PSAR

Projet Systèmes et Applications Répartis

Algorithme d'exclusion mutuelle pour les réseaux dynamiques

Master Informatique - Spécialité Systèmes Applications Répartis

Étudiants:

Radia Amina Achaibou, Lotfi Gadouche, Samy Djama

May 23, 2021

Table des matières

1 Introduction

Le problème de l'exclusion mutuelle (EM) est un problème fondamental dans les systèmes répartis qui consiste à garantir, à un moment donné, une ressource partagée logiquement ou physiquement est accessible par au plus un processus. Toute solution au problème d'EM doit garantir deux conditions essentielles: la sûreté impliquant l'utilisation de la ressource par au plus un processus. La vivacité, chaque processus pourra accéder à la ressource partagée autant de fois qu'il le souhaite.
Le problème d'EM a été largement traité dans les réseaux statiques répartis. Ainsi, plusieurs algorithmes ont été proposés. Selon deux approches essentielles: l'approche à jeton et l'approche à permissions.

Dans l'approche à jeton, deux méthodes sont couramment utilisées et sont liées à la demande du jeton et à la circulation de jeton. Dans la première méthode, un noeud demandant la SC est amené à avoir le jeton ; le problème de base est comment l'atteindre. Dans certains algorithmes, la requête est adressée à tous les noeuds du réseau [1]; dans d'autres, une structure logique est définie pour désigner le détenteur du jeton, la requête est envoyée le long d'une branche d'un graphe asyclique dirigé en direction du noeud détenteur du jeton. Dans [12], Ricart et Agrawala proposent un algorithme, sur une topologie complète de réseau, qui requiert au maximum N messages pour réaliser l' EM. Suzuki et Kazami [18] proposent un algorithme, basé sur celui de Ricart et Agrawala, dans lequel la file de requêtes est transportée par le jeton. Singhal [13] a amélioré les performances de l'algorithme de Suzuki et Kazami jusqu'au maximum N messages dans le cas des charges élevées en utilisant une heuristique pour déterminer l'ensemble de noeuds qui peuvent posséder le jeton.

Dans l'approche à permission, pour entrer à sa SC, un noeud demandeur doit attendre de recevoir les réponses de tous les autres noeuds (ou d'un sous-ensemble dans certains algorithmes) du réseau. Dans l'algorithme proposé par Lamport [9], quand un noeud veut entrer à sa SC, il envoie une requête à tous les autres noeuds et attends leurs réponses. A la sortie de sa SC, il envoie un message de libération à tous les autres noeuds. Cet algorithme requiert 3(N-1) messages par entrée en SC.

Cependant, ces algorithmes initialement prévus pour les réseaux fixes ne peuvent pas s'appliquer aux environnements mobiles où les nœuds peuvent dynamiquement se connecter/déconnecter, se déplacer ou même tomber en panne. De plus, un nœud a une vision partielle du système limitée par sa portée de communication. Dans ce rapport, nous proposons un nouveau protocole d'EM pour les réseaux mobiles ad hoc basé sur l'approche à jeton. Le protocole ne repose pas sur la couche de routage, n'utilise pas de structure logique de communication et qui, pour la circulation du jeton, privilégie la satisfaction des demandes en fonction de leur ancienneté et selon une politique FIFO. Par ailleurs, le protocole proposé prend en charge quelques caractéristiques des réseaux mobiles ad hoc telles que : le mouvement des noeuds (i.e. les cassures et formations de liens). Le protocole favorise la scalabilité du réseau et considère un nombre indéterminé de noeuds.

Ce rapport est organisé comme suit : Après un bref aperçu sur quelques algorithmes déjà existant , dans la section 2, nous présentons les idées de base du protocole, les structures de données et les messages utilisés. Ensuite dans la section 3, une description du fonctionnement du protocole est donnée à travers les différents évènements qui le constituent. Un simple exemple d'exécution suivi d'une discussion sur certains aspects cachés du protocole sont donnés dans la section 4. La section 5 donne les différents résultats de simulation obtenus ainsi que leurs interprétations. Finalement, la section 6 conclue le rapport.

2 Préliminaires

Dans ce projet, nous travaillons sur une adaptation de l'algorithme de Suzuki Kazami sur un réseau sans fil ad-hoc qui est un type de réseau décentralisé avec une structure dynamique. Afin d'assurer le bon fonctionnement de l'algorithme d'EM proposé, plusieurs hypothèses ont été nécessaire

2.1 Modèle du réseau

Nous supposons un réseau connexe composé d'un nombre variable de nœuds mobiles, chaque nœud possède un identifiant unique. Les mouvements des nœuds du réseau sont libres (i.e. mobilité aléatoire). Les nœuds utilisent un support de communication sans fil, les liens physiques de communication sont bidirectionnels, FIFOs et fiables. Un nœud peut à tout moment quitter ou intégrer le réseau. Les mouvements des nœuds peuvent engendrer des créations et / ou des pertes de liens. Nous faisons aussi l'hypothèse que les pannes de noeud et des liaisons ne sont pas possibles et le réseau est toujours connexe.

2.2 Idées de base du protocole

Le protocole utilise l'approche à jeton dont l'unicité dans le réseau assure d'une manière intrinsèque l'EM dans celui-ci. Il combine la circulation et la demande de ce dernier. Le jeton est identifié de manière unique et transporte entre autres une file de requêtes construite par le detenteur du jeton quand il est en SC et la transmet au futur detenteur du jeton. Le protocole se base sur la connaissance locale, celle du voisinage immédiat et ne prend en considération que les liens physiques pour acheminer de l'information. Ce qui lui permet de favoriser la scalabilité du réseau et de considérer un nombre indéterminé de noeuds. D'autre part, le protocole fait recours à un certain nombre de mécanismes qui sont :

2.2.1 Rayon de diffusion

L'accès à la section critique est conditionné par la détention du jeton. Afin de l'acquérir, nous procédons à la diffusion de la requête. Pour minimiser le nombre de messages diffusés dans le cas d'un réseau constitué d'un grand nombre de nœuds nous procédons à la délimitation de la zone de diffusion,

pour cela nous introduisons le concept de zone de diffusion. Les requêtes valides reçues localement sont diffusées aux noeuds voisins jusqu'à atteindre le noeud détenteur du jeton. La zone de diffusion se base sur la propagation des ondes radio au niveau de la couche physique d'un noeud mobile. Cet aspect sera illustré lors de la conception de l'algorithme d'exclusion mutuelle.

2.2.2 Ancienneté des requêtes

Chaque site i détenteur du jeton reçoit une requête de demande d'entrée en section critique aura une connaissance de l'identité de l'émetteur de la requête. Du moment où la requête n'est pas satisfaite, le site i l'insère localement dans sa file d'attente. Afin d'assurer la vivacité de l'algorithme la liste des requêtes reçues par le détenteur du jeton est ordonnée par rapport a l'ancienneté de réception de la demande, suivant donc un ordre FIFO.

2.2.3 Recherche de route

La circulation des requêtes s'effectue par le processus d'inondation. Lorsque un site i reçoit une requête qui ne lui est pas destinée, il la diffuse à son tour à son voisinage local (zone de diffusion). Le processus d'inondation s'effectue une seule fois pour la même requête sur un site donné. Par exemple: le site 3 reçoit une requête à destination du site 10, il diffuse cette requête dans sa zone de diffusion locale. Le site 5 reçoit cette requête par le biais de la diffusion effectuée par le site 3 et à son tour effectue le même processus de diffusion pour atteindre le site 10. Dans le cas où le site 3 reçoit une deuxième fois la même requête, en comparant l'id du site initiateur et l'estampille, le site 3 ne participe plus au processus de diffusion car il a déjà diffusé cette même requête à t-2. Avec ce mécanisme nous évitons l'interblocage dans le réseau et la circulation infinie des requêtes qui provoque une surcharge au niveau du réseau dû aux nombre de messages en transit par inondation.

3 Description du protocole

Le texte du protocole est formé d'un ensemble de primitives liées aux différents événements le constituant. Au niveau de chaque noeud, les primitives s'exécutent de manière atomique, cependant des événements peuvent se produire durant l'exécution de la SC.

3.1 Propagation d'une requête

Un noeud i ayant comme identifiant id désirant accéder pour la k ème fois à sa SC est amené à diffuser sa demande à travers un message Request(id, K), dans son voisinage selon un rayon de diffusion comme décrit dans la section 2.2.1. Tous les noeuds intermédiaires recevant cette requête se chargent de la retransmettre une seul fois en la véhiculant de proche en proche selon le rayon indiqué jusqu'au nœud détenteur du jeton.

3.2 Conditions d'accès à la SC

Un noeud demandeur de SC accède à sa SC dans les cas suivants :

- Il détient le jeton à l'état de repos.
- Il reçoit le jeton et de plus, il est destinataire.
- Si le processus qui possède le jeton n'est pas en SC, il renvoie immédiatement le jeton au processus demandeur. Sinon, il attend la sortie de la SC et envoie le jeton au premier processus dont la requête n'a pas été satisfaite.
- Les requêtes pendantes sont transmises dans le message du jeton en respectant l'ordre FIFO.

3.3 Déplacement du jeton

Un noeud j détenant le jeton se charge de le véhiculer de proche en proche en utilisant le processus de transmission par inondation en prenant en compte la zone de diffusion de chaque site si l'un des cas suivants se présente :

- Le noeud j est à l'état de repos et il vient de recevoir une nouvelle requête.

- Le noeud j vient de sortir de sa SC et sa file d'attente contient au moins une requête.

En dehors des cas cités ci-dessus, le jeton est gardé localement.

3.4 Mise à jour des files d'attente

Rappelons que le service d'exclusion mutuelle entretient une file d'attente Q_i transportée par le jeton. La mise à jour de cette file se fait lorsque le détenteur du jeton reçoit une requête (Request) alors qu'il est lui même en SC ou lorsque une requête vient d'être satisfaite. Les mises à jour de ccette file permettent de purger les requêtes déjà satisfaites et de se rendre compte de celles qui sont nouvelles.

4 Exemple d'exécution et discussion

4.1 Exemple d'exécution

La figure 1 montre un scénario d'exécution du protocole d'une manière qui permet de faciliter la présentation.

Initialement le nœud 2 est porteur du jeton et les nœuds 6 et 10 veulent entrer en SC. On suppose qu'ils effectuent leurs (25, 30) ème requêtes. On note que la requête du noeud 6 atteint le noeud 2 avant la requête du noeud 10. Le réseau est connexe et les requêtes sont transmisent le long du chemin en broadcast. Les deux nœuds 6 et 10 diffusent leurs requêtes selon le rayon de diffusion dans l'optique d'atteindre le nœud détenteur du jeton. On note que la réception d'une requête déjà diffusée par le même nœud est simplement détruite et non consommée afin d'éviter l'interblocage.

Le noeud 6 diffuse sa requête contenant son id et le numéro de requête effectué, le noeud 7 réceptionne la requête car il fait partie de la portée du noeud 6. On note que le noeud 14 n'était pas dans la zone de diffusion du noeud 6 au moment de l'envoie de la requête, à son tour il incrémente l'entrée du vecteur des requêtes pendantes correspondant RN et diffuse la requête. Le message est transmis le long du chemin avec le même mécanisme jusqu'à ce qu'il arrive au niveau du nœud détenteur du jeton. Le nœud 2 reçoit les

deux demandes d'entrée en SC de la part du nœud 6 et 10. Puisque le mécanisme de traitement des requêtes pendantes au niveau du noeud détenteur du jeton est FIFO, la requête du noeud 6 est satisfaite en premier et le jeton (token) est envoyé au nœud demandeur avec le même mécanisme de diffusion jusqu'au nœud 6. On note que la file d'attente des requêtes non satisfaites ainsi que le vecteur des requêtes satisfaites LN sont transmis dans le jeton.

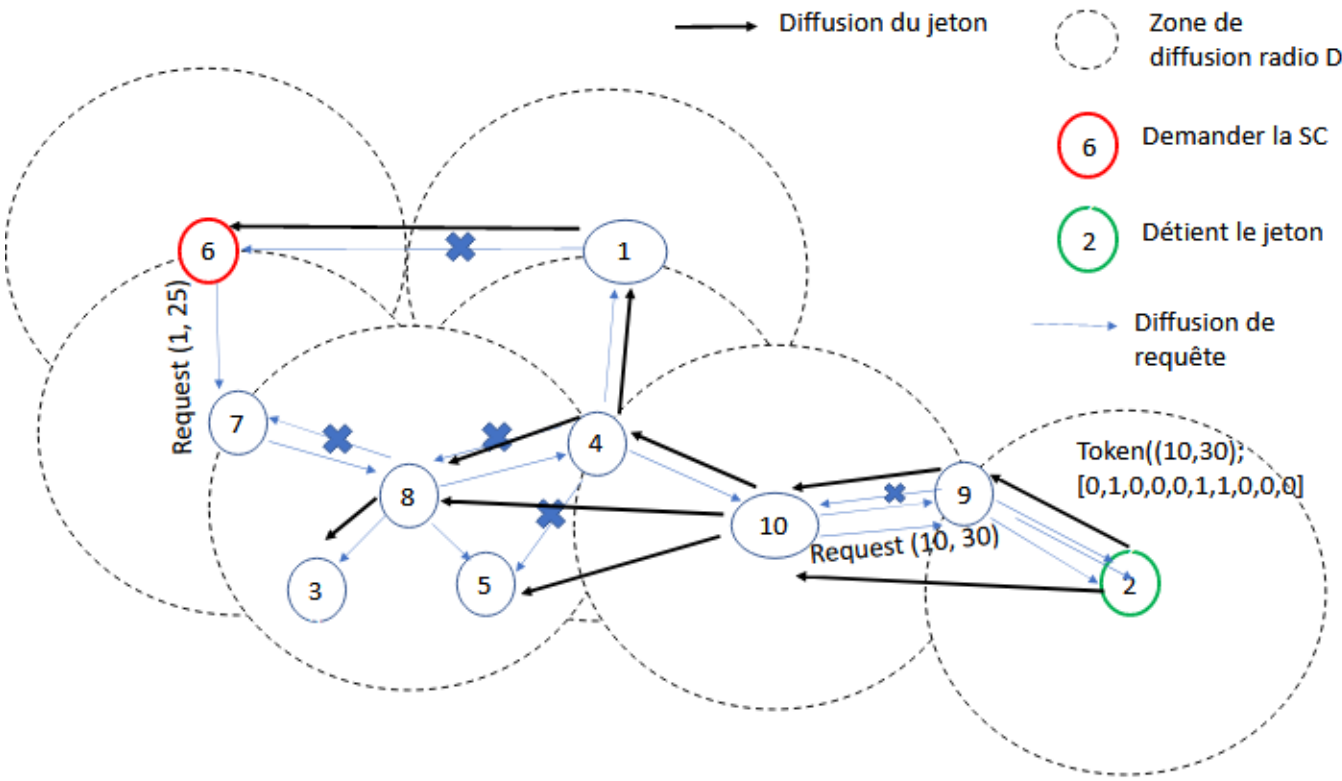

Figure 1: Un simple exemple d'exécution

Algorithm 1 Pseudo code du protocole

Require: Etat: requesting, not_requesting, critical_section.
Token: indique la présence du jeton sur le noeud.
RN: vecteur des requête différées.
LN: vecteur des requêtes satisfaites.
Q: file d'attente des requêtes non satisfaite transporté par le jeton.
RequestL: Liste des requêtes Request(id_{sender}, Clock) reçu.
TokenL: Liste des requêtes token(id_{dst}, Clock) reçu.

Ensure: gestion des entrées/sorties de la SC et acheminement du jeton.

C_i: l'horloge locale du site i.

Initialisation des variables locales (S_j)
Token= ($S_i == S_1$)
Etat = not_requesting;
$RN[j] = 0, j = 1, 2, ...N$;
$LN[j] = 0, j = 1, 2, ...N$;
$Q = \emptyset$

Request_CS (i):
Etat=requesting
if $Token == false$ **then**
RN[i] = RN[i] +1
diffuser REQUEST(Si,k)
attendre (Token == true)
end if
Etat = critical_section;

Release_CS (i):
$LN[i] = RN[i]$;
if Q != $\emptyset$ **then**
Token = false;
site = extraire (Q);
diffuser TOKEN (Q,LN)

Receive_Request_CS(S_j, REQUEST (j,k))
$RN[j] = max(RN[j], k)$
if $((RequestL.id == j)$**and**$(Reequest.H > RequestL.Clock))$ **then**
Remove($RequestL[j]$)
end if

if $(Token = true)$**and**$(Etat == not\_requesting)$**and**$(RN[j] > LN[j])$ **then**
Token = false;
diffuser TOKEN (Q,LN)
else if $(Token == false)$ **then**
diffuser REQUEST(j,k)
end if
if $(etat == critical\_section$ **then**
Ajouter j à Q
end if

Recieve_Token (TOKEN (Q,LN))
if Token.dest == S_j **then**
Token = true;
LN = TOKEN.LN ;
Q = TOKEN.Q
else
diffuser TOKEN (Q,LN)
end if

L'algorithme d'exclusion mutuelle conçu pour assurer un accès en EM à la section critique pour les réseaux mobiles satisfait les propriétés suivante:

- Sûreté:à tout instant il y a au plus un processus dans la section critique. Si ($etat_i = critical\_section$ et $token == true$) alors ($etat_j! = critical\_section$) pour tout $j <> i$
- Vivacité :Tout processus qui demande la section critique l'obtiendra au bout d'un temps fini (qui peut être grand mais finira par l'obtenir). ($etat_i = requesting$) apres un certain temps ($etat_i = critical\_section$) Garantir l'absence d'interblocage et de famine.

Note: après l'implémentation et les tests de performance il s'est avéré qu'il y a quelques interblocages dû à l'algorithme de mobilité choisi, qui assure un

mouvement aléatoire linéaire des nœuds mobile. Nous adapterons notre solution au autres algorithmes de mobilité (modèle de mobilité à marche aléatoire restreinte, modèle de mobilité markovien normal...) pour nos travaux futur.

5 Simulation

L'évaluation des performances du protocole est effectuée à l'aide de l'outil Omnet++ version 5.6 portable sur les systèmes Windows et Linux avec l'extention Inet.Le simulateur permet la représentation des réseaux ad-hoc avec un nombre paramétrable d'hôtes qui se déplacent dans un espace libre d'obstacles. Chaque hôte a une puissance d'émission définie qui affectent la portée des communications. Chaque hôte mobile est un module composé qui encapsule les modules suivants:

- La couche physique
- La couche Mac
- La couche de routage
- La couche applicative

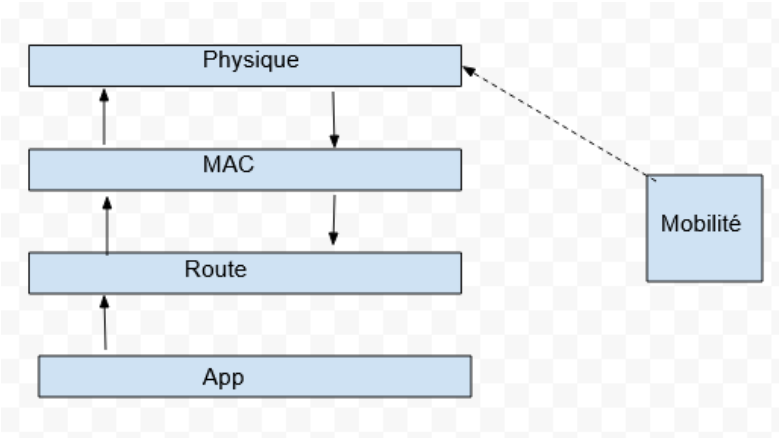

Figure 2: La structure interne d'un noeud ad-hoc

Dans notre cas, la couche de routage n'a pas été exploitée. Les communications entre les couches s'effectuent via des messages d'échange (des broadcasts). Les paramètres de simulation sont effectuées au niveau du fichier omnetpp.ini.

5.1 La couche physique

Le simulateur Omnet implémente la couche physique de chaque hôte. En particulier, il se soucie de la création à la volée des sockets qui permettent l'échange de messages entre les hôtes. Cette capacité dynamique représente une contribution importante au module sans fil existant pour OMNeT ++ (Inet).

Chaque fois qu'un hôte se déplace de sa position, une vérification interdistance sur chaque nœud est effectuée. Si un hôte se rapproche suffisamment (en fonction de la puissance de transmission du nœud mobile) d'un nouveau voisin, ces opérations ont lieu:

- Une nouvelle socket est créée pour les deux hôtes.
- Un lien physique bidirectionnel est créé entre les deux hôtes.

Cela donne à la liaison des caractéristiques de retard, de débit et de probabilité d'erreur. Chaque nœud a sa propre puissance de transmission et une portée.

5.2 La mobilité

Le modèle de mobilité est le seul module qui ne rentre pas dans la structure de la couche protocolaire OSI des hôtes ad-hoc. Comme son nom l'indique, ce module se soucie de la mobilité du nœud dans lequel est encapsulé. Chaque hôte possède son propre module de mobilité. En changeant simplement l'algorithme de mobilité choisi au niveau du fichier omnetpp.ini.

Nous avons opté pour une mobilité aléatoire des noeuds qui est caractérisé par un modèle de mouvement totalement imprévisible où la vitesse et la direction ne sont pas corrélées. En raison de son caractère aléatoire, il produit un mouvement assez irréaliste. Quoi qu'il en soit, ce type de mouvement des

noeuds est souvent utilisé comme un algorithme de mouvement du pire des cas.

5.3 La couche MAC

Ce module décrit la couche MAC du modèle OSI. Il est possible d'insérer différents protocoles de contention de canaux tels que CSMA / CA, MACA, MACAW et tout autre algorithme existant.

Dans notre protocole, la couche implémentée est beaucoup plus simple. Les messages sortants sont laissés passer. Les entrants sont à la place livrés aux niveaux supérieurs avec une politique de file d'attente expliquée dans la section 3.

5.4 Génération du trafic

Au niveau de chaque nœud un trafic est généré (envoi des requêtes). Au niveau du fichier node.cc un auto-message est ajouté pour déclencher l'opération d'envoi des requêtes. Le trafic est modélisé en générant une rafale de paquets de requêtes en broadcast vers la destination qui détient le jeton, le chemin emprunté entre la source émettrice de la requête et le nœud détenteur du jeton est aléatoire. Le débit d'envoi de messages est défini comme paramètre dans le omnet.ini.

6 Analyse des performances

Pour étudier le comportement de l'approche d'exclusion mutuelle sur un réseau ad hoc, les tests suivants ont été effectué:

- Temps moyen d'obtention de la SC en fonction du nombre de noeuds présent dans le réseau.
- Temps moyen d'obtention de la SC en fonction de la mobilité des noeuds: le temps écoulé entre l'envoi de la requête de demande d'entrée en SC et l'obtention du jeton. Étant donné que seuls les paquets de données qui survivent tout au long du chemin vers les destinations sont pris en compte (car on introduit une probabilité d'erreur), un faible temps signifie que la plupart des paquets de demande d'SC sont livrés au noeud détenteur du jeton qui est au voisinage proche, et dans le cas

ou le noeud détenteur du jeton est distant par rapport au noeud demandeur, le temps de d'exécution est plus important. Ainsi, la mesure du temps moyen d'obtention de la SC nous fournit des informations sur la capacité de survie du protocole.

6.1 Paramètres de simulation

Toutes les analyses de simulation ont été effectuées avec le réglage suivant:
Les hôtes

Nombre de noeuds	100
Nombre de noeuds qui émettent une requête	100

La couche physique

Porté de communication	120m
Bande passante	1 Mb/s (802.11a)
Latence du chanal	300 us
Probabilité d'erreur	variable selon les testes

Le message (la data)

Taille du message	200 Bytes
Intervalle d'envoi	[0,1s]

6.2 Temps moyen d'obtention de la SC en fonction du nombre de noeuds dans le réseau

Temps moyen d'obtention de la SC en fonction du nombre de noeuds dans le réseau est le temps nécessaire pour qu'un noeud reçoit le jeton (jeton livré à la destination) suite à une demande d'entrée en SC. Le graphique présenté ici montre le comportement de l'approche d'exclusion mutuelle proposée en référence au temps moyen d'obtention de la SC et le nombre de noeuds du réseau. La mobilité des noeuds est figée (vitesse moyenne des hôtes).

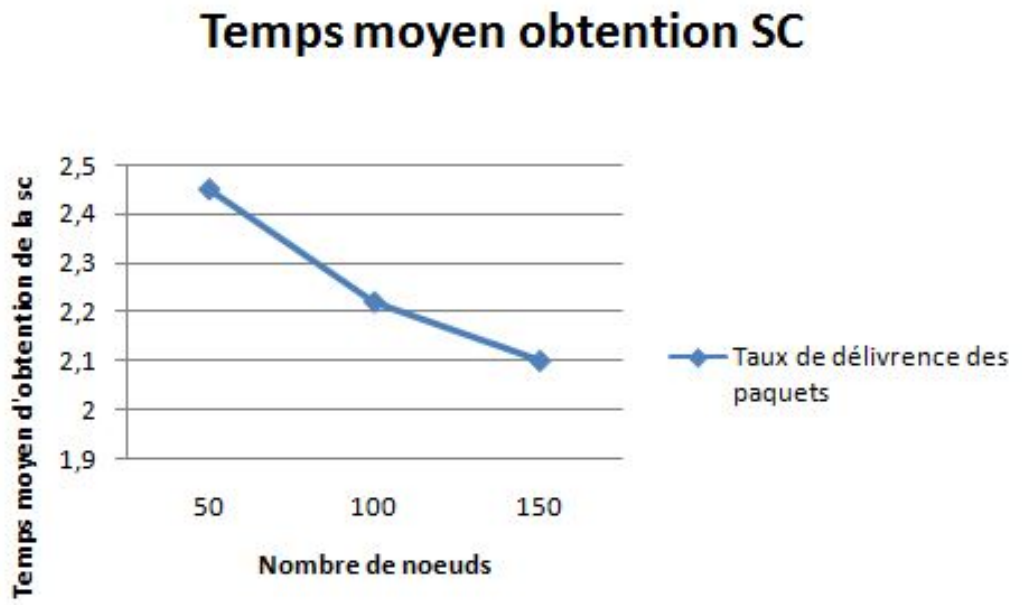

Figure 3: Temps moyen d'obtention de la SC en fonction de la mobilité des noeuds en m/s

6.3 Temps moyen d'obtention de la SC en fonction de la mobilité

Cette métrique montre le temps moyen qu'un paquet a effectué pour arriver à sa destination. Seuls les messages correctement livrés participent au décompte. Un nœud désirant accéder en SC diffuse sa requête dans le réseau pour atteindre le nœud détenteur du jeton.Le temps de transfert est faible lorsque le noeud détenteur du jeton est à proximité proche, par contre le temps devient plus élevée lorsque le noeud demandeur et le noeud détenteur du jeton sont éloigné.

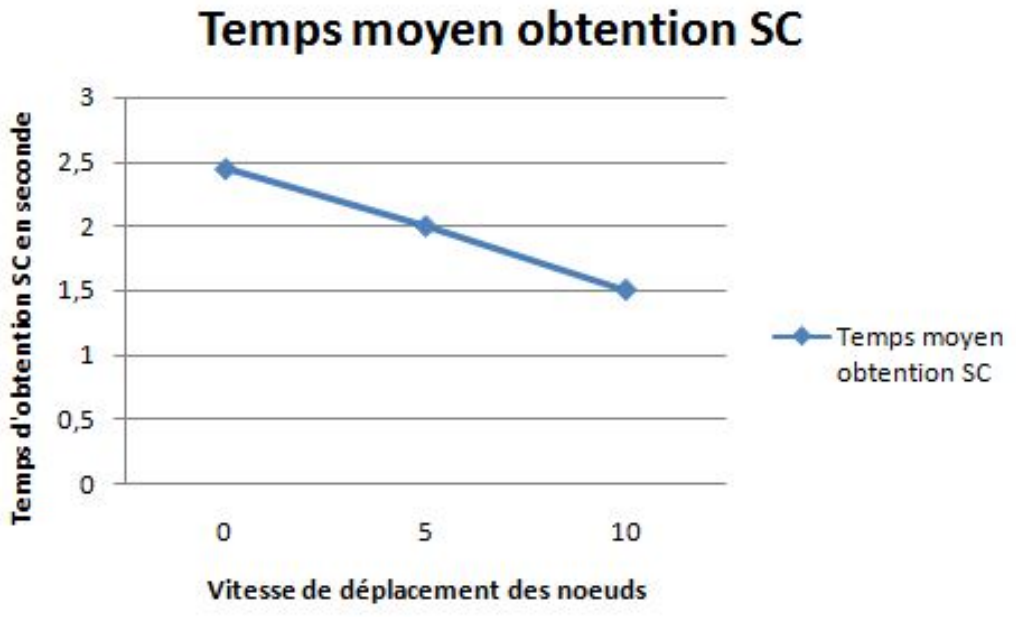

Figure 4: Temps moyen d'obtention de la SC en fonction de la mobilité des noeuds en m/s

7 Conclusion

Dans ce rapport, nous avons présenté une solution au problème d'EM dans les réseaux mobiles ad hoc. Le protocole présenté se base sur les liens physiques, n'utilise aucune structure logique. Les requêtes sont acheminées en broadcast suivant la portée de chaque nœud qui est son rayon de diffusion lui permettant une connaissance locale afin de couvrir une proportion de nœuds demandeurs.

Le jeton est géré selon une méthode qui combine la circulation et la demande. De plus, la politique de satisfaction des nœuds favorise les requêtes les plus anciennes selon un ordre FIFO. La mobilité des nœuds fait partie intégrante de la solution proposée grâce aux différents mécanismes présentés dans ce qui précédait. Cependant, le protocole ne prend pas en charge les mécanismes de régénération du jeton en cas de perte et ne tolère pas les

partitionnements du réseau.

A partir des différentes métriques effectuées, nous pouvons constater que le protocole donne des résultats satisfaisants dans sa globalité. Le protocole se montre bien adapté à la mobilité, car cette dernière n'influe pas beaucoup sur ses performances globales. Le nombre de messages émis par SC est assez faible surtout pour les faibles et moyennes charges car les nœuds demandeurs et les nœuds porteurs du jeton se trouvent à proximité (leurs portée se chevauchent).

Comme perspectives, nous aimerions améliorer notre protocole en modifiant la politique de satisfaction des nœuds en favorisant les nœuds les plus proches. Aussi, nous travaillerons sur la prise en charge des pannes des nœuds et le mécanisme de recherche et régénération du jeton en cas de perte.

References

[1] Divyakant Agrawal and Amr El Abbadi. An efficient and fault-tolerant solution for distributed mutual exclusion. 9(1):1–20.

[2] Hagit Attiya, Alex Kogan, and Jennifer L. Welch. Efficient and robust local mutual exclusion in mobile ad hoc networks. 9(3):361–375.

[3] B. R. Badrinath, Arup Acharya, and Tomasz Imielinski. Designing distributed algorithms for mobile computing networks. 19(4):309–320.

[4] B. R. Badrinath, Arup Acharya, and Tomasz Imielinski. Structuring distributed algorithms for mobile hosts. In 14th International Conference on Distributed Computing Systems, pages 21–28. IEEE.

[5] Y.-I. Chang, Mukesh Singhal, and Ming T. Liu. A fault tolerant algorithm for distributed mutual exclusion. In Proceedings Ninth Symposium on Reliable Distributed Systems, pages 146–154. IEEE.

[6] Andrzej Goscinski. Two algorithms for mutual exclusion in real-time distributed computer systems. 9(1):77–82.

[7] Jean-Michel Helary, Noel Plouzeau, and Michel Raynal. A distributed algorithm for mutual exclusion in an arbitrary network. 31(4):289–295.

[8] Ashish Khanna, Joel JPC Rodrigues, Naman Gupta, Abhishek Swaroop, Deepak Gupta, Kashif Saleem, and Victor Hugo C. de Albuquerque. A mutual exclusion algorithm for flying ad hoc networks. 76:82–93.

[9] Leslie LAMPORT. Time, clocks, and the ordering of events in a distributed system. (5):179–196.

[10] Shojiro Nishio, Kin F. Li, and Eric G. Manning. A resilient mutual exclusion algorithm for computer networks. 1(3):344–356.

[11] Michel Raynal. A simple taxonomy for distributed mutual exclusion algorithms. 25(2):47–50.

[12] Glenn Ricart and Ashok K. Agrawala. An optimal algorithm for mutual exclusion in computer networks. 24(1):9–17.

[13] Mukesh SINGHAL. A heuristically-aided algorithm for mutual exclusion in distributed systems. 38(5):651–662.

[14] Ichiro Suzuki and Tadao Kasami. A distributed mutual exclusion algorithm. 3(4):344–349.

[15] Jennifer E. Walter, Jennifer L. Welch, and Nitin H. Vaidya. A mutual exclusion algorithm for ad hoc mobile networks. 7(6):585–600.