2 et sera mis dans Bag_3 avec le nœud E. Ce dernier reçoit le jeton et devient le nœud privilégié.

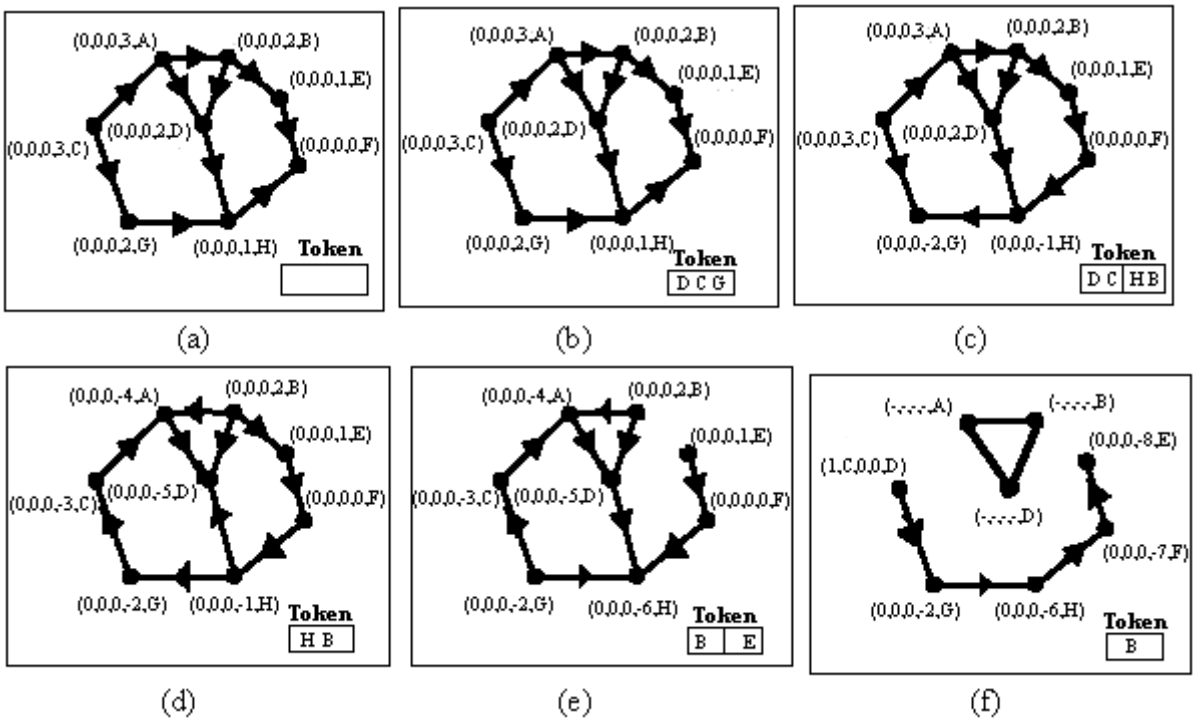


Figure 4.2

4.7. Code source de l'algorithme au niveau du nœud i

Dans cet algorithme, nous définissons les variable suivantes :
Bag_{min}/Bag_{max}: L'ensemble Bag qui a le numéro de séquence minimal/ maximal.
Insert(Bag, j): Insère j dans l'ensemble Bag de Q.
MaxSN(): Fonction qui retourne le numéro de séquence maximum dans Q.

Processus d'application

```
Begin
< non Critical Section >
Send RequestCS_i to mutual exclusion process;
Wait for EnterCS_i ;
< Critical Section >
Send ReleaseCS_i to mutual exclusion process;
End.
```

Processus d'exclusion mutuelle

```
Init()
Requesting_i := FALSE ;
tokenHolder_i := FALSE ;
firstReceivedRequest := FALSE ;
select a node k to become the privileged node
if i =k then
  tokenHolder_i := TRUE;
  token.Q :=∅ ;
  Initiate the creation  of the DAG;
endif ;
```

```
When RequestCS_i is received from AP
Do
Requesting_i := TRUE ;
if tokenHolder_i = TRUE
Then  Send EnterCS_i  to AP;
Else Send Request(i) to NL ;
endif ;
Endo ;
```

```
When ReleaseCS_i is received from AP
Do
Requesting_i := FALSE ;
tokenHolder_i :=FALSE ;
firstReceived :=FALSE ;
If token.Q ≠∅ Then
  Send token( Bag_min) to the Network Layer ;
endif ;
Endo ;
```

When *token* is received From NL
Do
token.Q := *token.Q* –{i};
token.total := *token.total* +1 ;
token-seq$_i$:= *token.total* ;
tokenHolder :=TRUE ;
token-copy$_i$:= token;
Send *EnterCS*$_i$ to the AP;
EnDo ;

When *Request(j)* is received From NL
Do
If *firstReceivedRequest* =FALSE **Then**
If *token.Q*=∅ **Then**
Insert(token.Q.Bag$_{min}$, j);
Send token to NL ;
Else
k:=MaxSN();
k :=k+1 ;
Create the set Bag$_k$;
Insert(token.Q.Bag$_{min}$, j);
endif ;
firstReceivedRequest :=TRUE ;

Else Insert(token.Q.Bag$_{min}$, j);
endif;
EnDo ;

Network Layer (NL)

When *token (Bag$_k$)* received From MEP
Do
Send token to *nextToken*;
EnDo ;

When *Request(j)* is received From MEP
Do
Send *Request(i)* To *Next*$_i$;
Endo;

When *Request (j)* is received From the lower Layer
Do
If i= *Next*$_i$ **Then**
deliver Request (j) to MEP
Else Send Request(j) to *Next*$_i$;
endif ;
EnDo ;

When *token(dest)* is received From the lower layer
Do
Update the height of i ;
Send myHeight$_i$ over all the outgoing links;
If *dest* =i **Then deliver** *token(dest)* to MEP;
  If (*nextToken*=i) **Then**
    If(|token.Q.Bag$_{min}$| > 1) **Then**
      Send Find (i, Bag$_{min}$, (i),1) to j $\forall$ j $\in$ *incomming$_i$*;
    Else if (|token.Q| > 1) **Then**
      Send Find (i, Bag$_{min+1}$, (i),1) to j $\forall$ j $\in$ *incomming$_i$*;
    endif;
 endif;
Else Send token(dest) to *nextHop*;
endif ;
Endo ;

When Find(src, Bag$_k$, path, hop) is received
Do
path:=*path* : i; /* **:** is an operation that append i to path */
If(|outgoing$_i$|=0) **Then**
  Send *Reply(path, src ,i, hop)* to *path[hop]*;
Else If (i$\in$ Bag$_k$)**Then**
  Send *Reply(path,src, i, hop)* to *path[hop]*;
  endif;
Else
  Send *Find(src, Bag$_k$, path, hop+1)* to j $\forall$ j $\in$ *incomming$_i$;*
endif;
Endo ;

When *Reply(path, src, dest, hop)* is received From j
Do
nextToken=dest;
nextHop=j;
if(src$\neq$ i) **Send** *Reply(path, src, dest, hop-1) to path[hop]*
Endo ;

When *linkDown(j)* received From j
Do
N$_i$:=N$_i$ -{j} ;
Update the height of i ;
If j=*Next$_i$* **Then**
  If $\exists$ outgoing node **Then**
    next= outgoing node That has the minimal height ;
   Else Execute partial reversal procedure;
  endif ;
endif ;
Endo ;

When *linkUp(j)* received From j
Do
$N_i := N_i \cup \{j\}$;
If $myheight_i$=(-, -,-, -, i) **Then**
  Update the height of i ;
  next:=j;
  Send $myheight_i$ to all j / j∈ N_i and height[j]=null ;
endif;
If ∃ request in wait **Then Send** Request to next ;
endif;
Endo ;

4.8. Preuve de la correction de l'algorithme

Définition : On dit qu'un nœud i est dans sa section critique, si $requesting_i$ = vrai et $tokenHolder_i$= vrai

Théorème 1 (*Condition de sûreté*): A tout moment, il y'a au plus un seul nœud dans sa section critique.

Preuve : On va montrer qu'à tout moment, il y'a au plus un seul nœud qui vérifie les conditions : $requesting_i$ = vrai et $tokenHolder_i$= vrai.

Initialement, un nœud donné *k* (le nœud privilégié) possède le jeton, donc $tokenHolder_k$=vrai, dans ce cas aucun nœud n'est dans sa section critique, quand le nœud *k* reçoit une requête d'un autre nœud *j*($requesting_j$=vrai), il met $tokenHolder_k$ à faux,.
Quand le jeton arrive au nœud j, il met $tokenHolder_j$ à vrai et entre dans sa section critique.

Quand il y'a plusieurs requêtes dans le système, chaque nœud met son *requesting* à vrai, le nœud qui possède le jeton va choisir un seul nœud à qui il va délivrer le jeton. Ce dernier, quand il reçoit le jeton met son *tokenHolder* à vrai et entre dans sa section critique. Ce qui montre qu'à tout moment, il y'a au plus un seul nœud dans sa section critique.

Lemme 1 : Une requête sera toujours livrée au nœud privilégié.

Preuve :
En absence de la mobilité, le nœud envoie sa requête au nœud désigné par la variable *Next*, cette requête va suivre les liens sortants définis par le DAG jusqu'à ce qu'elle arrive au nœud privilégié.
On considère maintenant le mouvement du jeton, en même temps que la requête traverse le graphe pour atteindre le nœud privilégié, dans ce cas après la stabilisation du jeton au niveau d'un nœud la requête va atteindre ce nœud en suivant les liens sortants du DAG qui garantit l'existence de chemin, entre tout nœud non privilégié et le nœud privilégié.

Dans le cas de la mobilité, quand il y'aura une défaillance d'un lien entre les nœuds i et j, le nœud i met à jour sa variable *Next*, soit en choisissant un autre voisin sortant ou par l'exécution d'une procédure d'inversement de liens s'il perd tous ses liens sortants. Ce qui implique qu'il y'a toujours à partir de n'importe quel nœud un chemin vers le nœud privilégié.

Si la défaillance du lien entre les nœuds i et j produit un partitionement du graphe, la requête sera mise en attente au niveau de la couche réseau du nœud i qui aura une taille indéfinie. Comme le partitionement n'est pas permanent, quand la partition du nœud i rejoint la partition du nœud privilégié, le nœud i va mettre à jour sa taille et sa variable *Next*. Ceci permet de construire un chemin vers le nœud privilégié.

Lemme 2 : Un nœud privilégié va délivrer le jeton au prochain nœud dans la file de requête en un temps fini.

Preuve :
En absence de la mobilité, quand le nœud privilégié choisit le prochain nœud qui a le nombre de sauts minimum pour qu'il puisse entrer dans sa section critique, il utilise la route construite par le protocole de routage pour délivrer le jeton au nœud destinataire.

Dans le cas de la mobilité, si le jeton arrive à un nœud intermédiaire trouve que le lien vers le prochain nœud est rompu, il fait appel à une procédure de recherche pour localiser le nœud désiré.

Si le nœud privilégié ne reçoit aucune réponse pour le message *Find(*Bag_k*)* il conclut que ces nœuds sont dans partitions autres que la partition du nœud privilégié, alors les requêtes de l'ensemble Bag_k seront mises dans l'ensemble Bag_{k+1} et un message *Find(*Bag_{k+1}*)* est diffusé.

En plus de ça, l'utilisation du protocole TCP[MAL 01] au niveau de la couche transport va garantir que le jeton ne va être perdu dans le cas d'une défaillance de route. Ce qui fait que le jeton va être éventuellement reçu par le nœud prochain dans la file *Q*.

Théorème 2 : (*Condition de vivacité*) Chaque processus qui envoie une requête, obtiendra l'accès à sa section critique.

Preuve : Nous montrons que chaque nœud qui envoie un message *RequestCS* reçoit un message *EnterCS*.

Dans le lemme 1, nous avons montré que chaque requête atteint le nœud privilégié sera mise dans la file de requête du jeton, cette file est constituée d'éléments Bag_i ordonnés suivant la politique FIFO, et dans ces Bag_i les requêtes sont ordonnées suivant le nombre de sauts par rapport au nœud privilégié.

Cette politique implique qu'une requête de Bag_{i+1} ne sera satisfaite qu'après que toutes les requêtes de Bag_i soient satisfaites.

Une requête qui appartient dans l'ensemble Bag_k sera servie après la satisfaction de toutes les requêtes de Bag_j, j<k et avant la satisfaction des requêtes de l'ensemble Bag_{k+1}.

Comme le partitionnement du réseau est temporaire, les requêtes du Bag_i seront satisfaites après un temps fini. Ce qui est démontre dans le lemme 2.

Ce qui implique que chaque nœud qui a envoyé une requête, recevra le jeton après un temps fini.

4.9. Les avantages de l'algorithme

1- Le processus d'exclusion mutuelle n'utilise aucune structure logique mais exploite celle de la couche réseau.
2- Chaque type de message (requête ou jeton) est routé par un protocole de routage donné ce qui rend la couche réseau flexible et extensible.
3- Le processus d'exclusion mutuelle dans les nœuds intermédiaires n'est pas invoqué, donc l'algorithme d'exclusion mutuelle s'exécute seulement entre les nœuds qui ont demandé l'accès à la section critique.
4- Le processus d'exclusion mutuelle n'a pas besoin de spécifier à qui il va envoyer les requête ou le jeton mais cette tâche est assurée par les protocoles de routage.
5- La file Q est ordonnée suivant une politique qui minimise le nombre de messages et assure la condition de vivacité

4.11. Les inconvénients de l'algorithme

1- La couche réseau supporte plusieurs protocoles de routage ce qui représente une charge supplémentaire en terme d'espace de stockage.
2- La taille du jeton est grand. Si *m* est le nombre de requêtes, alors sa taille sera $O(m \log_2 n)$ bits, mais elle sera toujours proportionnelle à *m*.

4.12. La perte du jeton

Dans le chapitre 4.3, Nous avons supposé que la perte de messages et la panne des nœuds ne se produit pas. Il est important dans la conception d'un algorithme d'exclusion mutuelle de prendre en compte ces types de défaillance parce qu'elles peuvent entraîner le blocage de l'algorithme, spécialement pour un algorithme basé sur les jetons, où un seul jeton circule dans le réseau.

Le jeton peut être perdu, soit parce que le nœud qui possède le jeton tombe en panne, soit parce que le message *token* a parcouru une route rompue.

Nous traitons ici un seul cas de défaillance c'est la panne des nœuds. Nous supposons l'existence d'un mécanisme qui permet à un nœud de détecter la panne d'un nœud voisin. Si un nœud tombe en panne, il y' a au moins un de ses voisins qui va rester correct.

Un nœud qui détecte la panne de son voisin j, diffuse dans le réseau un message *Fail(j)*, un nœud qui reçoit ce message va traiter trois cas.

Cas1 : Si le nœud défaillant ne participe pas à l'exécution de l'algorithme, il n' y aura aucun effet sur l'algorithme d'exclusion mutuelle, mais les protocoles de routage doivent réagir si ce nœud appartient à une route active.

Cas 2 : Si le nœud défaillant est en attente du jeton, son identificateur sera enlevé de la file de requêtes quand le nœud privilégié reçoit le message *Fail(j)*.

Cas 3 : Si le nœud défaillant est le nœud privilégié, un de ses voisins (nœud k) diffuse dans le réseau un message *tokenloss (j)*.
Un nœud i qui reçoit ce message exécute la séquence suivante :
Il enlève j de la file *Q* de $tokencopy_i$.
Il envoie au nœud qui a détecté la panne (nœud k) son $token\text{-}seq_i$.

Le nœud k choisit le nœud qui a un $token\text{-}\ seq_i$ maximum et il lui envoie un message pour qu'il regénère le jeton, ce dernier va générer un nouveau jeton *t* et affecte à t *le* contenu de $tokencopy_i$, ensuite il va demander à son protocole de routage TORA de construire un nouveau DAG.

4.13. Conclusion

L'algorithme présenté dans ce chapitre garantit les conditions d'exclusion mutuelle dans le cas de mobilité et des pannes, il s'exécute au-dessus d'un protocole de routage qui utilise deux techniques de routage, TORA pour router les messages de requête et un protocole réactif de type source routing pour router le message jeton, cette méthode permet d'introduire une flexibilité au protocole de routage et donner une grande performance.

Pour qu'un nœud délivre une requête au nœud privilégié il a besoin d'une connaissance locale, et pour que le jeton soit livré au nœud prochain il a besoin d'une connaissance globale.

La file de requêtes du jeton est ordonnée suivant une politique qui minimise le nombre de messages et d'énergie consommée et qui respecte la propriété de non famine.

Cet algorithme traite le cas de partitionement et le cas de panne des unités mobile.

Chapitre 5

Comparaison de performances entre les algorithmes d'exclusion mutuelle dans les réseaux Ad Hoc

5.1. Introduction

Comme nous avons présenté précédemment, les unités mobiles dans les réseaux Ad hoc se caractérisent par :
Une bande passante modeste.
Des sources d'énergies limitées comme les batteries.
Vitesse de CPU et capacité de mémoire moyenne.

D'après ces caractéristiques, il sera utile et indispensable d'étudier les performances de chaque algorithme d'exclusion mutuelle afin de sélectionner l'algorithme qui respecte le mieux les contraintes physiques et qui s'adapte bien aux changements de la topologie.

L'évaluation des algorithmes d'exclusion mutuelle se fait en deux façons, soit par le calcul de complexité de différents métriques, soit par simulation comme le simulateur Glomosim [ZEN 98].

L'étude de complexité permet de calculer plusieurs métriques spécifiques aux algorithmes d'exclusion mutuelle, et nous allons montrer l'impact de la mobilité sur ses métriques.
Evidemment, l'algorithme d'exclusion mutuelle le plus favori est celui qui résiste plus à la mobilité des nœuds et aux changements de la topologie.

5.2. Les métriques de complexité calculées

Pour mesurer et comparer entre les performances des algorithmes d'exclusion mutuelle, nous allons mesurer pour chaque algorithme les métriques suivantes :

1) Le nombre de messages échangés par accès en section critique (M): Combien de messages l'algorithme à besoin pour donner à un nœud l'accès à la section critique ?
Cette métrique est mesurée en deux cas :
Une charge faible : Au plus un seul nœud demande l'accès à la section critique.
Une charge élevée : Chaque nœud dans le réseau demande l'accès à la section critique.

2) Le délai de synchronisation (T): Le temps entre deux accès successifs en section critique. Il est exprimé par le nombre maximum de messages séquentiels nécessaire après qu'un nœud *i* quitte la section critique et avant qu'un autre nœud *j* entre dans la section critique (cette métrique est calculée sous une charge élevée).

3) Nombre de sauts par section critique : Le nombre de sauts traversés par les messages d'exclusion mutuelle (Les messages requêtes et jetons) dans le cas d'une charge élevée..

4) Complexité de routage (R): Le nombre de messages de contrôle générés par la couche réseau.

5) Complexité de contrôle : La taille des informations de contrôle en bit envoyés avec chaque message en bits.

Ces métriques vont être mesurées dans le cas d'une topologie statique et dans le cas d'une topologie dynamique, pour montrer l'impact de la mobilité sur le comportement de ses algorithmes et sur leurs métriques de complexité.

5.3. Les paramètres du réseau

Pour calculer les métriques de complexité aux pires des cas, on a besoin de définir quelques paramètres propres au réseau.
Parmi ces paramètres, on trouve :

N: Le nombre des nœuds.
m : le nombre de nœuds qui désirent l'accès à la section critique.
D : Le diamètre du réseau (le plus long chemin dans le réseau).
Δ : Le degré maximum d'un nœud dans le réseau (le nombre de voisins d'un nœud)

Nous allons comparer les algorithmes d'exclusion mutuelle suivants : l'algorithme Reverse Link (RL) [WAL 98], l'algorithme de Baldoni [BAL 02] et notre algorithme proposé dans le chapitre 4.

4.4. Les métriques mesurées dans le cas statique

5.4.1. Le nombre de messages échangés par entrée dans la section critique (M)

L'algorithme de [WAL 98] génère *O(2D)* messages par entrée dans la section critique : *O(D)* messages pour envoyer une requête, et *O(D)* messages pour envoyer le jeton. Dans le cas d'une charge élevée, il génère 4 messages.

L'algorithme de [BAL 02] génère *O(n+1)* messages dans le cas d'une charge faible (pour une seule requête le jeton va traverser tout l'anneau logique du réseau), et 2 messages dans le cas d'une charge élevée [BAL 01].

Notre algorithme nécessite l'envoi de 2 messages d'application par entrée dans la section critique (un message requête et un message jeton).

5.4.2. Le délai de synchronisation (T)

Pour que le droit d'accès à la section critique passe d'un nœud à un autre nœud, l' algorithme de [WAL 98] a besoin de générer *O(D)* messages jeton.

Tandis que dans l'algorithme de [BAL 02] et notre algorithme, un nœud qui sort de sa section critique envoie le jeton à son voisin ce qui nécessite 1 message.

5.4.3. Nombre de saut par section critique (Hop)

Dans l'algorithme de [WAL 98], les message d'exclusion mutuelle traversent 4 sauts.

Dans l'algorithme de [BAL 02] et notre algorithme, les messages de requêtes et le message jeton traversent *O(D+1)* sauts (*O(D)* sauts pour le message requête et 1 saut pour le message jeton).

5.4.4. Complexité de routage (R)

L'algorithme de [WAL 98] n'envoient des information de routage supplémentaire que dans le cas de circulation du jeton pour mettre à jour les tailles des nœuds, tandis que l'algorithme de [BAL 02] utilise un protocole de routage proactif où chaque nœud envoie périodiquement sa table de routage à tous les autre nœuds du réseau ce qui représente une complexité de $O(N^2)$. L'algorithme proposé utilise un protocole de routage de type source routing qui nécessite *O(N)* messages pour localiser le prochain nœud privilégié et *O(D)* pour envoyer un message *Reply*, ce qui nécessite une complexité de *O(N+D)*. Le tableau suivant résume les résultats présentés ci dessus :

		WAL 98	BAL 02	Notre algorithme
M	Charge Faible	*O(2D)*	*O(n+1)*	*2*
	Charge élevée	4	2	
T		*O(D)*	*1*	*1*
Hop		*4*	*O(D+1)*	*O(D+1)*
R		$O(\Delta)$	$O(N^2)$	*O(N+D)*

Figure 5.1

5.5. Les métriques mesurées dans le cas dynamique

5.5.1. Le nombre de messages échangés par entrée à la section critique

Dans l'algorithme de [WAL 98], dans le cas d'une charge faible, chaque fois qu'un nœud qui envoie une requête, trouve que le lien vers son nœud désigné par *Next* est rompu, il renvoie sa requête, ce qui nécessite *O(D)* messages supplémentaires.

Dans le cas d'une charge élevée, un nœud a besoin de retransmettre sa requête vers son voisin, ce qui représente 1 message supplémentaire.

Dans l'algorithme de [BAL 02], le changement de la topologie n'influe pas sur le comportement de l'algorithme car il s'exécute au-dessus d'un protocole de routage, ce qui fait $O(n+1)$ messages d'application dans le cas d'une charge faible et 2 messages d'application dans le cas d'une charge élevée.

Dans notre algorithme, le changement de topologie n'influe pas sur la valeur de M présenté dans la figure 5.1.

5.5.2. Le délai de synchronisation

Le changement de topologie n'influe pas sur la valeur de T présenté dans la figure 5.1. L'algorithme de [WAL 98] a besoin de générer $O(D)$ messages entre deux accès successifs en section critique tandis que dans l'algorithme de [BAL 02] et notre algorithme, un nœud qui sort de sa section critique envoie le jeton à son voisin ce qui nécessite un seul message.

5.5.3. Nombre de saut par section critique (Hop)

Dans le cas d'une topologie dynamique, la destruction d'un lien (i, j) dans l'algorithme de [WAL 98] fait que le nœud i renvoie sa requête ce qui fait que les message d'exclusion mutuelle traverse 5 sauts par entrée dans la section critique. Tandis que dans l'algorithme de [BAL 02] et notre algorithme nécessite $O(D+1)$ sauts.

5.5.4. Complexité de routage (R)

Dans le cas d'une topologie dynamique, la destruction d'un lien (i, j) dans l'algorithme de [WAL 98] fait que le nœud i envoie sa nouvelle taille à tous ses voisins ce qui représente une complexité de $O(\Delta)$, tandis que l'algorithme de [BAL 02] utilise un protocole de routage proactif où chaque nœud envoie d'une manière périodique sa table de routage à tous les autre nœuds du réseau ce qui représente une complexité de $O(N^2)$. Notre algorithme proposé dans le chapitre 4 utilise un protocole de routage de type source routing, une destruction d'une route utilisé par le message jeton nécessite l'exécution d'une procédure de recherche pour localiser le prochain nœud privilégié, ce qui représente $O(N+D)$ messages de routage supplémentaire.

Le tableau suivant résume les résultats présentés ci dessus :

		WAL 98	BAL 02	Notre algorithme
M	Charge Faible	*O(3D)*	*O(n+1)*	*2*
	Charge élevée	5	2	
T		*O(D)*	*1*	*1*
Hop		*5*	*O(D+1)*	*O(D+1)*
R		*O(Δ)*	*O(N^2)*	*O(2[N+D])*

Figure 5.2

5.6. Complexité de contrôle

L'algorithme de et [WAL 98] ne transmet aucune information supplémentaire avec les messages ce qui fait sa complexité de *O(*1*)* bits. L'algorithme de [BAL 02] transmet avec le message jeton un tableau *T* de taille n et chaque entrée *T[i]* montre si un nœud i à déjà reçu le jeton pendant le tour courrant., ce qui représente une taille de *O(n)* bits.

Dans notre algorithme, le message jeton contient l'identité de tous les nœuds qui ont demandé l'accès en section critique, si m est le nombre de nœuds qui ont fait leur requête alors la taille du jeton est de *O(m* log_2 *n)* bits.

5.7. Conclusion

D'après notre étude sur les métriques de performance, nous remarquons que l'algorithme de [WAL 98] ne donne pas de bonnes performances, car chaque fois qu'un nœud i qui a envoyé une requête à j trouve que le lien (i, j) est rompu, renvoie la requête. L'algorithme de [BAL 02] donne de bons résultats dans le cas d'une charge élevée. Notre algorithme nécessite seulement l'envoi de 2 messages d'application.

L'algorithme de [BAL 02] et notre algorithme donnent un délai de synchronisation meilleure que celui donné par [WAL 98], tandis que pour les nombre de sauts traversés par section critique et les messages de routage, l'algorithme de [WAL 98] donne de meilleures résultats que ceux de [BAL 02] et de notre algorithme.

L'étude de complexité ne permet pas de faire une bonne comparaison entre les algorithmes car chaque algorithme utilise une architecture différente des autres et sa propre interprétation des métriques. Dans l'algorithme [WAL 98] : le nombre de messages par une entrée dans la section critique est le nombre de messages générés par la couche réseau (chaque saut représente une génération d'un message) alors que dans l'algorithme de [BAL 02] et notre algorithme, ce nombre représente le nombre de messages générés par la couche application (ce nombre ne tient pas compte du nombre de sauts traversés).
Donc, pour faire une bonne comparaison entre les algorithmes, Nous passons à la simulation.

Chapitre 6 Simulation des algorithmes d'exclusion mutuelle

6.1. Introduction

Le simulateur que nous allons utiliser pour évaluer les algorithmes d'exclusion mutuelle est *GloMoSim* (*Global Mobile Simulator*) [ZEN98], que nous avons étendu afin de faire nos propres mesures. Il utilise le principe de simulation par évènements discrets fournis par le langage *Parsec*[MEY 99].

6.2. Le Langage Parsec

6.2.1. Définition

PARSEC (*Parallel Simulation Environment for Complex system*) est un langage dérivé d'un autre langage appelé *MAISIE*, basé sur le langage C pour la simulation des évènements discrètes. Il est développé à l'université UCLA (*University Califorinia Los Angeles*).

Un de ses avantages les plus importants est sa possibilité d'exécuter un modèle de simulation à événement discret par plusieurs protocoles de simulation asynchrone parallèle dans différentes architectures parallèle, il sépare entre le modèle de simulation et les algorithmes (séquentielle ou parallèle) utilisés pour l'exécution du modèle.

Les objets dans un système physique (processus physique) sont modélisés par des processus logiques nommés *entity*. Les interactions entre les processus physiques sont modélisées par l'échange de messages entre ses entités correspondantes[DER 01].

6.2.2. Les Entités

Une entité représente un objet spécifique dans le système physique, chaque programme dans PARSEC doit avoir une entité principale *driver* qui initialise l'exécution de la simulation et qui a le rôle de la fonction *main* dans le langage C.

6.2.3. Les Messages

PARSEC utilise des objets de type *message* composés de nom et une liste de paramètres. Les entités communiquent entre elles en utilisant les messages, chaque entité a un buffer unique, des primitives d'envoi et de réception asynchrone de messages sont fournies pour respectivement le dépôt et l'enlèvement des messages du buffer. Ces primitives sont *send* et *receive*.

6.2.3 Les Evénements

Chaque événement dans le modèle de simulation à événement discret simule certaine activités dans le système physique qui entraîne d'autres objets, chaque événement est associé à un estampille qui indique le temps auquel l'événement correspond apparaît dans le système.

Le temps de la simulation peut seulement avancer quand l'entité reçoit un message ou quand elle exécute l'instruction *hold*.

6.3. Le simulateur GloMoSim

6.3.1. Introduction

GloMoSim est un environnement de simulation à grande échelle pour les réseaux sans fil et filaires. Il a été conçu en utilisant la capacité de la simulation parallèle fournie par Parsec. Il modélise des réseaux semblables à l'architecture à sept couches de la norme OSI.

Pour des raisons de performance, GlomoSim utilise une seule entité Parsec, cette dernière représente une région géographique de simulation où elle englobe plusieurs nœuds mobiles. Chaque nœud mobile utilise la pile des protocoles présentée dans la figure 6.1.

La couche Application
La couche Transport
La couche réseau (protocoles de routage)
La couche Liaison de données (MAC)
La couche Radio
La couche Mobilité

Tableau 6.1

GloMoSim utilise un fichier de configuration qui contient différents paramètres, ces paramètres représentent La configuration du réseau à simuler. Parmi ces paramètres on trouve :

- *SIMULATION_TIME* : Représente le temps maximum d'une simulation.
- *TERRAIN-RANGE-X* et *TERRAIN-RANGE-Y* : Représentent la longueur et la largeur du terrain physique simulé.
- *NUMBER-OF-NODES* : Représente le nombre de nœuds dans le réseau.
- *POWER-RANGE* : Représente la portée de communication de chaque nœud.
- *BANDWIDTH* : Représente la bande passante utilisée par les nœuds pour transmettre les messages.
- *MOBILITY* : Définit Le modèle de mobilité utilisé.
- *MAC-PROTOCOL*: Définit le protocole utilise au niveau de la couche MAC.
- *ROUTING-PROTOCOL* : Définit le protocole de routage utilise au niveau de la couche réseau.
- *TRANSPORT-PROTOCOL* : Définit le protocole de transport utilisé.
- *APP-CONFIG-FILE* : Définit le fichier qui précise les applications lancées par les nœuds.

6.4. Environnement de simulation

6.4.1 modèle de mobilité

Parmi les modèles qu'on trouve dans Glomosim on a choisi le modèle de mouvement *Waypoint*. Dans ce modèle un nœud sélectionne aléatoirement une destination dans le

terrain physique, il se déplace vers cette destination avec une vitesse aléatoire dans l'intervalle [*MOBILITY-WP-MIN-SPEED*, *MOBILITY-WP-MAX-SPEED*], après qu'il atteint sa destination, le nœud reste pour une période de temps égale à *MOBILITY-PAUSE*. Ce modèle est plus proche du mouvement réel des nœuds par rapport aux autres modèles.

6.4.2. Surface simulée

Nous Choisissons un terrain carré de longueur 750 m.

6.4.3. Modèle de propagation

Nous Choisissons le modèle *Free Space* (propagation libre) qui suppose que le signal se propage de l'émetteur vers le récepteur sans qu'il y ait des obstacles entre eux, et le signal s'atténue en fonction de la distance entre le nœud source et le nœud destination, ce modèle est simple car il élimine l'effet des obstacles dans la simulation.

6.4.4. Protocole de la couche MAC

Nous avons choisi comme protocole dans la couche MAC le IEEE 802.11, c'est un protocole utilisé dans les réseaux sans fil. Il a les avantages suivants :

- Il évite les collisions par la vérification du canal avant de l'utiliser.
- Chaque transmission d'un paquet est suivie par un acquittement si ce paquet est reçu par le récepteur.
- L'absence d'un acquittement après plusieurs transmissions indique la défaillance de lien entre les deux nœuds.
- Il utilise un buffer pour sauvegarder les paquets même si le système est oisif.

6.4.5. La couche application

Chaque nœud dans le réseau utilise une application qui génère des message suivant une distribution poissonnière avec une moyenne λ qui représente le nombre de requêtes par seconde générées par un nœud, alors pour un nœud donné, le temps entre sa sortie de sa section critique et l'envoi de sa prochaine requête suit une distribution exponentielle avec une moyenne de $\left(\frac{1}{\lambda}\right)$ unités de temps. Quand un nœud entre dans sa section critique, il reste pour une durée de *E* unité de temps.

6.4.6. Constantes utilisées dans la simulation

Le tableau 6.2 présente des constantes utilisées dans tous les scénarios d'exécution.

NUMBER-OF-NODES	25
SIMULATION_TIME	10 minutes
BANDWIDTH	2 Mbits/s
MOBILITY-PAUSE	1 seconde

Tableau 6.2

6.5 Les paramètres de comparaison

6.5.1 La mobilité

La mobilité est un paramètre important dans l'évaluation des réseaux ad hoc, elle cause le changement de la topologie, il existe plusieurs définitions de ce paramètre comme la vitesse des nœuds ou le *pause time* dans le modèle *Waypoint*, ces définitions n'expriment pas vraiment la mobilité car il existe des cas où les nœuds se déplacent avec une grande vitesse ou avec un pause time très faible dans une même direction sans qu'il y 'aura un changement dans la topologie du réseau, d'autre part si les nœuds se déplacent avec une vitesse faible ou un grand pause time mais s'éloignant les uns des autres on aura un changement de liens important.
L' exemple suivant illustre l'invalidité de ces définitions : soit un réseau contenant 2 nœuds liés. Quand ils se déplacent avec une grande vitesse et/ou un petit pause time mais dans la même direction, les 2 nœuds restent liés, d'autre part si les nœuds se déplacent avec une vitesse faible et/ou un grand pause time dans des directions opposées l'un par rapport à l'autre, après un certain temps le lien sera défaillant. Pour cela nous avons opté pour une autre définition de la mobilité présentée dans [LAR 98]. Cette définition est basée sur le mouvement relatif entre les nœuds ce qui exprime bien le changement de la topologie du réseau.
La mobilité est une fonction de la vitesse et du modèle de mouvement, elle est représentée par un nouveau paramètre (facteur de mobilité), il est calculé durant la simulation avec un certain taux Δt. La formule du facteur de mobilité *Mob* est :

$$Mob = \sum_{i=1}^{n} \frac{M_i}{n}$$

$$M_x = \sum_{t=0}^{T-\Delta t} \frac{\left| Ax(t) - Ax(t+\Delta t) \right|}{T}$$

$$A_x(t) = \sum_{i=1}^{n} \frac{dist(n_x, n_i)}{n-1}$$

Les valeurs de *Mob* sont obtenues en fixant le paramètre *MOBILITY-WP-MIN-SPEED* à 0 et en variant le paramètre *MOBILITY-WP-MAX-SPEED*.

Le tableau ci dessous (Tableau 6.3) donne la signification des variables utilisées dans la formule du facteur de mobilité.

$dist(n_x, n_y)$	Distance entre le nœud x et le nœud y
n	Le nombre de nœuds
$A_x(t)$	La distance moyenne entre le nœud x et tous les autres nœuds à l'instant t
M_x	la mobilité relative moyenne du nœud x par rapport à tous les autres nœuds durant le temps de simulation
T	Le temps de simulation
Δt	L'intervalle de temps utilisé dans le calcul

Tableau 6.3

Nous calculons $A_x(t)$ après chaque Δt c.-à-d. pour t=0, t= Δt, t=2 Δt,……,t=T. Pour notre simulation nous avons fixé Δt à 0.1 seconde.

Pour tester la validité de ce paramètre, nous avons ajouté le calcul du nombre moyen de liens changés durant la simulation [BAD 03]. Un changement de lien est soit la création d'un nouveau lien ou la défaillance d'un lien existant.

Nous définissons le calcul du nombre moyen de liens changés *LINK_ CHANG* par la formule suivante :

$$LINK\_\ CHANG = \sum_{t=\Delta t}^{T} link\_change(t) \times \frac{\Delta t}{T}$$

$$Link\_change(t) = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} \left[state(t,i,j) \oplus state(t-\Delta t,i,j) \right]$$

$$state(t,i,j) = \begin{cases} 1 \text{ si i est en liaison avec j à l'instant t} \\ 0 \text{ sinon} \end{cases}$$

La variation du nombre de liens changés en fonction de la mobilité est représentée dans la figure suivante :

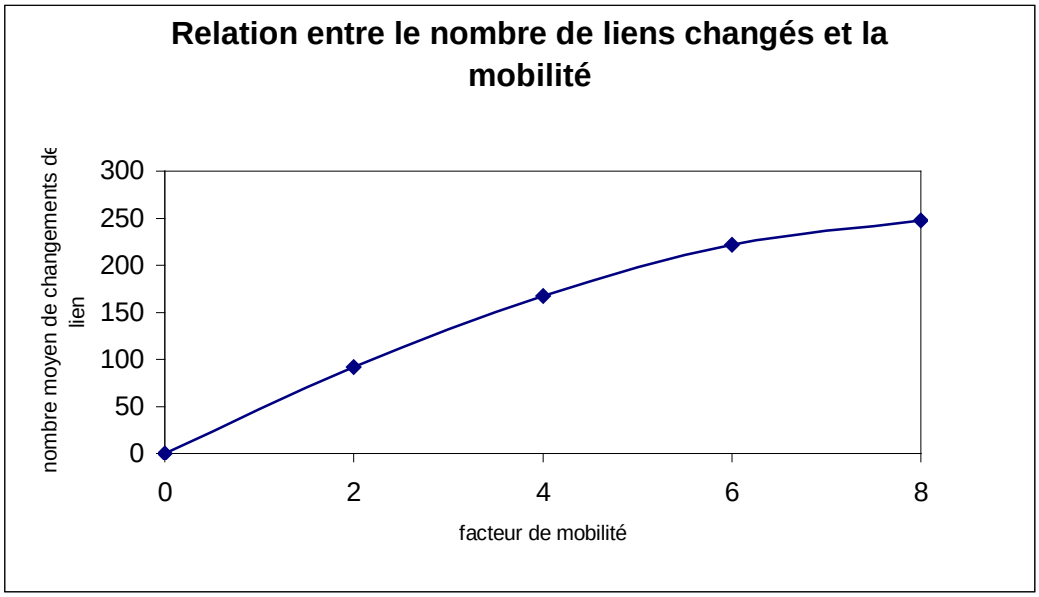


Figure 6.1

Les résultats de simulations dans ce chapitre sont donnés pour quatre types de mobilité : *Mob*=0 (Topologie statique), *Mob*=1(Mobilité faible), *Mob*=5 (Mobilité moyenne), *Mob*=10 (Mobilité élevée).

6.5.2 La charge

Ce paramètre représente le nombre de requêtes générés par seconde par un nœud. Le processus d'envoi de requêtes suit une distribution poissonnière avec une moyenne λ, ceci implique que pour un nœud donné, le temps entre sa sortie de sa section critique et la génération de sa prochaine requête suit une distribution exponentielle avec une moyenne de $\left(\frac{1}{\lambda}\right)$ unités de temps. Nous avons définit $\left(\frac{1}{\lambda}\right)$ dans le fichier de configuration de *GloMoSim* par le paramètre *INTERARRIVAL-TIME*.

L'exécution se déroule comme suit : chaque nœud dans le réseau génère une requête suivant une loi de poisson. Quand il obtient le jeton, il reste dans la section critique pour un durée de *E* unité de temps. Quand il sort de la section critique il va attendre un temps qui suit une loi exponentielle avec une moyenne de $\left(\frac{1}{\lambda}\right)$ avant de générer la prochaine requête. Ce processus se répète jusqu'à la fin de la simulation.

6.5.3. Temps de séjour dans la section critique

Ce paramètre représente le temps *E* que le nœud va prendre pour rester dans la section critique. Le temps *E* est défini dans le fichier de configuration de *GloMoSim* par le paramètre *CS-EXECUTION-TIME*.
Dans toutes les simulations, nous fixons *CS-EXECUTION-TIME* à 1 seconde.

6.6. Les métriques mesurées

Pour comparer les protocoles d'exclusion mutuelle, nous choisissons de mesurer cinq métriques qui sont :

6.6.1. Le nombre d'accès en section critique

Cette métrique permet de connaître combien de fois l'algorithme d'exclusion mutuelle peut entrer dans la section critique durant le temps de simulation. Elle permet aussi de connaître le taux de service du système. Formellement ce taux est défini par la formule $\left(\frac{1}{T+E}\right)$ tels que *T* et *E* sont respectivement le délai de synchronisation et le temps de séjour dans la section critique. Le nombre d'accès en section critique pendant une durée de *SIMULATION_TIME* est défini par la variable *TotalCSAccessNumber*

$$TotalCSAccessNumber = \left(\frac{SIMULATION\_TIME}{T+E}\right)$$

L'objectif d'un algorithme d'exclusion mutuelle est de réduire *T* autant que possible. Nous constatons que quand $T \to 0$, $TotalCSAccessNumber \to \left(\frac{SIMULATION\_TIME}{E}\right)$.
Dans notre simulation cette valeur égale à 600 accès.

6.6.2. Le nombre moyen de messages par entrée dans la section critique

Le nombre de messages (requêtes, jetons) échangés par un accès en section critique, il est donné par la formule suivante :

$$AvgMsgPerCS = \frac{TotalCSMsg}{TotalCSAccessNumber}$$

Tel que : *TotalCSMsg* : est le nombre de messages (requêtes, jetons) envoyés par tous les nœuds du réseau.

Cette métrique permet de mesurer bien la variable M définie dans le chapitre 5. Un algorithme d'exclusion mutuelle sera considéré optimal que les autre si sa variable *AvgMsgPerCS* est inférieure à celles des autres algorithmes.

6.6.3. Le nombre moyen de sauts par section critique

Les algorithmes d'exclusion mutuelle de BV [BAL 02] et notre algorithme présenté dans le chapitre 4 s'exécutent au niveau de la couche application. Ils n'incluent pas le nombre de sauts dans le calcul de la métrique *AvgMsgPerCS*, tandis que l'algorithme RL [WAL 98] qui s'exécute au niveau de la couche réseau prend en compte le nombre de sauts. Ceci implique que les résultats obtenus en calculant la métrique *AvgMsgPerCS* ne reflètent pas bien les performances de chaque algorithme. Ceci implique aussi que le nombre de sauts par entrée dans la section critique (*AvgHopPerCS*) est plus convivial pour faire la comparaison entre les algorithme d'exclusion mutuelle Cette métrique est définie par la formule suivante :

$$AvgHopPerCS = \frac{TotalCSHop}{TotalCSAccessNumber}$$

Tel que : *TotalCSHop* est le nombre de sauts traversés par les messages requêtes et jetons échangés entre les nœuds du réseau.

6.6.4. Le nombre de messages de routage

Cette métrique permet d'observer l'impact d'un algorithme d'exclusion mutuelle sur le protocole de routage. Elle calcule tous les paquets de routage générés pendant l'exécution d'un algorithme d'exclusion mutuelle.

Pour l'algorithme RL qui s'exécute au niveau de la couche réseau, il génère les messages de routage pour mettre à jour la taille de nœuds quand un il perd les dernier lien sortant vers le nœud privilégié ou quand le jeton circule dans le réseau.

Pour l'algorithme BV, il s'exécute au dessus d'une couche réseau qui utilise un protocole de routage proactif, Ce dernier est basé sur l'algorithme de *Bellemand-Ford*

[BER 92] pour trouver le plus proche chemin entre un nœud source et un nœud destinataire. Il génère des tables de routage d'une façon périodique et les envoie à tous les nœuds du réseau.

Pour notre algorithme, il s'exécute au-dessus d'une couche réseau qui supporte deux types de protocole de routage, l'un c'est TORA [PAR 97] pour router les requêtes et l'autre c'est un protocole de routage réactif de type source routing pour router le jeton, il est similaire à DSR [JOH 96] où il emploi la procédure de recherche (génération des messages *Find* et *Reply*) pour trouver le prochain nœud privilégié.

6.6.5. Le temps moyen d'attente pour un accès en section critique

Cette métrique permet de connaître combien de temps il faut attendre pour entrer dans la section critique. Elle est définie par la formule suivante :

$$AvgTimePerCS = \frac{TotalCSWaitTime}{TotalCSAccessNumber}$$

Tel que *TotalCSWaitTime* : est le temps total d'attente pour tous les accès en section critique.

Cette métrique est en fonction du nombre de nœuds en attente et le délai de synchronisation.

6.7. Les résultats de simulation

Chaque point dans les figures suivantes représente la moyenne de cinq exécutions.

Nous faisons la comparaison entre trois algorithmes d'exclusion mutuelle :

1- L'algorithme Reverse Link (RL) [WAL 98]
2- L'algorithme de Baldoni (BV) [BAL 02]
3- Notre algorithme présenté dans le chapitre 4

6.7.1. Le nombre d'accès en section critique

Dans la figure 6.2, nous remarquons que quand la charge augmente, le nombre d'accès en section critique augmente et quand la mobilité augmente ce nombre diminue. l'algorithme RL donne de meilleure résultats que les autres algorithmes. Dans le cas de notre algorithme, il exécute une procédure de recherche pour trouver le prochain nœud privilégié ce qui augmente le délai de synchronisation et diminue le nombre d'accès en section critique. Pour l'algorithme BV, le jeton circule entre tous les nœuds du réseau même entre les nœud qui ne désire pas entrer en section critique, ce qui augmente le temps d'attente et diminue le nombre d'accès en section critique.

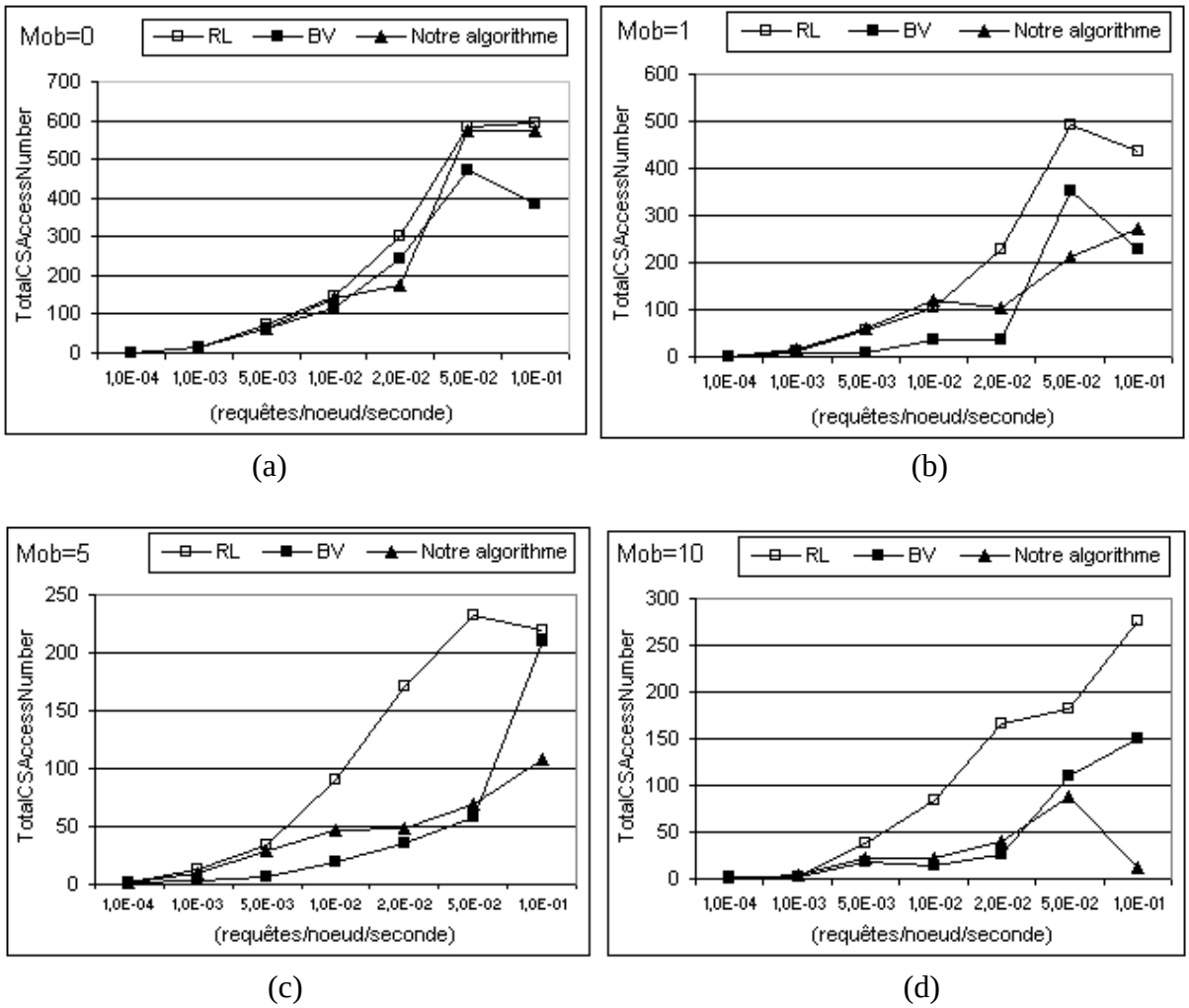


Figure 6.2

6.7.2. Le nombre moyen de messages par entrée dans la section critique

Dans la figure 6.3, nous remarquons que le nombre moyen de messages par entrée dans la section critique dans le cas de notre algorithme est meilleure que les autres algorithmes. Il n'est pas influencé par la variation de mobilité sauf dans le cas où le chemin obtenu après l'exécution de la procédure de recherche est rompu. Dans ce cas, l'algorithme doit renvoyer le jeton.

Dans le cas de l'algorithme BV, le nombre moyen de messages par entrée dans la section critique diminue quand la charge augmente. Dans le cas d'une charge faible(10^{-4}), l'algorithme BV envoie le jeton à tous le nœuds du réseau, tandis que dans le cas d'une charge élevé (10^{-1}), le nombre d'accès en section critique augmente (chaque nœud qui reçoit le jeton entre dans la section critique), ce qui fait que le nombre moyen de messages par entrée dans la section critique diminue.
Les résultat de la Figure 6.3(b) (mobilité faible), sont beaucoup proches de ceux de la figure 6.3(a), ils ne sont pas influencé par les changements de topologie.

Dans les figures 6.3(c) (mobilité moyenne) et 6.3(d)(mobilité élevée), le probabilité des partitionements augmentent, ce qui fait que le nœud privilégié doit faire après chaque période de temps une tentatives pour retransmettre le jeton aux nœuds qui n'ont pas encore le reçu.

Dans le cas de l'algorithme RL, les résultats des figures 6.3(a) (topologie statique) et 6.3(b) (mobilité faible) sont presque similaires. Dans le cas d'une mobilité moyenne(Figure 6.3(c)) et une mobilité élevée(Figure 6.3(d)), le nombre de messages par entrée dans la section critique augmente avec l'augmentation de la mobilité parce que les probabilités de partitionement augmentent et la structure logique utilisée par l'algorithme (DAG) ne peut pas détecter les partitionements, ce qui fait que les requêtes circule continuellement dans le réseau, en plus de ça, chaque nœud i dont sa file de requête n'est pas vide et perd la connexion avec son voisin désigné par $Next_i$ doit retransmettre la requête sur un autre chemin.

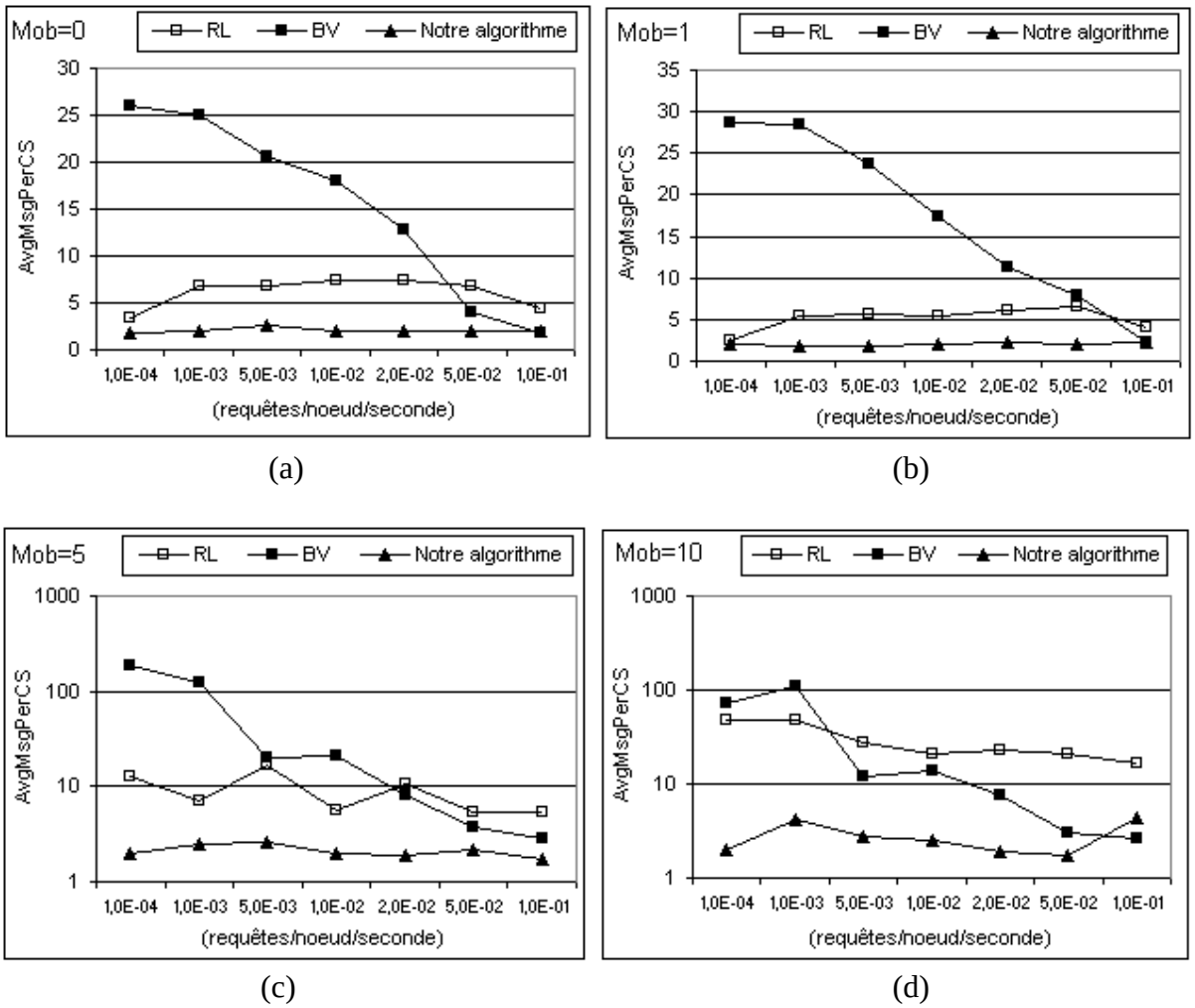


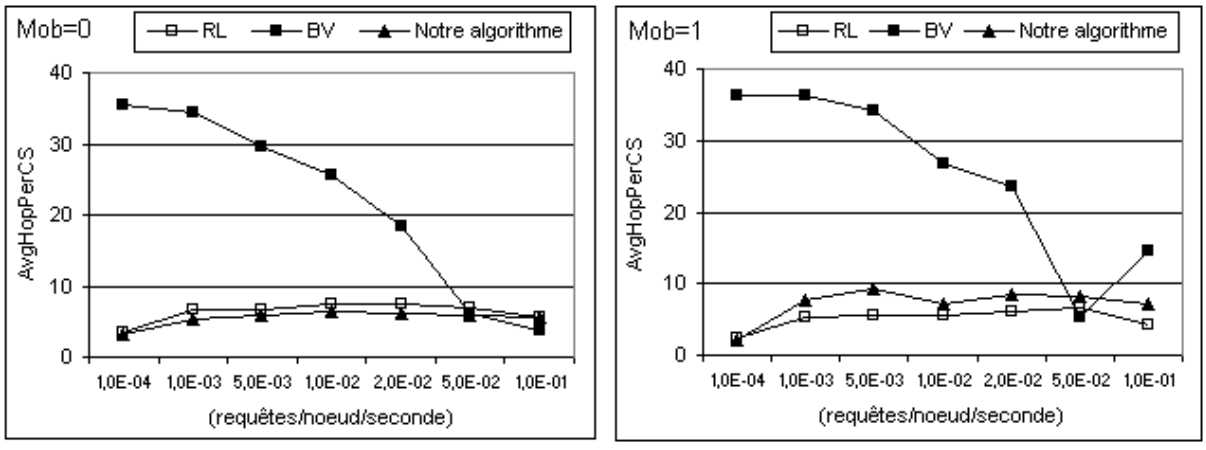
Figure 6.3

6.7.3. Le nombre moyen de sauts par section critique

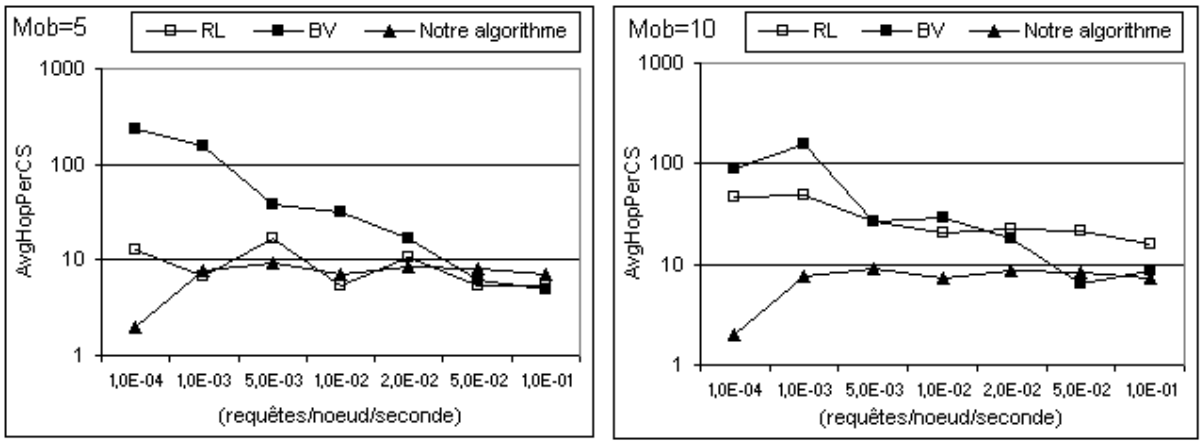
Dans la figure 6.4, nous remarquons que notre algorithme donne de bonnes résultats concernant le nombre moyen de sauts par section critique. L'algorithme emploi une politique qui choisit parmi les candidats de l'ensemble Bag_k, le nœud le plus proche en terme de nombre de sauts. Notre algorithme n'est pas beaucoup influencé par les changements de la topologie parce que la structure logique employé par le protocole de routage TORA permet de détecter le partitionement du réseau, ce qui fait que la requête ne va pas circuler continuellement dans le réseau, en plus de ça il n'envoie le jeton qu'après l'établissement d'un chemin entre le nœud privilégié courant et le prochain nœud privilégié. Quand ce chemin est rompu, le nœud intermédiaire qui possède le jeton exécute une procédure de recherche pour localiser le prochain nœud privilégié avant d'envoyer le jeton.

Dans le cas de l'algorithme RL, le nombre moyen de sauts par section critique est le même que la métrique calculée dans le paragraphe 5.2. Avec l'augmentation de la mobilité le nombre de retransmission des requêtes augmente et les requêtes circule continuellement dans le réseau dans la partition qui ne contient pas le jeton.

Dans le cas de l'algorithme BV, le nombre de sauts par entrée dans la section critique diminue avec l'augmentation de la charge. Quand la charge est faible, la circulation du jeton entre tous les nœuds du réseau permet à un nombre faible de nœuds d'entrer dans la section critique, alors que dans le cas d'une charge élevée un nombre important de nœuds entre dans la section critique.
Avec l'augmentation de la mobilité , le nombre de sauts par entrée dans la section critique augmente, ceci est du aux informations fournies par le protocole de routage qui seront de plus en plus imprécis et le prochain nœud privilégie choisi n'est pas forcément le plus proche, donc le protocole de routage nécessite un certain temps avant de se converger vers un état stable et cette convergence est difficile à réaliser dans des situations où les changements de la topologie sont continues.



(a) (b)



(c) (d)

Figure 6.4

6.7.4. Le nombre de messages de routage

Dans la figure 6.5, nous remarquons que le nombre de messages générés par le protocole de routage lié à l'algorithme BV est constant, il n'est pas influencé par la variation de la charge. Ceci est du au type de l'algorithme de routage qui est proactif, il génère les messages de routage (table de routage) d'une façon périodique mais ce nombre augmente quand la mobilité augmente.

Dans le cas de notre algorithme, le nombre de messages de routage augmente avec l'augmentation de la charge, parce que chaque fois que le nœud privilégié veut trouver le nœud qui va recevoir le jeton, il exécute une procédure de recherche qui consiste à diffuser dans le réseau un message *Find* et attendre la réception d'un message *Reply*. L'exécution de cette procédure consomme une grande quantité de messages mais ces messages ont une faible taille par rapport à ceux générés par l'algorithme BV.

Dans le cas de l'algorithme RL, le nombre de messages de routage augmente avec l'augmentation de la charge, parce que chaque fois qu'un nœud reçoit le jeton il envoie à tous ses voisins un message indiquant sa nouvelle taille dans le DAG.

Nous remarquons aussi que le nombre de messages de routage pour l'algorithme RL et notre algorithme ne sont pas influencés par la variation de la mobilité mais par le nombre d'accès en section critique et de la variation de la charge.

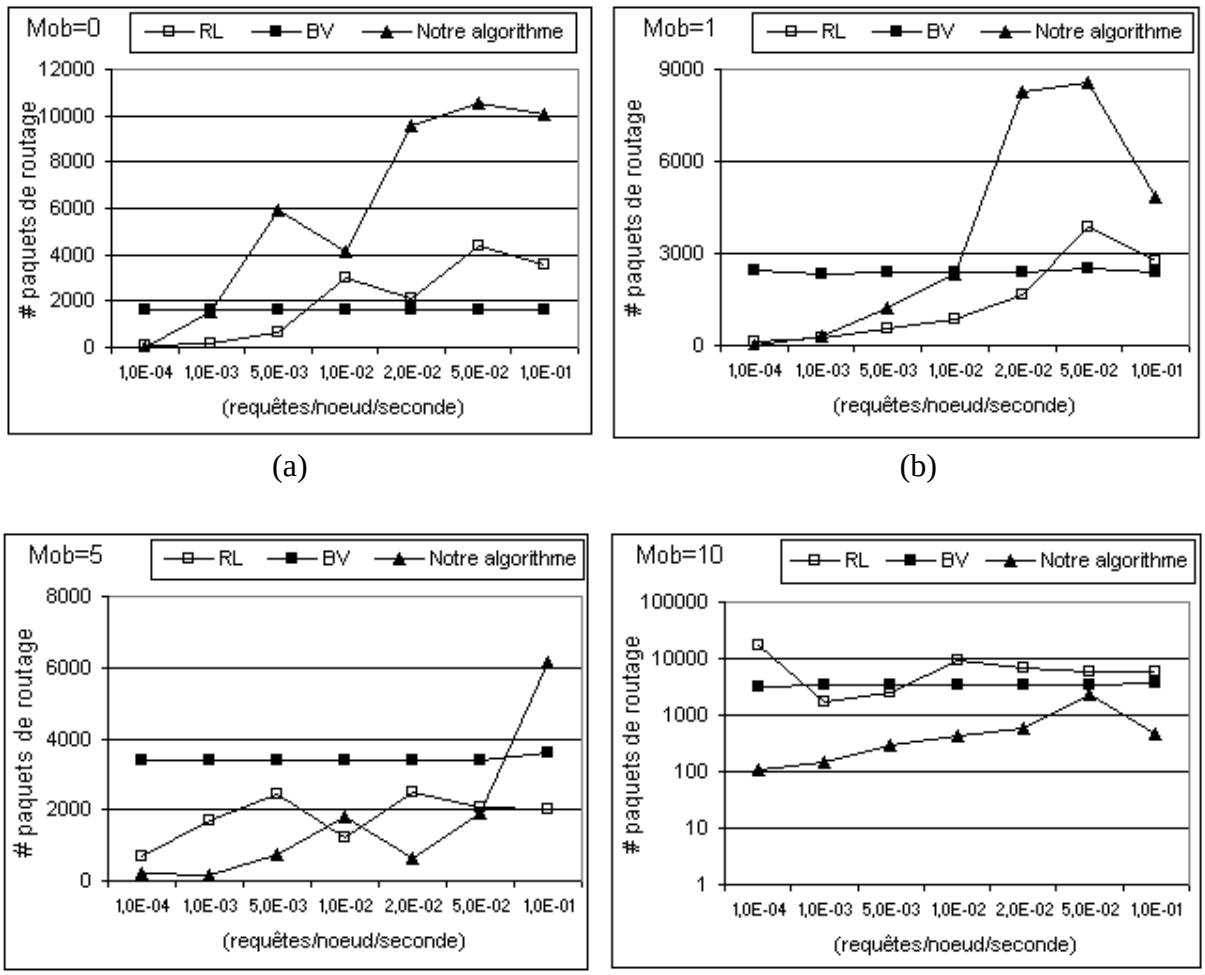


Figure 6.5

6.7.5. Le temps moyen d'attente pour un accès en section critique

Dans la figure 6.6, nous remarquons que le temps moyen d'attente par un accès en section critique augmente avec l'augmentation de la charge, ceci est du à l'augmentation des nœuds qui sont en attente pour accéder en section critique.

Dans le cas d'une topologie statique (Figure 6.6(a)), nous remarquons que le temps moyen d'attente dans notre algorithme est plus grand de celui des autres protocoles dans le cas statique, parce que quand un nœud sort de sa section critique et ne trouve pas un chemin vers le prochain nœud privilégié exécute une procédure de recherche pour trouver ce nœud, l'exécution de cette procédure nécessite un certain temps.
Dans le cas d'une topologie dynamique (Figures 6.6(b,c,d)), nous remarquons aussi que le temps moyen d'attente n'est pas influencé par la variation de la mobilité mais par le nombre d'accès en section critique.

Le temps moyen d'attente de l'algorithme RL est plus grand que les autres algorithmes quand la topologie devient dynamique (Figures 6.6(b,c,d)), parce que dans ce cas le

nombre de requêtes supprimées dans les files des nœuds intermédiaire augmente, ce qui fait que RL renvoie ces requêtes et d'ici le temps moyen d'attente augmente.

Dans le cas de l'algorithme BV, le temps moyen d'attente dépend du nombre d'accès en section critique.

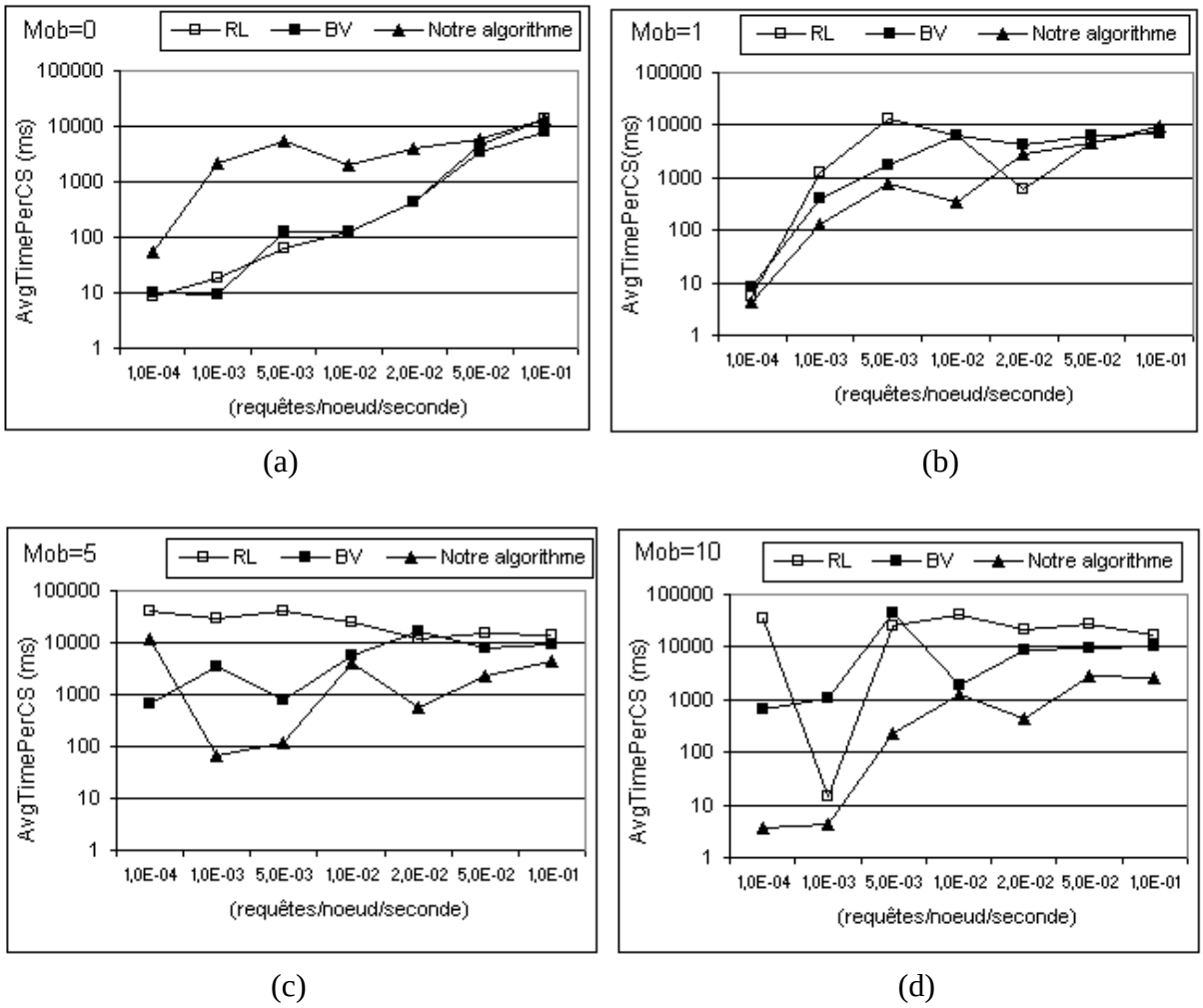


Figure 6.6

6.8. Conclusion

A partir des résultats de simulation, nous pouvons conclure que notre algorithme donne de bons résultats dans le cas du nombre moyen de messages et le nombre moyen de sauts par entrée dans la section critique mais Il génère un grand nombre de messages de contrôle et son nombre d'accès en section critique est faible dans le cas d'une mobilité élevée et moyenne. En ce qui concerne le temps moyen pour un accès en section critique dépend plus de la charge du réseau plus que la mobilité.

L'algorithme BV génère un nombre faible de messages par section critique dans le cas d'une charge élevée et son protocole de routage génère un nombre de messages de routage constant.

L’algorithme RL donne des résultats moyennes par rapports aux autres protocoles sauf pour le temps moyen pour un accès en section critique où il est le plus grand dans le cas d’une topologie dynamique.

Conclusion générale

Les réseaux mobiles sans fil peuvent être classés en deux catégories, les réseaux mobiles avec infrastructure comme les réseaux cellulaires et les réseaux mobiles sans infrastructure ou réseaux Ad Hoc. Ils se caractérisent par le changement rapide et fréquent de la topologie d'une manière imprévisible et par les ressources modestes des unités mobiles.

Le problème d'exclusion mutuelle dans les réseaux Ad Hoc est un problème complexe, car les protocoles d'exclusion mutuelle dans un tel environnement doivent tenir compte de la variation de la topologie, le partitionement éventuel du réseau, la limitation des sources d'énergie dans les unités mobiles.
Ces contraintes font que les protocoles d'exclusion mutuelle doivent générer une charge faible en termes de nombre de messages et de quantité d'énergie.

L'algorithme Reverse Link(RL) et l'algorithme de Baldoni donnent de bon résultats pour certains cas et moyens dans d'autres. Ils introduisent les nœuds qui n'ont pas demandé l'accès en section critique dans leur exécution.

L'algorithme proposé dans ce document s'exécute seulement entre les nœuds demandant l'accès à la section critique. Il n'utilise pas une structure logique supplémentaire mais il exploite celle de la couche réseau. Cette dernière a la propriété de supporter plusieurs protocoles de routage.
Notre algorithme utilise deux protocole de routage, chacun d'eux est utilisé pour router un type de message donné. Ce choix permet d'envoyer ces messages sur des routes fiables et optimales en terme de nombre de sauts. Il échange un nombre optimal de messages par entrée dans la section critique.

Toutefois, cet algorithme présente quelques inconvénients. Par exemple, son architecture suppose l'existante d'une couche réseau qui supporte différents protocoles de routage, ce qui nécessite des espaces de stockage supplémentaires. En plus de ça, la localisation du prochain nœud privilégié consomme beaucoup de messages de routage. Cet algorithme donne aussi une solution tolérante aux pannes où il traite le cas des pannes des unités mobiles et la perte du jeton.

L'architecture proposée peut être un modèle pour plusieurs algorithme distribué dans le réseau Ad Hoc, dans le quel il peut y' avoir plusieurs protocole de service au niveau de la couche application et chaque service utilise une ou plusieurs techniques de routage.

Une des perspectives de ce travail est d'étudier l'exclusion mutuelle dans des environnements où les unités mobiles peuvent tomber en panne et les liens de communication sont non fiables, l'autre perspective est de proposer un algorithme distribué d'allocation de ressources où il peut y avoir différents types de ressources et chaque type de ressource est disponible en plusieurs exemplaires.

Bibliographie

[BER 92] D. Bertsekas and R. Gallager, "*Data Networks*". Prentice Hall Inc, pp 297-333, 1992.

[BAD 94] B. R. Badrinath, A. Acharya, and T. Imielinski, "Structuring Distributed Algorithms for Mobile Hosts", *In the 14th International Conference on Distributed Computing Systems*, June 1994, pp 21-28.

[BAD 98] N. Badache, "La mobilité dans les systèmes répartis", *Technique et science informatiques*, Volume 17-n° 8, 1998, pp 969-997.

[BAD 02] N. Badache, A. Derhab, D. Djamel, "Analyse comparative des algorithmes de routage dans les réseaux mobiles Ad Hoc", LSI-TR-07-02, USTHB, Février 2002.

[BAD 03] N. Badache, D. Djenouri, A. Derhab, "Mobility impact on Mobile Ad Hoc Routing Protocols", *ACS/IEEE International Conference on Computer System and Applications (AICCSA'03)*, Tunis, Tunisia 14-18 July 2003.

[BAL 01] R. Baldoni, A. Virgillito. "A Token Based Mutual Exclusion Algorithm for Mobile Ad-Hoc Network ", Technical Report 28-01, Dipartimento di Informatica e Sistemistica, Universita di Roma "La Sapienza", available at http://www.dis.uniromal.it/~virgi, 2001.

[BAL 02] R. Baldoni, A. Virgillito, R. Petrassi, "A Distributed Mutual Exclusion Algorithm for Mobile Ad-Hoc Networks", *In Proc of the 7th IEEE Symposium on Computer and Communications (ISCC 20002)*, pp 539-545, Toarmun, Italy 1-4 July 2002.

[CHE 98] Tsu-Wei Chen and Mario Gerla. "Global State Routing: A new routing scheme for ad-hoc wireless networks". *Proceedings of the IEEE International Conference on Communications (ICC)*, Atlanta, GA, June 1998, pp. 171-175.

[CHI 97] C.C. Chiang, H-K Wu, Winston Liu, and Mario Gerla, "Routing in clustered multihop, mobile wireless networks", *The IEEE Singapore International Conference on Networks*, pp 197-211, 1997.

[DEF 01] X. Défago, "Distributed Computing on the Move: From mobile computing to cooperative robotics and nanorobotics", *Proceeding of the first annual Workshop on Principles of Mobile Computing (POMC*

2001), Newport, Rhode Island USA, August 2001.

[DER 01] A. Derhab, D. Djenouri,"*Simulation des algorithmes de routage dans les réseaux mobiles Ad Hoc*", Projet de fin d'étude, USTHB, Juin 2001.

[DUB 97] R. Dube et al. "Signal stability based adaptive routing for ad hoc mobile networks", *IEEE Pers. Comm*., pp 36-45, February 1997.

[DUC 92] D. Duchamp and N. F. Reynolds, "Measured performance of wireless LAN", Technical Report, Computer Science Department, Columbia University, NY, United States, September 1992.

[FRO 00] M. Frodigh, P. Johansson and P. Larsson, "Wireless ad hoc networking, The art of networking without a network", *Ericsson Review*, No. 4, 2000, pp 248-263.

[HAR 01] J. Harmas and K. Wu, "Qos Support in Mobile Ad Hoc Networks", *Crossing Boundaries – an interdisciplinary Journal*, VOL 1, No 1, Fall 2001.

[HAT 99] K. P. Hatzis, G. P. Pentaris, P. G. Spirakis, V. T. Tampakas and R. B. Tan, "Fundamental Control Algorithms in Mobile Networks", *Proc ACM Symposium on Parallel Algorithms and Architectures*, pp 251-260, 1999.

[HOU 01] A. Housni and M. Trehel, "A New Distributed Mutual Exclusion Algorithm for Two Groups", *In SAC 2001*, Las Vegas, NV, February 2001.

[IMI 94] Imienlinski T. and Badrinath B. R, "Mobile wireless computing: solutions and challenges in data management". *CACM*, 37(10), pp 18-28, October 1994.

[IWA 99] Iwata, C. C. Chiang, G. Pei, M. Gerla, and T.-W. Chen. "Scalable routing strategies for ad hoc wireless networks". *IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks*, pp.1369-1379, August 1999.

[JIA 99] Mingliang Jiang, Jinyang Li, Y.C. Tay. "Cluster based routing protocol", IETF Draft, August 1999.
http://www.ietf.org/internet- drafts/draft-ietf-manet-cbrp-spec-01.tx t.

[JOH 96] David.B Jhonson, David.A Maltz "*Dynamic Source Routing in Ad Hoc wireless*", in Mobile Computing, edited by T. Imielinski and H. Korth, chapter 5, pp. 153-181, Kluwer Academic Publishers, 1996.

[KAR 99] G. Karumanchi, S. Muralidharan and R. Prakash. “Information Dissemination in Partitionable Mobile Ad Hoc Networks”, Department of computer science, University of Texas at Dallas, 1999.

[KRI 96] P. Krishna. “*Performances issues in mobile wireless networks*”. In partial fulfillment of the requirement for the degree of Doctor of Philosophy, Texas A & U University, 1996.

[LAM 78] L. Lamport. “Time, clocks and the ordering events in a distributed system” *Communication of the ACM*, 21(7): 558-565, July 1978.

[LAR 98] Tony Larsson, Nicklas Hedman. “*Routing Protocols in wireless Ad hoc networks –A simulation study*”, Master’s Thesis, Lulea University of Technology Stockholm , 1998.

[MAE 85] M. Maekawa. “A $\sqrt{n}$ algorithm for mutual exclusion decentralized systems”, *ACM Transactions on Computer Systems*, 3(2): 145-159, May 1985.

[MACa 98] J. Macker and M. S. Corson. “Mobile Ad Hoc Networking and the IETF”, Mobile Computing and Communication Review, Volume 2, Number 1, pp 9-14, 1998.

[MACb 98] J. Macker and M. S. Corson. “RFC 2501: Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations”, Internet Draft, draft-ietf-manet-issues-01.txt, March 1998. Work in progress.

[MAL 00] N. Malpani, J. L. Welch, and N. H. Vaidya, “Leader Election Algorithms for Mobile Ad Hoc Networks”, *Proc. Fourth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pp. 96-103, 2000.

[MAL 01] N. Malpani, N. Vaidya and J. Welch “Distributed Token Circulation on Mobile Ad Hoc Networks”, *In Proc 9th International Conference on Network Protocols (ICNP)* , November 2001

[MEY 99] Richard A. Meyer “PARSEC User Manual Release 1.1”, UCLA Parallel Computing Laboratory, http://pcl.ce.ucla.edu/ , January 1999.

[MUR 96] S. Murthy and J.J. Garcia-Luna-Aceves. “An efficient routing protocol for wireless networks”. *ACM Mobile Networks and Application Journal, Special Issue on routing in mobile Communication Networks*, pp183-197, October 1996.

[PAR 97] V. D. Park and M. S. Corson. “A highly adaptive distributed routing

algorithm for mobile wireless networks". *Proceeding INFOCOM '97*, pp 1405-1413, April 1997.

[PEI 00] Guangyu Pei, Mario Gerla, Tsu-Wei Chen. "Fisheye State Routing in Mobile Ad Hoc Networks", *Proceedings of the IEEE International Conference on Communications (ICC)*, New Orleans, LA, June 2000, pp. 70-74.

[PER 94] Charles. E. Perkins and Pravin Bhagwat. "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computer", *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pp. 234-244, 1994.

[PER 99] Charles. E.Perkins, Elizabeth M.Royer. "Ad hoc on demand distance vector (AODV) algorithm". *In Systems and Applications (WMCSA'99)*, page 90- 100, 1999.

[PER 00] Charles. E.Perkins. "*Ad Hoc Networking*", Addison Wesly, ISBN 0-201-30976-9, 2000.

[PIT 93] Pitoura E. and Bhargava B. "Dealing with mobility". Issue and Research Challenges. Technical Report CSD-TR-93-070, Department of computer science, Purdue University, November 1993.

[RAY 89] K. Raymond. "A tree Based algorithm for distributed mutual exclusion". *ACM Transactions on Computer Systems*, 7(1): 61-77, February 1989.

[RIC 81] G. Ricart and A.K. Agrawala. "An optimal algorithm for mutual exclusion in computer network", *Communication of the ACM*, 24(1): 9-17, January 1981.

[ROY 99] E.M. Royer, C-K Toh. "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks", *IEEE Personal Communications*, pages 46-55, April 1999.

[SIN 89] M. Singhal. "A heuristically-aided algorithm for mutual exclusion in distributed systems", *IEEE Transactions on Computer Systems*, 38(5): 651-662, 1989.

[SIN 97] Mukesh Singhal, D. Manivannan. "A Distributed Mutual Exclusion Algorithm for Mobile Computing Environments", *Proc of the IASTED Intel Conf on Intelligent Information*, pp 557-561, 1997

[SUZ 85] I. Suzuki and T. Kasami. "A distributed mutual exclusion algorithm", *Transactions on Computer Systems*, 3(4): 344-349, November 1985.

[TOH 96] Chai-Keong Toh. “A novel distributed routing protocol to support ad hoc mobile computing”, *Proceeding 1996 IEEE 15th Annual Int'l. Phoenix Conf. Comp. And Commun*, pp 480-86, March 1996.

[TRI 98] Nishith D. Tripathi, Nortel. “Handoff in Cellular Systems”, *IEEE Personal Communication*, pp 26-37, December 1998.

[WAL 97] J. Walter, S. Kini. “Mutual Exclusion on Multihop, Mobile Wireless networks”, Technical Report 97-014, Texas A&M University, 1997.

[WAL 98] J. Walter, J. Welch and N. Vaidya. “A Mutual Exclusion Algorithm for Ad Hoc Mobile Networks”, *In Dial M for Mobility workshop*, Dallas, TX, USA, October 1998.

[WALa 01] J. Walter, G. Cao and M. Mohanty. “A K-Mutual Exclusion Algorithm for Wireless Ad Hoc Networks”, *Proceeding of the first annual Workshop on Principles of Mobile Computing (POMC 2001)*, Newport, Rhode Island USA, August 2001.

[WALb 01] J. Walter, G. Cao and M. Mohanty. “A Token Forwarding K-Mutual Exclusion Algorithm for Wireless Ad Hoc Networks”, Technical Report, Texas A&M University, 2001.

[ZEN 98] X. Zeng, R. Bagrodia and M. Gerla. “GloMoSim: A library for the parallel simulation of large-scale wireless networks”, *Proceeding of the 12th Workshop on Parallel and Distributed Simulations, PADS'98*, May 1998.

[ZHO 99] L. Zhou and Z. Haas. “Securing Ad Hoc Networks”, *IEEE Network*, pp. 24-30 November/December 1999.