الجمـــهوريـــة الجـــزائريـــة الديـمقراطيـــــــة الشـعبيـــــة
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعـلـيـم الـعـالـــي و الـبـحـث الـعـلـمي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جـامـعـة عمّـــار ثـلـيـــجـــي بالأغـــواط
UNIVERSITE AMAR TELIDJI LAGHOUAT
كـلـيـــــة الـعـلـــــوم
FACULTE DES SCIENCES
قسم الاعلام الالي
DEPARTEMENT DE L'INFORMATIQUE

Mémoire de MASTER

Domaine : Mathématiques et Informatique
Filière : Informatiques
Option : Réseaux, Systèmes et Applications Réparties

Par:
Azzouz Bouchra

THEME

Simulation d'une nouvelle technique de choix de leaders dans un algorithme de partage de ressources tolérant aux pannes dans les réseaux mobiles ad hoc

Soutenu publiquement le 08-06-2017 devant le jury composé de:

Mlle S BenKouider	*M.C.(A)*	*Président*
Mlle Z AbdElhafidhi	*M.C.(A)*	*Examinatrice*
Mr T Allaoui	*M.C.(A)*	*Encadreur*
Mr Ch Kerrache	*M.C.(A)*	*Co-Encadreur*

Année Universitaire 2016/2017


بسم الله الرحمن الرحيم

Dédicace

Je dédie ce mémoire

à

Ma grande mère qu'Allah bénisse son âme

Mes parents

Mes soeurs, Mon frère

et Mes amies

Remerciement

Tout d'abord je remercie ALLAH qui m'a aidé et m'a donné la patience et le courage durant ces 5 ans d'étude.

Un grand merci à ma mère et mon père, pour leur amour, leurs conseils ainsi que leur soutien inconditionnel, qui m'a permis de réaliser les études que je voulais et par conséquent ce mémoire.

Je tien é remercier notre enseignant Mr Allaoui parce qu'il a accepté de m'encadrer, et aussi pour l'orientation, la patience qui ont constitué un apport considérable sans lequel ce travail n'aurait pas pu être mené au bon port.

On remercie également Mr Kerrache "mon CO-encadreur" pour ses bonnes explications qui m'ont éclairé le but de ce travail et sa collaboration avec moi pour accomplir ce modeste travail.

On tient à exprimer mes sincères remerciements à tous les professeurs qui m'ont enseigné et qui par leurs compétences m'ont soutenu dans la poursuite de mes études.

Résumé

La capacité de traitement des terminaux mobiles augmente de plus en plus, mais dans la plupart des temps on arrive pas à répartir la charge entre les noeuds, ceci est d'autant vrai lorsque les terminaux sont hétérogènes, ils ont donc des capacités de calcul et sources d'énergie différentes ce qu'il conduit à des situations où quelques terminaux sont fortement chargés, tandis que le reste est faiblement chargés. Donc il faut toujours prendre en considération les caractéristiques des MANETs lors de la conception d'un nouveau protocole pour les réseaux mobile ad hoc.

Ce mémoire présente une nouvelle technique de choix de leaders dans un algorithme de partage de ressources dans les réseaux mobiles ad hoc tolérant aux pannes et il assure l'équilibrage de charges.

Premièrement, on présente une vue générale sur les réseaux mobiles ad hoc, ensuite on explique le problème de partage de ressource dans les systèmes distribués et dans les MANETs.

Deuxièmement on présente notre nouvelle technique de choix de leaders dans un algorithme de partage de ressources dans les MANETs qu'il assure l'équilibrage de charges, et on le compare avec l'ancienne version [1] qui ne traite pas le problème de l'équilibrage de charges. Et on essaye d'intégrer le mécanisme de tolérance aux pannes dans notre algorithme. La comparaison a été faite après la simulation de nos algorithmes à l'aide du simulateur $NS2$.

Mots-clés : Réseaux mobiles ad hoc, MANET systèmes distribués, le partage de ressources, l'équilibrage de charges, tolérance aux pannes, Simulation, $NS2$.

Abstract

The processing capacity of the mobile terminals increases more and more, but in most times it is not possible to distribute the load between the nodes, this is all the more true when the terminals are heterogeneous, they therefore have different computing capacities and different energy sources what it will lead to situations where some terminals are heavily loaded, while the rest is weakly charged. So always we should be aware when designing a new protocol for mobile ad hoc networks.

This paper presents a new technical of selecting leaders in fault-tolerant k mutual exclusion algorithm in mobile ad hoc networks that ensures load balancing.

First, we will see a general view about ad hoc mobile networks, then we will explain the problem of resources sharing in distributed systems and in MANETs.

Secondly we will present our algorithm of selecting leaders in fault-tolerant k mutual exclusion algorithm in MANETs, that ensures the loads balancing, and we will compare it with the old version which does not deal with the problem of loads balancing. And we will integrate the mechanism of fault tolerance in our algorithm. The comparison was made after the simulation of our algorithms using $NS2$ simulator.

Keywords : Mobile Ad hoc networks, MANET, distributed systems, resource sharing, load balancing, fault tolerance, simulation, $NS2$.

Liste des acronymes

MANET	Mobile Ad-hoc Networks
NAM	Network AniMator
AODV	Ad hoc On-Demand Distance Vector
NS-2	Network Simulator 2
TAM	Temps d'attente Moyen
NMM	Nombre de messages moyen
TUM ou **TUEM**	Taux d'utilisation d'énergie moyen
ET	Écart-type
SC	Section Critique
TCL	Tool Command Language
OTCL	Object Tool Command Language

Table des figures

Liste des tableaux

Table des matières

Introduction générale

Contexte et problématiques

L'évolution des technologies sans fil, offre aujourd'hui de nouvelles perspectives dans le domaine des télécommunications. L'évolution récente des moyens de la communication sans fil a permis la manipulation de l'information à travers des unités de calculs portables qui ont des caractéristiques particulières.

L'environnement mobile permet aux unités de calcul, une libre mobilité et il ne pose aucune restriction sur la localisation des usagers. La mobilité et le nouveau mode de communication utilisé engendrent de nouvelles caractéristiques propres à l'environnement mobile : une fréquente déconnexion, un débit de communication et des ressources modestes, et des sources d'énergie limitées. On distingue deux grandes familles de réseaux mobiles sans fil : les réseaux mobiles avec infrastructure (cellulaire) et les réseaux mobiles sans infrastructure (Ad Hoc).

Un réseau mobile ad hoc est une collection de noeuds sans fil qui peuvent être configurés dynamiquement n'importe où et n'importe quand sans utiliser une infrastructure de réseau préexistante. C'est un système autonome dans lequel les hôtes mobiles sont connectés par des liaisons sans fil et ils sont libres de se déplacer aléatoirement et agissent souvent comme des routeurs en même temps.

La non-existence d'un point central dans les réseaux mobiles ad hoc qui gère les différents fonctions dans le réseau pose beaucoup de problèmes et il rend plusieurs opérations difficiles à manipulé tels que le partage de ressources entre les terminaux de réseau, ainsi que la détection des pannes des terminaux et l'équilibrage de charge.

Objectif

L'objectif de ce mémoire est d'ajouter une amélioration à l'algorithme proposé dans [1] pour le rendre performant en terme de l'équilibrage de charges entre les noeuds. Donc on va utiliser une nouvelle technique de choix de leaders à l'algorithme [1], et on va essayer d'investir une des

méthodes utilisés dans les algorithmes de tolérance aux pannes pour le mettre performant.

Organisation du mémoire

Le mémoire est organisé comme suit :

Dans **le premier chapitre**, on va présenter les concepts de base des réseau mobile ad hoc, et on va décrire les caractéristiques, les avantages, et les applications de ce genre de réseaux. On s'intéresse aussi dans ce chapitre au problème de partage de ressource dans les systèmes distribués ainsi dans les réseaux mobiles ad hoc, et on va parler sur la notion de l'exclusion mutuelle et sa généralisation la k-exclusion mutuelle.

Dans **le deuxième chapitre**, on va présenter une nouvelle technique de choix de leaders dans un algorithme de partage de ressource dans les MANETs qui sert à partager k ressources entre les noeuds, en assurant un équilibre dans le niveau d'énergie des terminaux. Et on va le simuler en utilisant le fameux simulateur $NS2$ en variant les paramètres à chaque fois.

Dans **le troisième chapitre**, on va améliorer l'algorithme proposé dans le deuxième chapitre pour qu'il soit tolérant aux pannes, et on va le simuler et interpréter les résultats obtenus.

Finalement, on termine le mémoire par une conclusion qui résume notre travail.

Chapitre 1

Réseaux mobiles ad hoc : Vue générale, problème de partage de ressources

1.1 Introduction

L'évolution récente de la technologie dans le domaine de la communication sans fil et l'apparition des unités de calculs portables, poussent aujourd'hui les chercheurs à faire des efforts afin de réaliser le but des réseaux : " l'accès à l'information n'importe où et n'importe quel moment".
En citant le réseau mobile ad hoc (MANET), c'est une collection d'unités de calculs mobiles sans fil qui peuvent dynamiquement former un réseau pour échanger des informations sans utiliser une infrastructure de réseau fixe préexistante. Comparant avec un environnement statique, le nouvel environnement mobile, permet aux unités de calcul une libre mobilité, et il ne pose aucune restriction sur la localisation des usagers.
La mobilité et le nouveau mode de communication utilisé engendrent de nouvelles caractéristiques propres à l'environnement mobile : une fréquente déconnexion, des ressources modestes, et des sources d'énergie limitées.
Dans le réseau mobile ad hoc on trouve le problème de partage de ressource, où les unités mobiles peuvent partager une ou plus d'une ressource, donc il doit y avoir un algorithme d'exclusion mutuelle différent, sophistiqué pour contrôler l'accès aux ressources et adapté aux caractéristiques de réseau mobile ad hoc.

Dans ce chapitre, on va présenter le concept des réseaux mobiles ad hoc, un historique sur les MANETs, ses caractéristique, ses avantages, ses applications et les critères à prendre en considération lors du déploiement d'un MANET et on finira avec une conclusion.
On va aussi expliquer le problème de partage de ressources et on va définir c'est quoi une exclusion mutuelle, le type des solutions disponibles. Puis on va voir le problème d'exclusion et de la k-exclusion mutuelle dans les systèmes distribués afin de clarifier notre objectif qui est le problème de la k-exclusion mutuelle dans les réseau mobiles ad hoc.

1.2 Présentation des réseaux mobile ad hoc

1.2.1 Historique sur les MANETs :

La première génération de réseau ad hoc a été tracé aux années 1970s. Dans les années 1970s,ils sont appelés Packet Radio Network (PRNET). La DARPA (Defense Advanced Research Project Agency) a entrepris des recherches sur l'utilisation de la communication par commutation par paquets pour assurer une communication fiable entre les ordinateurs et les PRNET urbanisés. Fondamentalement, PRNET utilise la combinaison entre Areal Location of Hazardous Atmospheres(ALOHA) et Carrier Sense Multiple Access (CSMA) pour le routage à accès multiple et à vecteur de distance.

Le PRNET est ensuite transformé en Survivable Adaptive Radio Network (SURAN) au début des années 1980s. SURAN offre certains avantages en améliorant les performances de la radio (les rendant plus petits, moins chers et puissants). SURAN fournir aussi une résistance aux attaques électroniques.

Vers la même époque, United State Department of Defence (DOD) a continué de financer des programmes comme Globe Mobile Information System (GloMo) et Near Term Digital Radio (NTDR). GloMo utilise CSMA/CA et TDMA et fournit un réseau auto-organisé et auto-cicatrisant. La NTDR utilise le clustering et link state routing et organise un réseau ad hoc. NTDR est porté par l'armée américaine. Il s'agit le seul réseau ad hoc réel utilisé.

En raison de l'intérêt porté aux réseaux ad hoc, d'autres grands développements ont eu lieu en 1990. Le groupe fonctionnel de MANET est né dans l'Internet Engineering Task Force (IETF) qui a travaillé à des protocoles de routage standardisés pour MANET et donne lieu au développement de divers dispositifs mobiles comme PDAs, ordinateurs de poche(palmtops), notebooks , etc.

Pendant ce temps le développement de la norme IEEE 802.11 (WLAN) ont bénéficié au réseau ad hoc. D'autres normes sont également développées qui apportent des avantages au MANET comme Bluetooth et HIPERLAN.[4]

1.2.2 Définition des MANETs :

Les réseaux ad hoc aux quels nous nous sommes intéressés sont ceux décrits et étudiés par le groupe de travail MANET (Mobil Ad hoc Network) de l'IETF (Internet Engineering Tast Force). C'est un ensemble de terminaux mobiles ou non c'est en fonction de la nature de réseau, ces terminaux sont inter-connectés par des liaisons sans fils, ils sont des réseaux sans infrastructure et ils sont indépendants d'une administration centralisée.

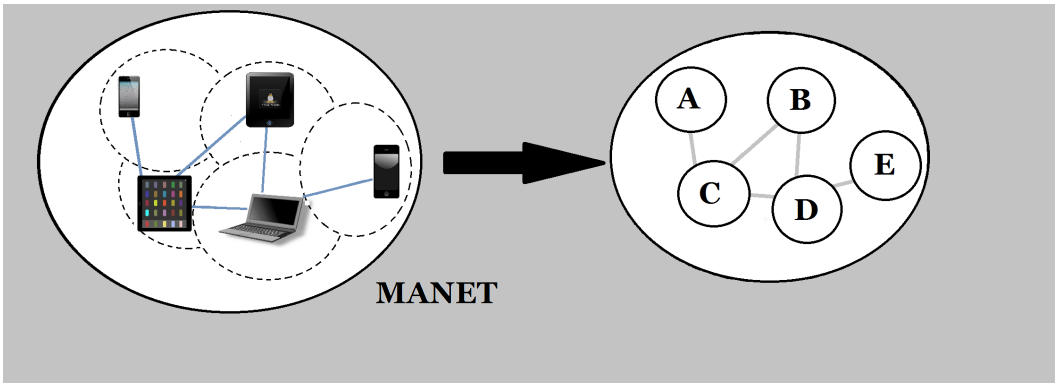


FIGURE 1.1 – Le réseau mobile ad hoc

1.3 Les caractéristiques des réseaux mobile ad hoc[9]

Terminal autonome : Chaque terminal mobile est un noeud autonome qui fonctionne comme un hôte ou un routeur. A coté de traitement de base d'un hôte les noeuds mobiles peuvent effectuer la fonction de transfert exactement comme un routeur.

Une opération distribuée : Il n'y a pas un contexte d'un réseau pour le contrôle central des opération d'un réseau, alors la gestion et le contrôle de réseau est distribues entre les terminaux, les noeuds collaborent entre eux et chaque noeud agit comme un relai pour mettre à jour des fonctions.

Un routage multi-saut : Quand la transmission d'un paquet de données d'une source à une destination, et que cette dernière est hors la portée de communication de l'émettrice, le paquet doit être envoyer via un ou plusieurs noeuds intermédiaires.

Une topologie dynamique : Puisque les noeuds sont mobiles, la topologie du réseau peut changer rapidement et imprévisiblement et la connectivité entre les terminaux varie avec le temps. Les noeuds mobiles dans le réseau dynamique établissent une connectivité entre eux en formant leur propre réseau.

la fluctuation de la capacité des liens : Le canal où les terminaux communiquent est soumis au bruit, dépérissement, interférence et plus que ça il a une bande passante limitée par rapport au réseau filaire.Des fois le chemin peut traverser plusieurs liaisons, et les liaisons elles-mêmes peuvent être hétérogènes.

Sécurité et vulnérabilité : Dans les réseaux ad hoc, le principal problème ne se situe pas tant au niveau du support physique mais principalement dans le fait que tous les noeuds sont équivalents et potentiellement nécessaires au fonctionnement du réseau. Les possibilités de s'insérer dans le réseau sont plus grandes, la détection d'une intrusion ou d'un déni de service plus délicate et l'absence de centralisation pose un problème de remontée de l'information de détection d'intrusions.

Des terminaux de petit poids : Dans la plupart des cas les noeuds sont des appareils mobiles avec un CPU de capacité de traitement très petite, une taille de mémoire petite aussi, et un stockage d'énergie très faible.Chaque appareil a besoin d'un algorithme optimisé et un mécanisme qui mit en place le calcul et les fonctions de communication.

1.4 Les avantages des réseaux mobile ad hoc[9]

- Ils fournissent un accès aux informations et services sans tenir compte la position géographique.
- Moins chère en comparant avec les réseaux filaires.
- Scalable, il accommode d'ajouter plus des noeuds.
- Les noeuds sont indépendants d'une administration centralisée.
- réseau peut être installer n'importe où à n'importe quel moment.
- Robustesse due à l'administration décentralisée.

1.5 Les applications des réseaux mobile ad hoc[4]

L'auto-reconfiguration, le déploiement facile, la décentralisation et l'indépendance de l'infrastructure de MANET conviennent à la communication.

- Anciennement, MANET a été utilisé pour la communication dans les applications militaires.
- MANETs sont largement utilisés dans les endroits où l'infrastructure fixe pour la communication est détruite, ou dans les situations comme le tremblement de terre, inondation, les explosions, les accidents aériens, les zones de catastrophe et la calamité naturelle.
- MANETs jouent un rôle vital dans le contrôle de foule et la surveillance.
- La flexibilité a avancé son utilisation dans les applications d'affaires telles que les conférences, le transfert de fichiers, les applications web, l'automatisation des maisons par exemple : le verrouillage et le déverrouillage des porte, et pour faire fonctionner les lumières à distance.

— D'autres applications incluent des procédures d'enregistrement en vol et la gestion de trafic.

1.6 Les critères à prendre en considération lors du déploiement d'un MANET[9]

L'imprévisibilité de l'environnement : Les réseaux ad hoc peuvent être déployer dans des terrains inconnus, des conditions dangereuses et même des environnements hostiles, où la falsification ou la destruction d'un noeud peut être imminente. Selon l'environnement les échecs d'un noeud peuvent se produire fréquemment.

La non fiabilité du médium sans fil : La communication via un médium sans fil est non fiable et enclin aux erreurs due à la variation des conditions de l'environnement, donc la qualité des liaisons sans fils reste imprévisible.

Ressources limitées : Les noeuds en MANET ont des ressources très limitées tels que l'énergie, la capacité de calcul. Si les noeuds se situent dans une place où il n'est pas possible de recharger la batterie, pour bien exploiter cette limitation ils doivent avoir des algorithmes écoénergitique.

La topologie dynamique : La topologie de réseau mobile ad hoc peut changer constamment à cause de la mobilité des noeuds, durant le mouvement des noeuds il se peut que certains noeuds quittent la portée d'un certains noeuds et entre dans un autre, ça va résulter la rupture de quelque liaisons et la création d'autre.

En raison de ces problèmes, les MANET sont enclins à de nombreux types de défauts, tels que :

— Les erreurs de transmission : La non fiabilité du médium sans fil et l'imprévisibilité de l'environnement peuvent entrainer d'envoyer un paquet déformé et recevoir un paquet erroné.
— L'échec d'un noeud : les noeuds peuvent échouer à tous moments à cause des conditions de l'environnement, ils peuvent quitter le réseau volontairement ou à cause de l'épuisement de l'énergie.
— L'échec d'une liaison : les échecs des noeuds vont causer la rupture des liaisons entre les noeuds, et les échecs des liaisons vont obliger la source de trouver des nouvelles routes à travers autres liaisons.
— La ruptures des routes : Quand la topologie de réseau change les routes deviennent expirées et incorrectes, donc les paquets sont envoyés via des routes viciées, qui peuvent éventuellement être abandonnés ou retardés.
— Les noeuds et les liaisons sont congestionnés : En raison de la topologie de réseau et la nature des protocoles de routage, certains noeuds deviennent sur-utilisés (congestionnés) et ça va provoquer soit un retardement ou bien une perte de paquets.

1.7 Le problème de partage de ressources

1.7.1 L'exclusion mutuelle

On dit qu'une ressource peut exécuter en exclusion mutuelle lorsqu'elle est partagée entre plusieurs processus et qu'il faut qu'un seul processus qu'il l'utilise à la fois pour éviter l'incohérence.

1. **Ressource critique** :une ressource qui doit être utilisée par qu'un seul processus à la fois.
2. **Section critique** :c'est une section de code où le processus manipule la ressource critique.

Pour garantir l'accès exclusif à la ressource critique il doit y avoir un un protocole d'acquisition pour retarder l'accès du processus à la section critique, et un protocole de libération qui doit être exécuter après la section critique pour dire aux autres processus demandeurs de cette ressource qu'elle est libre. La forme générale d'un processus qui utilise une ressource critique est comme suite :

Protocole d'acquisition
<**Section critique**>
Protocole de libération

1.7.2 Les états possibles d'un processus

Un processus peut prendre plusieurs états :

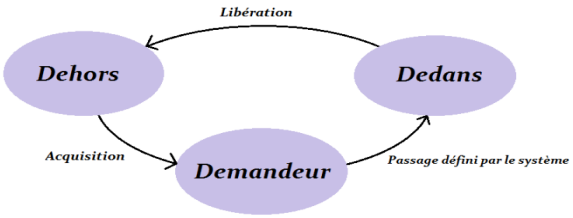


FIGURE 1.2 – Les états d'un processus.

— **Demandeur** : il prend cet état quand il demande à utiliser la ressource critique.

— **Dedans** : quand il est entrain d'exécuter la section critique.

— **Dehors** : quand il est ni à l'état demandeur ni à l'état dedans.

1.7.3 Le problème d'exclusion mutuelle dans les systèmes répartis

Plusieurs algorithmes ont été proposés pour résoudre le problème de l'exclusion mutuelle dans les systèmes centralisés, on cite les sémaphores de Dijkstra[5] et les moniteurs. Et avec l'évolution

des systèmes informatiques il apparait un nouveau problème qui est "le problème d'exclusion mutuelle dans les systèmes réparti". La première définition du problème de l'exclusion mutuelle a été donnée par Dijkstra[5]. On peut résumer ses propositions dans les cinq assertions suivantes :

1. A un moment donné, on trouve qu'un seul processus dans la section critique.
2. La solution doit être symétrique, c'est-à-dire que l'on ne "peut pas introduire de priorité statique".
3. On ne peut pas prédire la vitesse des participants.
4. Tout site à l'état dehors peut quitter le système sans qu'il bloque les autres.
5. Si il y a plusieurs site veulent entrer en section critique, donc on doit décider à un temps fini quel site va entrer en section critique.

Cette définition est valable jusqu'à aujourd'hui, et on la retrouve dans les principales propriétés des algorithmes d'exclusion mutuelle.

1.7.4 Propriétés d'un algorithme d'exclusion mutuelle

Il faut que l'algorithme respecte les propriétés suivantes pour avoir une solution correcte :

— **La sureté** (Safety) : à n'importe quel moment on trouve qu'un seul site qui exécute la SC.
— **La vivacité** (Liveness) : il faut garantir que tous les sites à l'état demandeur acquérir la ressource critique dans un temps fini.

Il doit aussi assurer l'absence de ces deux problèmes :

— **L'interblocage** (Deadlock) :aura lieu quand il y a plusieurs sites à l'état demandeur chacun empêche l'autre à accéder à une ressource critique, donc aucun site ne peut obtenir la ressource.
— **La famine** (Starvation) :c'est lorsqu'un site à l'état demandeur n'entre jamais dans la SC.

1.7.5 Les classes de solution de l'exclusion mutuelle

Il sont classés en deux classes :

Les algorithmes à permissions :

l'idée de l'algorithme à permission est que chaque site qui veut entrer en SC doit demander la permission des autres sites, donc il ne peut pas entrer en SC qu'à la réception de l'autorisation de tous les autres sites.

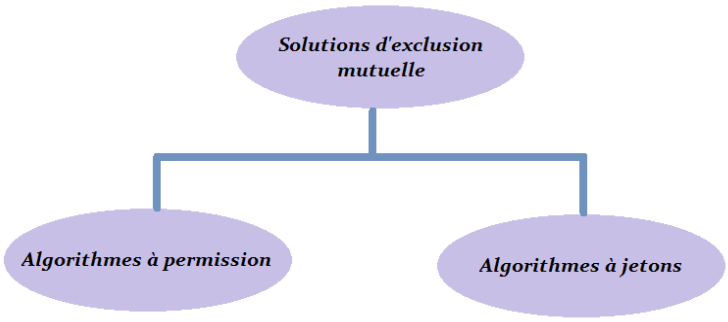


FIGURE 1.3 – Les classes d'algorithmes de l'exclusion mutuelle.

Les algorithmes à jeton :

Le jeton est considérer comme une autorisation, donc à la possession du jeton le site peut entrer en SC.

1.8 Le problème de k-exclusion mutuelle

1.8.1 Description et résolution du problème

Contrairement à l'exclusion mutuelle, dans k-exclusion mutuelle le processus a le choix d'utiliser k exemplaires de la même ressource, mais à condition un processus doit accéder seulement à un seul exemplaire.

Les solutions du problème de la k-exclusion mutuelle sont basées sur les solutions utilisées pour résoudre le problème de l'exclusion mutuelle, donc on a deux types : celle qui utilise la permission et celle qui utilise le jeton, la première solution qui existe est l'algorithme de Raymond, et parmi les solutions basées sur les jetons on cite l'algorithme de Srimani et Reddy[11].

1.9 Le problème de l'exclusion mutuelle dans le réseau mobile ad hoc

Les solutions proposées au problème de l'exclusion mutuelle sont classées en deux catégories, les solutions basées sur les permissions et celles qui utilisent les jetons, ces solutions doivent respecter les caractéristiques des MANETs telle que la mobilité etc.

Parmi les solutions proposées est l'algorithme de J.Walter et S.kini[15], et il a été modifier par J.walter et J.welch et vaidya[16], plusieurs algorithmes sont basés sur cet algorithme.

1.10 Le problème de la k-exclusion mutuelle dans le réseau mobile ad hoc

Ici les processus partagent k exemplaires de la même ressource. Plusieurs algorithmes sont proposés dans [15] traitent le problèmes de la k-exclusion mutuelle.

1.11 Conclusion

Dans ce chapitre on a vu le concept de MANET, ses caractéristique,et aussi on a vu que le réseau mobile ad hoc possède assez d'avantages par rapport aux autres réseaux par sa facilité de déploiement en cas d'urgence ou de travaux temporaires dont les autres réseaux engendrent des frais importants. Mais en même temps plusieurs problèmes existent, qui donnent l'occasion aux chercheurs de faire des nouvelles recherches sur les MANETs.

On a vu aussi quelques notions sur l'exclusion mutuelle, les propriétés de l'algorithme d'exclusion mutuelle et les classes de solutions d'exclusion mutuelle, et on a décrit le problème de l'exclusion mutuelle et la k-exclusion mutuelle dans les systèmes distribués et ainsi dans les réseaux mobiles ad hoc.

Dans le prochain chapitre, on va présenter et simuler une nouvelle technique de choix de leaders dans un algorithme de partage de ressources dans les réseaux mobiles ad.

Chapitre 2

Nouvelle technique de choix de leaders dans un algorithme de partage de ressources dans les MANET

2.1 Introduction

Les réseaux mobile ad hoc sont distincts du systèmes distribués traditionnels, ils sont des réseaux dynamique et auto-organisés. Contrairement aux systèmes distribués traditionnels, les noeuds sont mobiles ils entrent et quittent le réseau fréquemment, et ils ont une ressource d'énergie très limitée ce qu'il faut y avoir des protocoles trop spécifiques tels que les protocoles de routage, et de sélection de leaders ...etc.

Un leader dans un réseau mobile ad hoc fait un travail particulier ce qu'il nécessite plus d'énergie par rapport aux autres noeuds, donc il faut partager cette période(leader) entre les noeuds pour allonger la durée de vie du réseau.

Dans ce chapitre, on va proposer et simuler un nouvelle technique de choix de leaders dans un algorithme de partage de ressource dans les réseaux mobiles ad hoc qui assure l'équilibrage de charge entre les noeuds.

2.2 Le principe de l'algorithme proposé dans [1]

Le but de cet algorithme de partage de ressources est d'assurer la K-exclusion mutuelle par l'échange d'un nombre réduit de messages. Donc l'idée est d'ajouter deux champs dynamiques dans la table de routage de chaque noeud qui sont jeton libre et jeton présent. A chaque modification au niveau de ces deux variables, les noeuds doivent modifier leurs tables de routage grâce aux messages de contrôle utilisés pour le routage afin de minimiser le nombre de messages échangés.

2.3 Le principe de l'algorithme

Notre algorithme traite le problème de la k-exclusion mutuel, il assure le partage de k ressources entre les noeuds, quand un noeud détient une ressource critique(imprimante, ...) on le considère comme un leader. L'objectif de notre algorithme est non pas que traiter le problème de k-exclusion mutuelle mais on essaye d'allonger la durée de vie de réseau et aussi en conservant l'équilibrage de charge dans le réseau.

La nouvelle idée ajoutée à l'algorithme proposé dans [1] est d'ajouter un nouveau champ dans la table de routage à coté des deux autres champs (jeton libre,jeton présent), c'est l'énergie d'un noeud, et on combine le facteur de l'énergie et la distance lors de la sélection d'un leader. La présence d'un jeton libre veut dire que le leader détient la ressource mais ne l'utilise pas, et la présence d'un jeton présent veut dire qu'il a entrain d'utiliser la ressource.

Lorsqu'un noeud veut utiliser une ressource critique il va consulter sa table de routage afin de sélectionner le leader qui a un rapport $\frac{Energie}{Distance}$ le plus grand, et s'il trouve un noeud qui a une énergie inférieure à un certain seuil minimal il la choisit pour ne pas perdre le jeton.

La table de routage est comme suit :

2.3.1 Hypothèses

On suppose que :
- Le nombre de noeuds est connu, ainsi que le nombre de ressources.
- Les canaux de communication sont fiables.
- Le réseau n'est pas partitionné.
- Les noeuds ne tombent pas en panne.

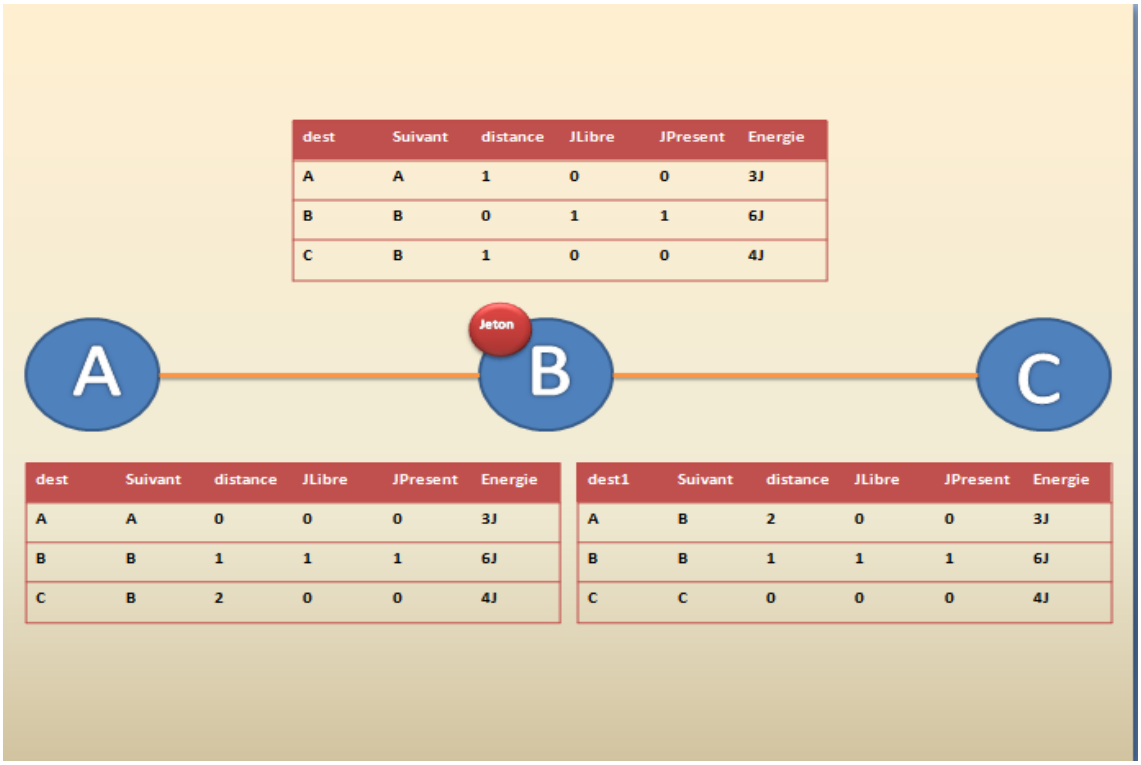


FIGURE 2.1 – La forme de la table de routage

2.4 Le nouvel algorithme

2.4.1 Les variables locales

Chaque noeud dans le réseau manipule les variables suivantes :

- $Demandeur_i$:quand elle est égale à vrai, il signifie que le noeud i a demandé une ressource critique.
- $jeton_i$:elle égale à vrai quand le noeud i possède un jeton.
- $ListeDemandeur_i$:c'est une file qui contient les identités des noeuds qui ont demandé une ressource critique.
- $AODV_i(jetonLibre, jetonPresent, Energie)$:c'est les informations que le noeud va les utiliser lors de la sélection d'un leader.

2.4.2 Les messages utilisés

- **Requête(jeton,i)** :envoyé par le noeud i pour demander une ressource.
- **Accord (jeton,Liste)** :envoyé à noeud demandeur pour lui donner l'autorisation pour utiliser la ressource critique.
- **Rejet (j)** :envoyé par le noeud j pour dire au demandeur qu'il a déjà donné la ressource à un autre noeud.

2.4.3 Les fonctions de l'algorithme

On a deux types de noeuds :

1. **Un noeud leader.**
2. **Un noeud simple.**

Algorithm 1 Initialisation du noeuds simples

$Demandeur_i \leftarrow faux$;
$ListeDemandeur_i \leftarrow NIL$;
$jeton \leftarrow faux$;
$AODV_i[i].jetonLibre \leftarrow 0$;
$AODV_i[i].jetonPresent \leftarrow 0$;

Algorithm 2 Initialisation du noeuds leaders

$Demandeur_i \leftarrow faux$;
$ListeDemandeur_i \leftarrow NIL$;
$jeton \leftarrow faux$;
$AODV_i[i].jetonLibre \leftarrow 1$;
$AODV_i[i].jetonPresent \leftarrow 1$;

Un noeud qui veut utiliser une ressource critique va lancer la fonction **Demander une ressource critique**, il sélectionne le leader adéquat et il demande à lui un jeton pour utiliser la ressource quand cette dernière devient libre.

Algorithm 3 Demander une ressource critique

$Demandeur_i \leftarrow vrai$;
if $AODV_i[i].jetonLibre = 0$ **then**
  $leader \leftarrow Chercher - detenant - adequat$;
  $Envoyer$**Requete(jeton,i)** à $leader$;
end if
$Attendre(jeton_i = vrai)$;
$AODV_i[i].jetonLibre \leftarrow 0$;
$AODV_i[i].jetonPresent \leftarrow 1$;
Utiliser la ressource critique ;

La procédure **Chercher le leader adéquat** montre une nouvelle technique pour sélectionner le leader adéquat, ici le choix est basé sur le leader qui a le plus grand rapport $\frac{Energie}{Distance}$ ou bien celui qui a un niveau d'énergie le plus bas.

Algorithm 4 Chercher le leader adéquat

```
RapportMax ← 0;
permis ← vrai;
EnergieInitiale ← 6;
EnergieMin ← 1000;
Min ← 0;
for j=0 a nombre de noeuds do
  if AODV_i[i].jetonLibre = 1 then
    distance ← AODV_i[j].Distance;
    energie ← AODV_i[j].Energie;
    Rapport ← energie/distance;
    if Rapport > RapportMax et permis then
      RapportMax ← Rapport;
      leader ← j;
      Min ← distance;
    else
      if energie < 0.4 * EnergieInitiale et energie < EnergieMin then
        leader ← j;
        EnergieMin ← energie;
        Min ← distance;
        permis ← faux;
      end if
    end if
  end if
end for
if Min = 0 then
  for j=0 a nombre de noeuds do
    if AODV_i[i].jetonPresent = 1 then
      distance ← AODV_i[j].Distance;
      energie ← AODV_i[j].Energie;
      Rapport ← energie/distance;
      if Rapport > RapportMax et permis then
        RapportMax ← Rapport;
        leader ← j;
        Min ← distance;
      end if
    end if
  end for
end if
return leader;
```

Quand un noeud reçoit un message **Requête(jeton,j)** il vérifier s'il possède un jeton libre, si oui il envoie un **Accord(jeton,liste)** au noeud demandeur. S'il est entrain d'utiliser la ressource critique (c-à-d il possède un jeton present) ici il met le noeud demandeur en attente. S'il n'a pas aucun jeton il le répond par un **Rejet(j)**.

Algorithm 5 Réception d'une Requête(jeton,j)

if $AODV_i[i].jetonLibre = 1$ **then**
  $jeton_i \leftarrow faux$;
  *Envoyer***Accord(jeton,Liste)**aj;
  $AODV_i[i].jetonLibre \leftarrow 0$;
  $AODV_i[i].jetonPresent \leftarrow 0$;
else
  if $AODV_i[i].jetonPresent = 1$ **then**
    $ListeDemandeurs_i \leftarrow ListeDemandeurs_i + j$;
    $Liste \leftarrow ListeDemandeurs_i$;
  else
    *Envoyer***Rejet(i)** à j;
  end if
end if

A la réception d'un message **Rejet(j)** par un noeud demandeur il met à jour sa table de routage et sélectionne un nouveau leader pour demander un jeton.

Algorithm 6 Réception d'un Rejet(j)

$AODV_i[j].jetonLibre \leftarrow 0$;
$AODV_i[j].jetonPresent \leftarrow 0$;
$leader \leftarrow Chercher - detenant - adequat(AODV_i)$;
*Envoyer***Requete(jeton,i)** à $leader$;

Quand un noeud reçoit un message **Accord(jeton,Liste)**, il modifie la variable jeton à vrai, et met à jour sa liste de demandeur.

Algorithm 7 Réception d'un Accord(j,Liste)

$jeton_i \leftarrow vrai$;
$ListeDemandeurs_i \leftarrow Liste$;

A la libération de la ressource critique le noeud vérifie sa liste de demandeur, s'il trouve pleine il sélectionne un nouveau leader adéquat afin d'envoyer à lui le jeton pour utiliser la ressource, sinon le jeton reste détenu par ce noeud.

Algorithm 8 Libérer une ressource critique

$Demandeur_i \leftarrow faux$;
if $Tete(ListeDemandeurs_i) \neq NIL$ **then**
  while $Tete(ListeDemandeurs_i) \neq NIL$ **do**
    $distance \leftarrow AODV_i[Tete(ListeDemandeurs_i)].Distance$;
    $energie \leftarrow AODV_i[Tete(ListeDemandeurs_i)].Energie$;
    $Rapport \leftarrow energie/distance$;
    if $Rapport > RapportMax$ **then**
      $RapportMax \leftarrow Rapport$;
      $d \leftarrow Tete(ListeDemandeurs_i)$;
      $Min \leftarrow distance$;
    end if
  end while
  $Liste \leftarrow ListeDemandeurs_i$;
  $Envoyer$**Accord(jeton,Liste)** à d;
  $jeton_i \leftarrow faux$;
else
  $AODV_i[i].jetonLibre \leftarrow 1$;
  $AODV_i[i].jetonPresent \leftarrow 1$;
end if

2.5 Évaluation des performances de notre algorithme

2.5.1 Les paramètres d'évaluation

Le temps d'attente moyen (TAM) : C'est la somme des temps attendus par chaque noeud avant de satisfaire leurs demande pour utiliser les ressources sur le nombre de demandes.
$\frac{\sum TempsD'attente}{NombreDeRequetes}$

Le nombre de messages moyen (NMM) : c'est la somme de messages envoyés par chaque noeud lors de la demande d'une ressource sur le nombre total des requêtes.
$\frac{NombreDeMessageTotal}{NombreDeRequetes}$

Le temps d'utilisation d'énergie moyen (TUM) : C'est la somme de taux d'énergie utilisée pour chaque noeud sur le nombre de noeuds.
$\frac{\sum TauxD'utilisationD'energie}{NombreDeDoeuds}$

L'écart-type de TUM : $\sqrt{\frac{(\sum TauxD'utilisationD'energie_i - TUM)^2}{nombreDeNoeuds}}$

2.5.2 Scénarios et paramètres de simulations

Pour analyser les résultats de simulation de notre nouvel algorithme et les comparer avec l'ancien algorithme, on va simuler quatre scénarios et dans chaque scénario on varie la valeur d'un paramètre.

Le tableau 2.1 résume ces quatre scénarios.

	Nombre de requêtes	Nombre de ressources	Nombre de noeuds	La portée de communication
Scen1	$3 <= A <= 20$	5	50	200
Scen2	20	$3 <= B <= 15$	50	200
Scen3	20	5	$20 <= C <= 50$	200
Scen4	20	5	50	$100 <= D <= 500$

TABLE 2.1 – Les différents scénarios utilisés dans la simulation.

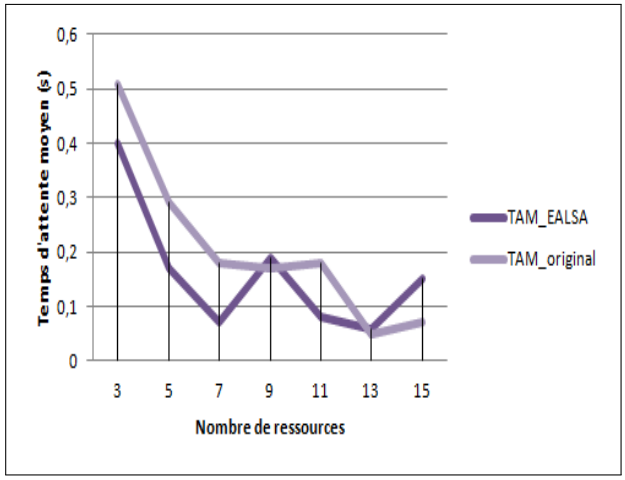
2.5.3 Analyse de résultats de simulation

Les graphes suivants résument une comparaison entre les résultats de simulation de l'ancien algorithme et le nouvel algorithme qu'on a proposé.

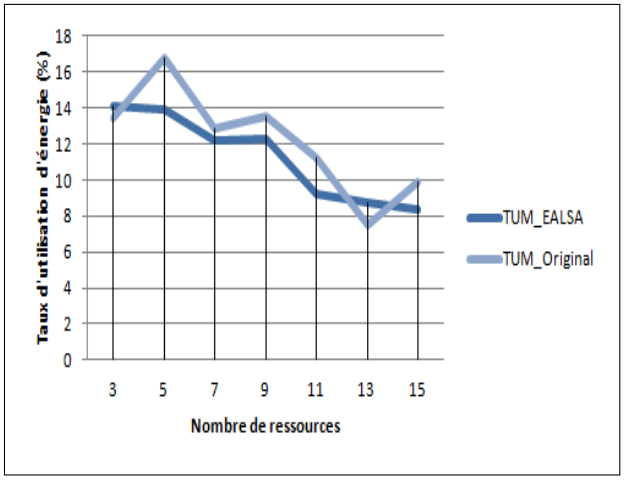
Variation du nombre de ressources On remarque que le TAM et le NMM pour les deux courbe diminue, c'est à cause de la disponibilité de ressources, ce qu'il influe aussi au TUM.

On remarque dans la figure 2.2d que l'équilibrage de charge dans le nouvel algorithme entre les noeuds est meilleur que l'ancien parce le choix d'un leader ne dépend de l'énergie.

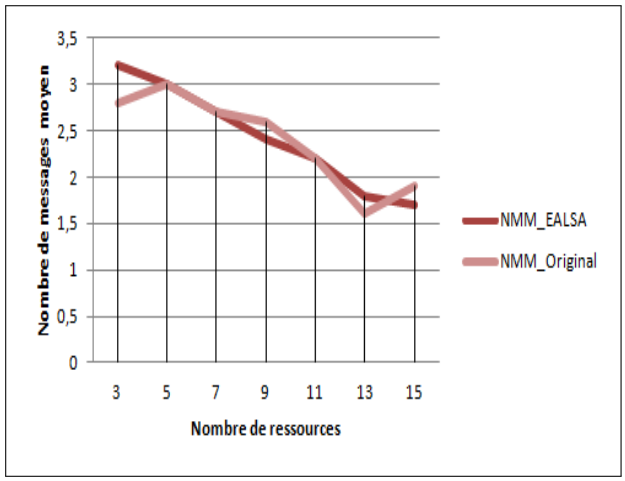
(a)



(b)



(c)



(d)

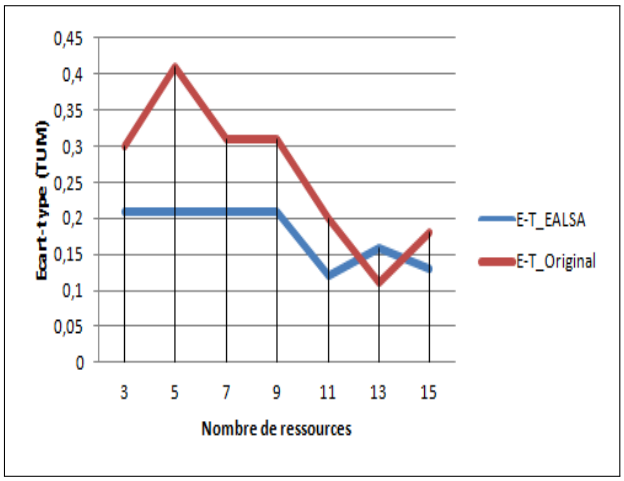


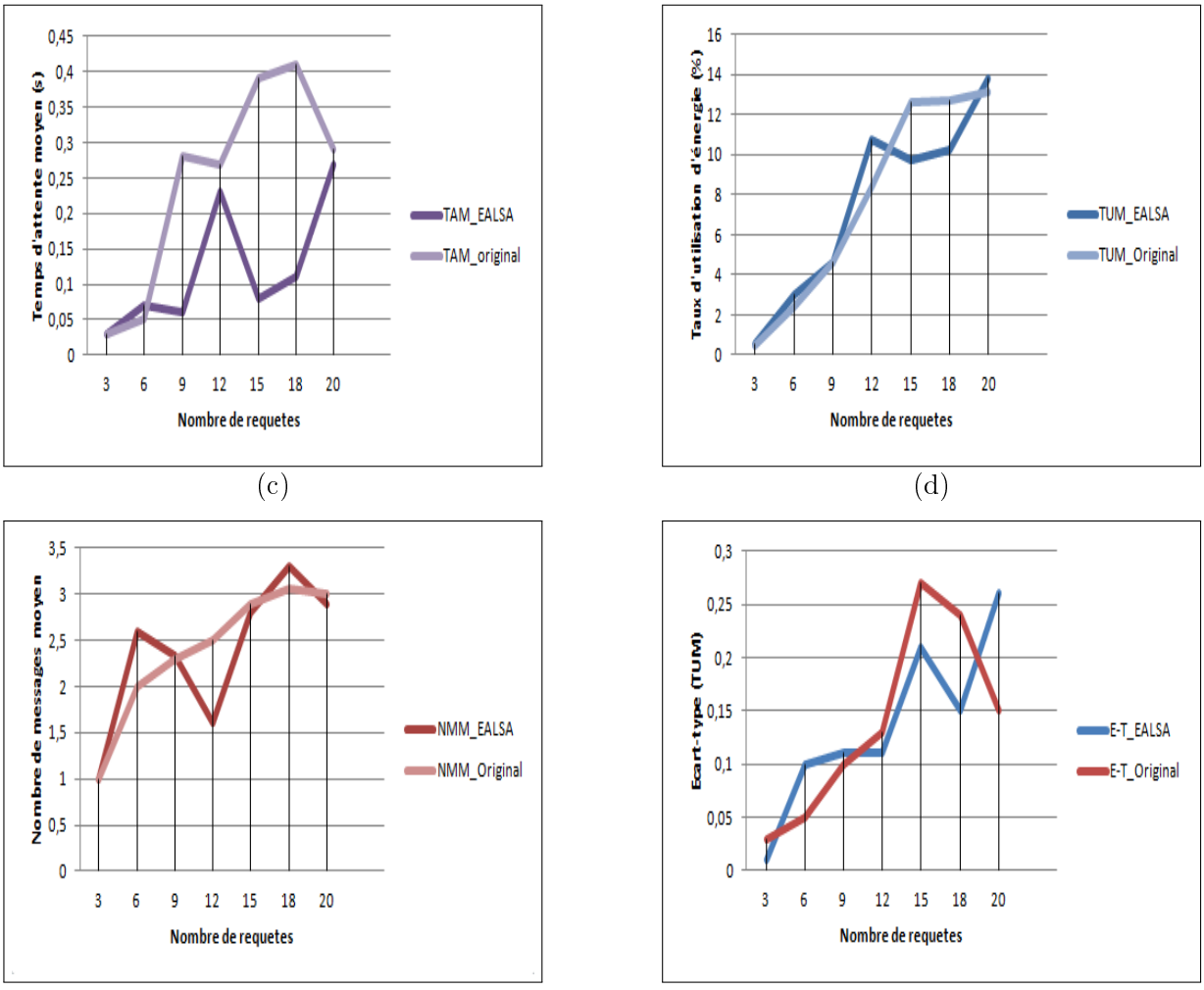
FIGURE 2.2 – Influence du nombre de ressources sur TAM, NMM, TUM, et l'écart-type du TUM

Variation du nombre de requêtes Pour le NMM on remarque dans la figure 2.3c qu'il y a une augmentation pour les deux courbe à cause de l'augmentation du nombre de requêtes, mais on trouve que la courbe de l'ancien algorithme a un NMM inférieur par rapport au nouvel, c'est tout à fait normal parce que dans l'ancienne approche le noeud envoi une requête au noeud le plus proche.

Dans la figure 2.3b pour les deux courbes l'augmentation du TUM est presque linéaire, on justifié ça par l'augmentation du nombre de messages échangés.

On remarque dans la courbe 2.3d que l'équilibrage de charge entre les noeuds dans notre algorithme est meilleur lorsque le nombre de requêtes est supérieur à 12 requêtes, mais il est meilleur dans l'ancien algorithme quand le nombre de requêtes est inférieur à 12. On justifie ça, que la nouvelle technique utilisée pour demander une ressource a une influence positive que lorsque le nombre de requêtes est grand.

(a) (b)



(c) (d)

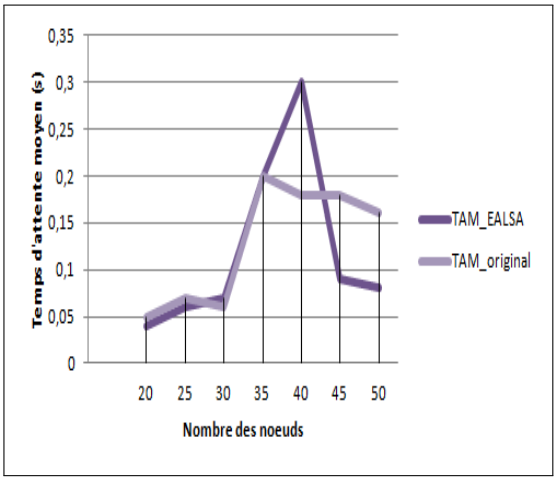
FIGURE 2.3 – Influence du nombre de requêtes sur TAM, NMM, TUM, et l'écart-type du TUM

Variation du nombre de noeuds On observe dans la figure 2.4a on justifie le TAM élevé quand le nombre de noeuds égale 40 par une pool de pannes.

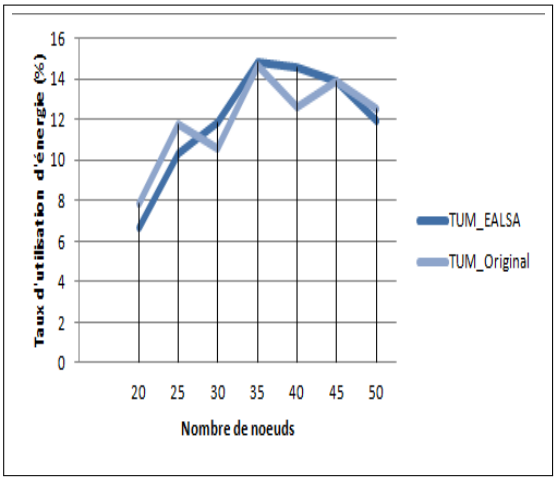
Dans la figure 2.4b le TUM dans les deux algorithmes augment avec l'augmentation du nombre de noeuds et c'est tout à fait normal parce qu'on aura plus de noeuds qui vont participer dans le transfert des paquets ce qu'il nécessite une énergie.

Dans la figure 2.4d on trouve que l'équilibrage de charge dans le nouvel l'algorithme est généralement bon en comparent avec l'ancien, tout simplement parce que notre choix est basé sur le rapport $\frac{Energie}{Distance}$.

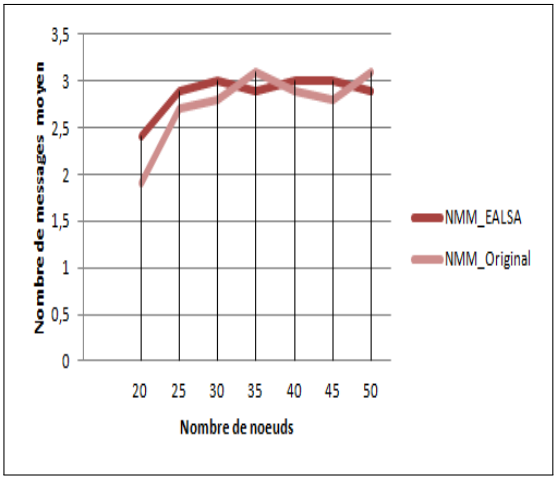
(a)



(b)



(c)



(d)

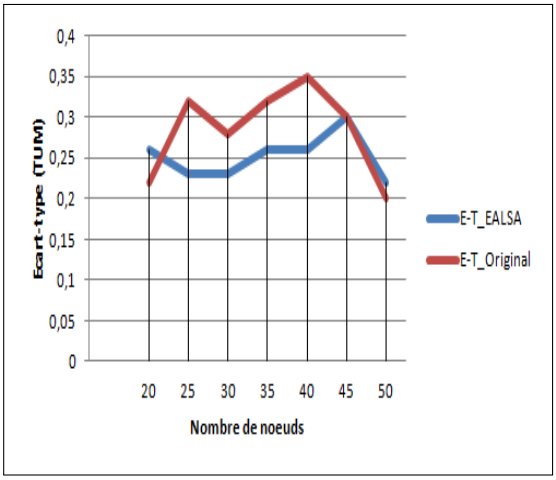


FIGURE 2.4 – Influence du nombre de noeuds sur TAM, NMM, TUM, et l'écart-type du TUM

Variation de la portée de communication Dans la figure 2.5a on remarque une diminution du TAM avec l'augmentation de la portée de communication dans la courbe de l'ancien algorithme parce que les noeuds ont de plus en plus des voisins directe donc moins de messages échangés ce qu'on le constate dans la figure 2.5c. Mais pour le nouvel algorithme le TAM est un peu fluctuant et le NMM reste stable lorsque la portée de communication est plus de 200 mètre.

Dans la figure 2.5d on trouve que l'équilibrage de charge est presque le même pour les deux approches, on justifie ça que la portée de communication n'a pas une influence.

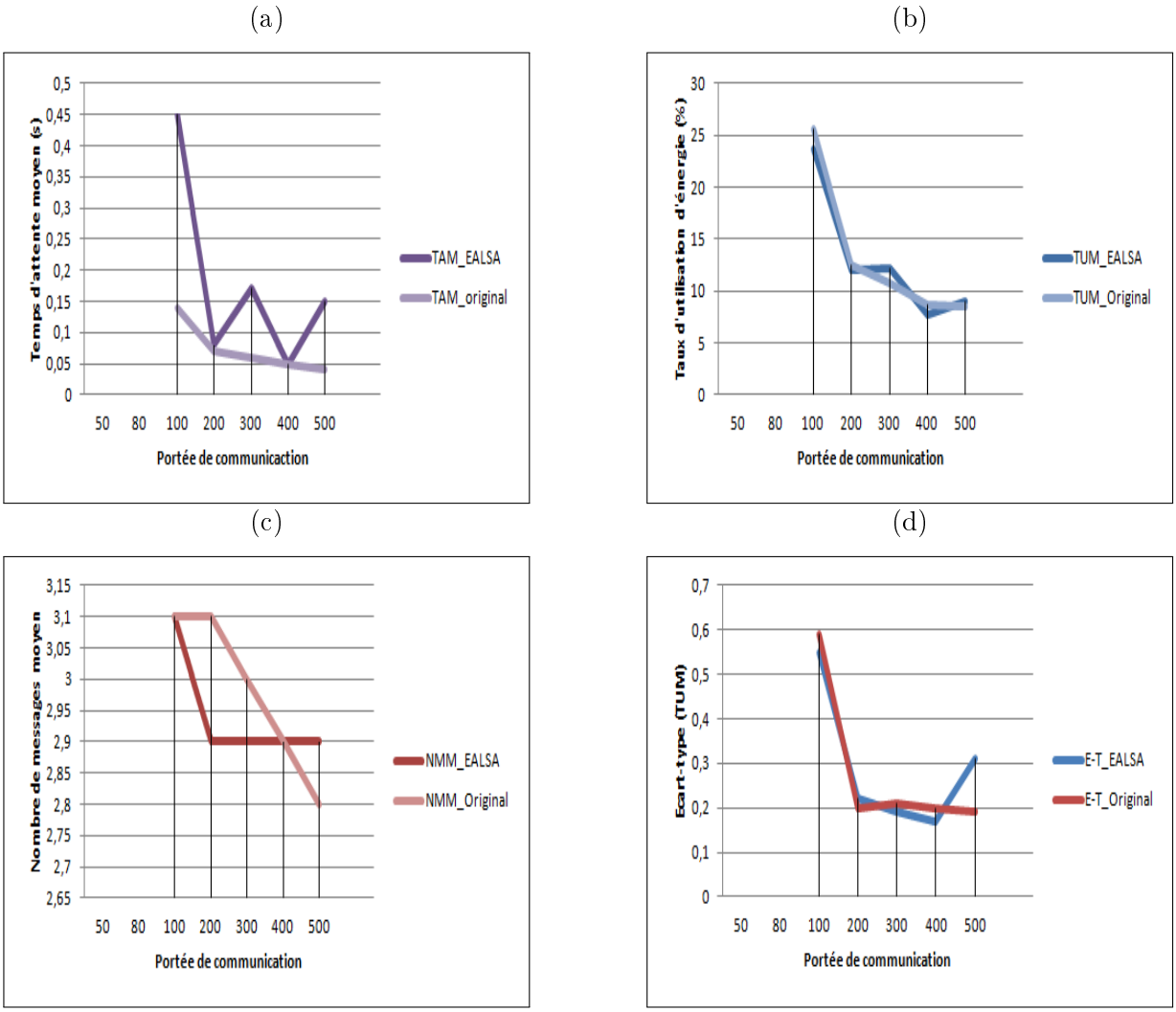


FIGURE 2.5 – Influence de la portée de communication sur TAM, NMM, TUM, et l'écart-type du TUM

2.6 Conclusion

Après voir la comparaison entre les résultats de simulation de notre nouvel algorithme et l'ancien algorithme, notre algorithme n'est pas toujours le meilleur mais pour l'équilibrage de charge est généralement le meilleur par rapport à l'ancienne approche, donc on peut dire que on a atteint notre objectif.

Chapitre 3

Nouvelle technique de choix de leaders dans un algorithme de partage de ressources dans les MANET tolérant aux pannes

3.1 Introduction

La tolérance aux pannes (on dit également « insensibilité aux pannes ») désigne une méthode de conception permettant à un système de continuer à fonctionner, éventuellement de manière réduite (on dit aussi en « mode dégradé »), au lieu de tomber complètement en panne, lorsque l'un de ses composants ne fonctionne plus correctement.[17]
l'inclusion d'un mécanisme de détection de pannes est importante lors d'une conception d'un nouveau protocole pour les réseaux mobile ad hoc, mais en réalité c'est difficile de l'implémenter parce que la plupart de types de pannes on peut pas les détecter à la préalable.

Dans ce chapitre on va présenter Nouvelle technique de choix de leaders dans un algorithme de partage de ressources tolérant aux pannes et assure l'équilibrage de charge entre les noeuds dans les réseaux mobile ad hoc.

3.2 Le principe de l'algorithme

L'idée ajoutée à l'algorithme expliqué dans le chapitre précédent pour le mettre tolérant aux pannes est qu'on a utilisé l'une des fameuses méthodes de tolérance aux pannes c'est la duplication des données.
On a ajouté un noeud duplicata du noeud leader **(D)** parce que ce type de noeuds jouent un rôle très important dans le réseau leur échec va conduire la perte des jetons et les listes des demandeurs, on l'a appelé un noeud sûr **(S)**.
L'ajout d'un noeud sûr ne suffit pas, parce que son échec va conduire aussi la perte de jeton, donc on a ajouté un autre noeud back-to-back **(B)**, ce noeud n'intervient que lorsque la panne atteint le **S**.
Dans notre algorithme on suppose que la panne est détectée à la préalable, ça veut dire avant que le noeud tombe complètement en panne à cause de déchargement de son batterie.

3.2.1 Hypothèses

On suppose que :
— Le nombre de noeuds est connu, ainsi que le nombre de ressources.
— Les canaux de communication sont fiables.
— Le réseau n'est pas partitionné.
— Les noeuds ne tombent à cause de perte d'énergie.

3.2.2 Le comportement de l'algorithme lors d'une panne

La panne peut atteindre qu'un seul type de noeud, soient D,S ou B.

La panne d'un leader : Il existe deux cas :

1. La liste des demandeurs est pleine comme il est illustré dans les figure 3.1a et 3.1b.
   — Le noeud S donne le jeton au noeud adéquat.
   — Il envoie ue mise à jour au noeud B.
2. La liste des demandeurs est vide comme il est illustré dans les figure 3.1c et 3.1d.
   — Le noeud S garde le jeton, et il considère lui-même comme un noeud D.
   — Il met l'ancien B comme un noeud S, et sélectionne un nouveau B.
   — Il informe les deux noeuds de la nouvelle mise à jour.

FIGURE 3.1 – Comportement de l'algorithme lors d'une panne de leader.

La panne d'un noeud S :

- Le noeud S demande l'intervention du noeud B.
- Le noeud B met lui-même comme noeud S, et il sélectionne un nouveau B.
- Il informe le leader et le nouveau B de la nouvelle mise à jour.

(a) (b)

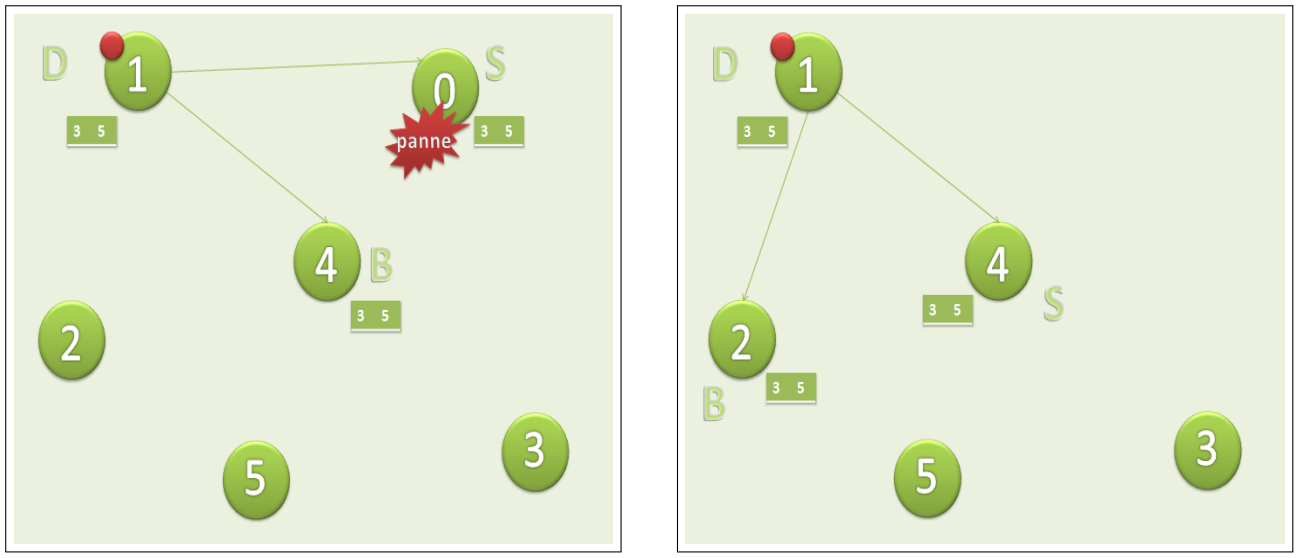


FIGURE 3.2 – Comportement de l'algorithme lors d'une panne de noeud S.

La panne d'un noeud B :

- Le noeud B demande l'intervention du noeud S.
- Le noeud S sélectionne un nouveau B.
- Le noeud S informe le nouveau B et le noeud D (leader).

(a) (b)

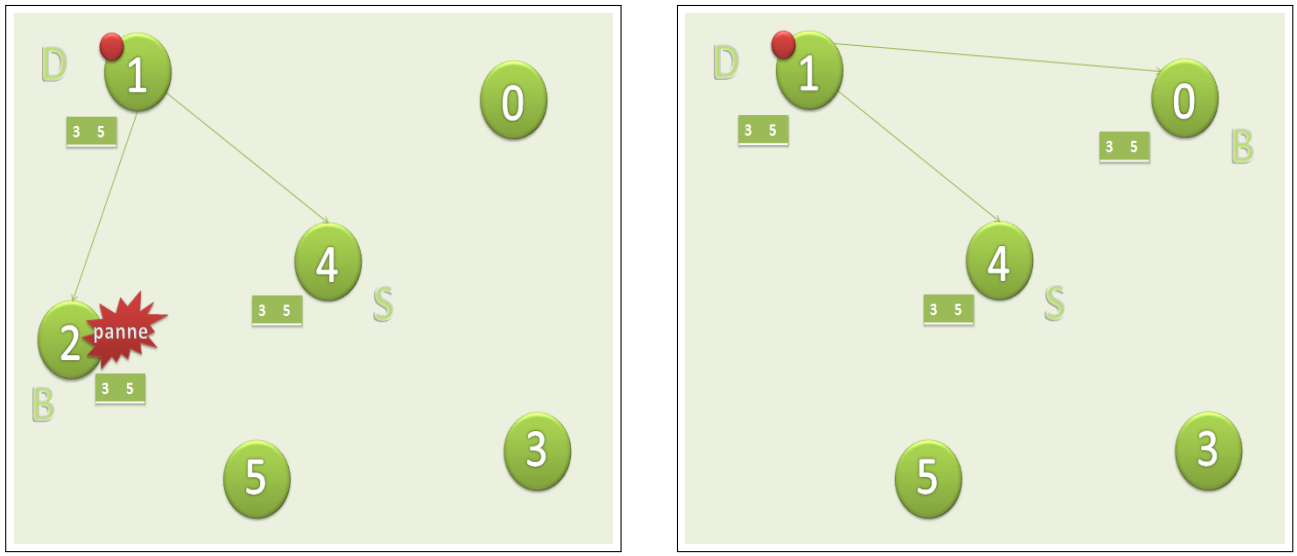


FIGURE 3.3 – Comportement de l'algorithme lors d'une panne de noeud B.

3.3 Le nouvel algorithme

3.3.1 Les variables locales

Chaque noeud dans le réseau manipule les variables suivantes :

- $Demandeur_i$:quand elle est égale à vrai, il signifie que le noeud i a demandé une ressource critique.
- $jeton_i$: elle égale à vrai quand le noeud i possède un jeton.
- $Enpanne_i$: indique que le noeud est en panne si elle est égale à vrai.
- $ListeDemandeurs_i$:c'est une file qui contient les identités des noeuds qui ont demandé une ressource critique.
- $Tolerance_i$:c'est une file duplicata du $ListeDemandeurs_i$, utilisé lorsque le noeud joue le rôle d'un noeud S ou B.
- idD : l'identité du leader.
- idS : l'identité du noeud sure.
- idB : l'identité du noeud Back-to-Back.
- $AODV_i(jetonLibre, jetonPresent, Energie)$:c'est les informations que le noeud va les utiliser lors de la sélection d'un leader.
- $AODV_i(S, B, EnPanne)$: c'est les informations qu'on va les utiliser lors d'une panne.

3.3.2 Les messages utilisés

- **Requête(jeton,i)** :envoyée par le noeud i pour demander une ressource.
- **Accord (jeton,Liste)** :envoyée à noeud demandeur pour lui donner l'autorisation pour utiliser la ressource critique.
- **Rejet (j)** :envoyée par le noeud j pour dire au demandeur qu'il a déjà donné la ressource à un autre noeud.
- **Panne(i)** : envoyé par un noeud lors de détection d'une panne.
- **MAJ-tolerance(Liste)** :envoyé par un leader à son noeud S quand il reçoit une nouvelle demande, et ce dernier à son tour le renvoie au noeud B.
- **MAJ-SB(d,s,b)** :envoyé par les noeuds D,S ou bien B lors d'une mise à jour.

3.3.3 Les fonctions de l'algorithme

On a deux types de noeuds :

1. **Un noeud leader.**
2. **Un noeud simple.**

Algorithm 9 Initialisation du noeuds simples

$Demandeur_i \leftarrow faux$;
$ListeDemandeur_i \leftarrow NIL$;
$Tolerance_i \leftarrow NIL$;
$jeton \leftarrow faux$;
$EnPanne_i \leftarrow faux$;
$AODV_i[i].S \leftarrow -1$;
$AODV_i[i].B \leftarrow -1$;
$AODV_i[i].jetonLibre \leftarrow 0$;
$AODV_i[i].jetonPresent \leftarrow 0$;
$AODV_i[i].EnPanne \leftarrow 0$;

Algorithm 10 Initialisation du noeuds leaders

$Demandeur_i \leftarrow faux$;
$ListeDemandeur_i \leftarrow NIL$;
$Tolerance_i \leftarrow NIL$;
$jeton \leftarrow faux$;
$EnPanne_i \leftarrow faux$;
$S \leftarrow ChoisirS(i)$;
$B \leftarrow ChoisirB(i)$;
$AODV_i[i].S \leftarrow S$;
$AODV_i[i].B \leftarrow B$;
$AODV_i[i].jetonLibre \leftarrow 1$;
$AODV_i[i].jetonPresent \leftarrow 1$;
$AODV_i[i].EnPanne \leftarrow 0$;
Envoyer **MAJ-SB**(i, S, B) à S;
Envoyer **MAJ-SB**(i, S, B) à B;

Un noeud qui veut utiliser une ressource critique va lancer la fonction **Demander une ressource critique**, il sélectionne le leader adéquat et il demande à lui un jeton pour utiliser la ressource quand cette dernière devient libre.

Algorithm 11 Demander une ressource critique

$Demandeur_i \leftarrow vrai$;
if $AODV_i[i].jetonLibre = 0$ **then**
  $leader \leftarrow Chercher - detenant - adequat$;
  Envoyer **Requete(jeton,i)** à $leader$;
end if
$Attendre(jeton_i = vrai)$;
$AODV_i[i].jetonLibre \leftarrow 0$;
$AODV_i[i].jetonPresent \leftarrow 1$;
Utiliser la ressource critique ;

La procédure **Chercher le leader adéquat** montre une nouvelle technique pour sélectionner le leader adéquat, ici le choix est basé sur le leader qui a le plus grand rapport $\frac{Energie}{Distance}$ ou bien celui qui a un niveau d'énergie le plus bas.

Algorithm 12 Chercher le leader adéquat

```
RapportMax ← 0;
permis ← vrai;
EnergieInitiale ← 6;
EnergieMin ← 1000;
Min ← 0;
for j=0 a nombre de noeuds do
  if AODV_i[i] − jeton − libre = 1 then
    distance ← AODV_i[j].Distance;
    energie ← AODV_i[j].Energie;
    Rapport ← energie/distance;
    if Rapport > RapportMax et permis then
      RapportMax ← Rapport;
      leader ← j;
      Min ← distance;
    else
      if energie < 0.5 et energie < EnergieMin then
        leader ← j;
        EnergieMin ← energie;
        Min ← distance;
        permis ← faux;
      end if
    end if
  end if
end for
if Min = 0 then
  for j=0 a nombre de noeuds do
    if AODV_i[i].jetonPresent = 1 then
      distance ← AODV_i[j].Distance;
      energie ← AODV_i[j].Energie;
      Rapport ← energie/distance;
      if Rapport > RapportMax et permis then
        RapportMax ← Rapport;
        leader ← j;
        Min ← distance;
      else
        if energie < 0.5 et energie < EnergieMin then
          leader ← j;
          EnergieMin ← energie;
          Min ← distance;
          permis ← faux;
        end if
      end if
    end if
  end for
```

Quand un noeud reçoit un message **Requête(jeton,j)** il vérifier s'il possède un jeton libre, si oui il envoie un **Accord(jeton,liste)** au noeud demandeur. S'il est entrain d'utiliser la ressource critique (c-à-d il possède un jeton présent) ici il met le noeud demandeur en attente. S'il n'a pas aucun jeton il le répond par un **Rejet(j)**.

Algorithm 13 Réception d'une Requête(jeton,j)

if $AODV_i[i].jetonLibre = 1$ **then**
  $jeton_i \leftarrow faux$;
  Envoyer **MAJ-SB**$(j, i, AODV_i[i].S)$ à $AODV_i[i].S$;
  Envoyer **MAJ-SB**$(-1, -1, -1)$ à $AODV_i[i].B$;
  $idB_i \leftarrow AODV_i[i].S$;
  $idS_i \leftarrow i$;
  $idD_i \leftarrow j$;
  $Liste \leftarrow ListeDemandeur_i$;
  *Envoyer***Accord**$(jeton, Liste)$ à j;
  Envoyer **MAJ-SB**(idD, idS, idB) à j;
  $AODV_i[i].jetonLibre \leftarrow 0$;
  $AODV_i[i].jetonPresent \leftarrow 0$;
  $AODV_i[i].S \leftarrow -2$;
  $AODV_i[i].B \leftarrow -1$;
else
  if $AODV_i[i].jetonPresent = 1$ **then**
    $ListeDemandeurs_i \leftarrow ListeDemandeurs_i + j$;
    $Liste \leftarrow ListeDemandeurs_i$;
    Envoyer **MAJ-tolerance**$(Liste)$ à $AODV_i[i].S$;
  else
    Envoyer **Rejet**(i) à j;
  end if
end if

A la réception d'un message **Rejet(j)** par un noeud demandeur il met à jour sa table de routage et sélectionne un nouveau leader pour demander un jeton.

Algorithm 14 Réception d'un Rejet(j)

$AODV_i[j].jetonLibre \leftarrow 0$;
$AODV_i[j].jetonPresent \leftarrow 0$;
$leader \leftarrow Chercher - detenant - adequat(AODV_i)$;
Envoyer **Requete**$(jeton, i)$ à $leader$;

Quand un noeud reçoit un message **Accord(jeton,Liste)**, il modifie la variable jeton à vrai, et met à jour sa liste de demandeur.

Algorithm 15 Réception d'un Accord(j,Liste)

$jeton_i \leftarrow vrai$;
$ListeDemandeurs_i \leftarrow Liste$;

Le message MAJ-tolerance est envoyé par le noeud D ou S lors d'une mise à jour de la liste des demandeurs.

Algorithm 16 Réception d'un MAJ-tolerance(Liste)

if $idS = i$ **then**
  $Tolerance_i \leftarrow Liste$;
  Envoyer **MAJ-tolerance**$(Liste)$ à idB;
else
  if $idB = i$ **then**
    $Tolerance_i \leftarrow Liste$;
  end if
end if

A la libération de la ressource critique le noeud vérifie sa liste de demandeur, s'il trouve pleine il sélectionne le premier demandeur afin d'envoyer à lui le jeton pour utiliser la ressource et il lui considère comme un noeud S du nouveau leader, l'ancien S va être le nouveau B, sinon le jeton reste détenu par ce noeud.

Algorithm 17 Libérer une ressource critique

$Demandeur_i \leftarrow faux$;
if $Tete(ListeDemandeurs_i) \neq NIL$ **then**
  $d \leftarrow Tete(ListeDemandeurs_i)$;
  $jeton_i \leftarrow faux$;
  Envoyer **MAJ-SB**$(d, i, AODV_i[i].S)$ à $AODV_i[i].S$;
  Envoyer **MAJ-SB**$(-1, -1, -1)$ à $AODV_i[i].B$;
  $idB_i \leftarrow AODV_i[i].S$;
  $idS_i \leftarrow i$;
  $idD_i \leftarrow d$;
  $Liste \leftarrow ListeDemandeur_i$;
  *Envoyer***Accord**$(jeton, Liste)$ à d;
  Envoyer **MAJ-SB**(idD, idS, idB) à d;
  $AODV_i[i].jetonLibre \leftarrow 0$;
  $AODV_i[i].jetonPresent \leftarrow 0$;
  $AODV_i[i].S \leftarrow -2$;
  $AODV_i[i].B \leftarrow -1$;
else
  $AODV_i[i].jetonLibre \leftarrow 1$;
  $AODV_i[i].jetonPresent \leftarrow 1$;
end if

Le message MAJ-SB est envoyé lors d'une mise à jour du l'un des sites D,S ou B.

Algorithm 18 Réception d'un MAJ-SB(d,s,b)

if $AODV_i[i].jetonPresent \neq 1$ **then**
  $idD \leftarrow d$;
  $idS \leftarrow s$;
  $idB \leftarrow b$;
  if $d = -1$ **then**
    $AODV_i[i].B \leftarrow -1$;
  else
    if $s = i$ **then**
      $AODV_i[i].S \leftarrow -2$;
      $AODV_i[i].B \leftarrow -1$;
    else
      $AODV_i[i].S \leftarrow -1$;
      $AODV_i[i].B \leftarrow -2$;
    end if
  end if
else
  $AODV_i[i].S \leftarrow s$;
  $AODV_i[i].B \leftarrow b$;
end if

La fonction **ChoisirS(i)** appelé quand un noeud veut choisir un noeud sure(S). la stratégie de sélection d'un noeud sure est selon l'énergie, pour garantir de ne pas tomber en panne à cause du déchargement de la batterie.

Algorithm 19 ChoisirS(i)

```
Min ← 1000;
monS ← −1;
for j=0 a nombre de noeuds do
  if AODV_i[j].S = −1 et AODV_i[j].B = −1 then
    if AODV_i[j].jetonLibre ≠ 1 et AODV_i[j].jetonPresent ≠ 1 then
      if AODV_i[j] − EnPanne ≠ 1 then
        distance ← AODV_i[j].Distance;
        energie ← AODV_i[j].Energie;
        Rapport ← energie/distance;
        if Rapport > RapportMax et energie > 2 then
          RapportMax ← Rapport;
          monS ← j;
        end if
      end if
    end if
  end if
end for
if monS = −1 then
  return monS;
else
  ChoisirS(i);
end if
```

La fonction **ChoisirB(i)** appelé quand un noeud veut choisir un noeud B. la stratégie de sélection d'un noeud Back-to-Back est selon l'énergie, pour garantir de ne pas tomber en panne à cause du déchargement de la batterie.

Algorithm 20 ChoisirB(i)

```
Min ← 1000;
monB ← −1;
for j=0 a nombre de noeuds do
  if AODV_i[j].S = −1 et AODV_i[j] − B = −1 then
    if AODV_i[j].jetonLibre ≠ 1 et AODV_i[j].jetonPresent ≠ 1 then
      if AODV_i[j].EnPanne ≠ 1 then
        distance ← AODV_i[j].Distance;
        energie ← AODV_i[j].Energie;
        Rapport ← energie/distance;
        if Rapport > RapportMax et energie > 2 then
          RapportMax ← Rapport;
          monB ← j;
        end if
      end if
    end if
  end if
end for
if monB = −1 then
  return monB ;
else
  ChoisirB(i);
end if
```

Ce message est envoyé quand un détecte une panne, si le noeud est un leader il demande l'intervention de son **S** et la même chose quand le noeud est **B**, si il est un noeud **S** il envoie un message **Panne()** au **B**.

Algorithm 21 Demander l'intervention lors d'une détection d'une panne

```
Demandeur_i ← faux;
Enpanne_i ← vrai;
AODV_i[i].EnPanne ← 1;
if AODV_i[i].jetonPresent = 1 then
  Envoyer Panne(i) à AODV_i[i].S;
else
  if idS ≠ −1 ou idB ≠ −1 then
    if idS = i then
      Envoyer Panne(i) à idB;
    else
      Envoyer Panne(i) à idS;
    end if
  end if
end if
```

Cette procédure est appelée à la réception d'une panne, le noeud va réagir selon le type de noeud qui a demander l'intervention, tout est expliquer dans les figures 3.1, 3.2 et 3.3.

Algorithm 22 A la réception d'une panne(j)

```
if idS = i then
  if idD = j then
    if Toletance_i ≠ NIL then
      l ← Tete(Tolerance_i);
      Tolerance_i ← Tolerance_i − l;
      Envoyer Accord(jeton, Tolerance) à l;
      idD ← l;
      EnvoyerMAJ-tolerance(Tolerance_i) à idB;
      EnvoyerMAJ-SB(idD, idS, idB) à idB;
      EnvoyerMAJ-SB(idD, idS, idB) à idD;
    else
      jeton_i ← vrai;
      AODV_i[i].jetonLibre ← 1;
      AODV_i[i].jetonPresent ← 1;
      AODV_i[i].S ← idB;
      AODV_i[i].B ← ChoisirB(i);
      idD ← −1;
      idS ← −1;
      idB ← −1;
      EnvoyerMAJ-SB(i, AODV_i[i].S, AODV_i[i].B) à AODV_i[i].S;
      EnvoyerMAJ-SB(i, AODV_i[i].S, AODV_i[i].B) à AODV_i[i].B;
    end if
  else
    if idB = j then
      idB ← ChoisirB;
      EnvoyerMAJ-SB(idD, idS, idB) à idB;
      EnvoyerMAJ-SB(idD, idS, idB) à idD;
    end if
  end if
else
  if idB = i then
    idS ← idB;
    idB ← ChoisirB;
    EnvoyerMAJ-SB(idD, idS, idB) à idB;
    EnvoyerMAJ-SB(idD, idS, idB) à idD;
  end if
end if
```

3.4 Évaluation des performances de notre algorithme

3.4.1 Scénarios et paramètres de simulations

Pour analyser les résultats de simulation de notre nouvel algorithme et le comparer avec l'ancien algorithme, on va simuler quatre scénarios et dans chaque scénario on varie la valeur d'un paramètre.

Le tableau 3.1 résume ces quatre scénarios.

	Nombre de requêtes	Nombre de ressources	Nombre de noeuds	La portée de communication	Nombre de pannes
scen1	$3 <= A <= 20$	5	50	200	5
scen2	20	$3 <= B <= 15$	50	200	5
scen3	20	5	$20 <= C <= 50$	200	5
scen4	20	5	50	$100 <= D <= 500$	5
scen5	20	5	50	200	$3 <= E <= 11$

TABLE 3.1 – Les différents scénarios utilisés dans la simulation.

3.4.2 Analyse des résultats de simulation

Les graphes suivants résument une comparaison entre les résultats de simulation de l'ancien algorithme et le nouvel algorithme qu'on a proposé.

Variation du nombre de ressources On observe généralement que le TAM diminue pour les deux algorithme, ce qu'il dit qu'ils ont bien réagis avec les pannes des noeuds. On remarque que le NMM des deux algorithmes est élevé, et il diminue progressivement avec l'augmentation du nombre de ressources. On justifie cette diminution progressive par le nombre de messages lors d'une panne. Dans la figure 3.4b on a pas une grande différence entre les deux courbes parce que le NMM est presque identique. Pour l'équilibrage de charge on constate une grande différence, notre algorithme est le meilleur c'est à cause de les technique utilisés lors de la sélection des leaders.

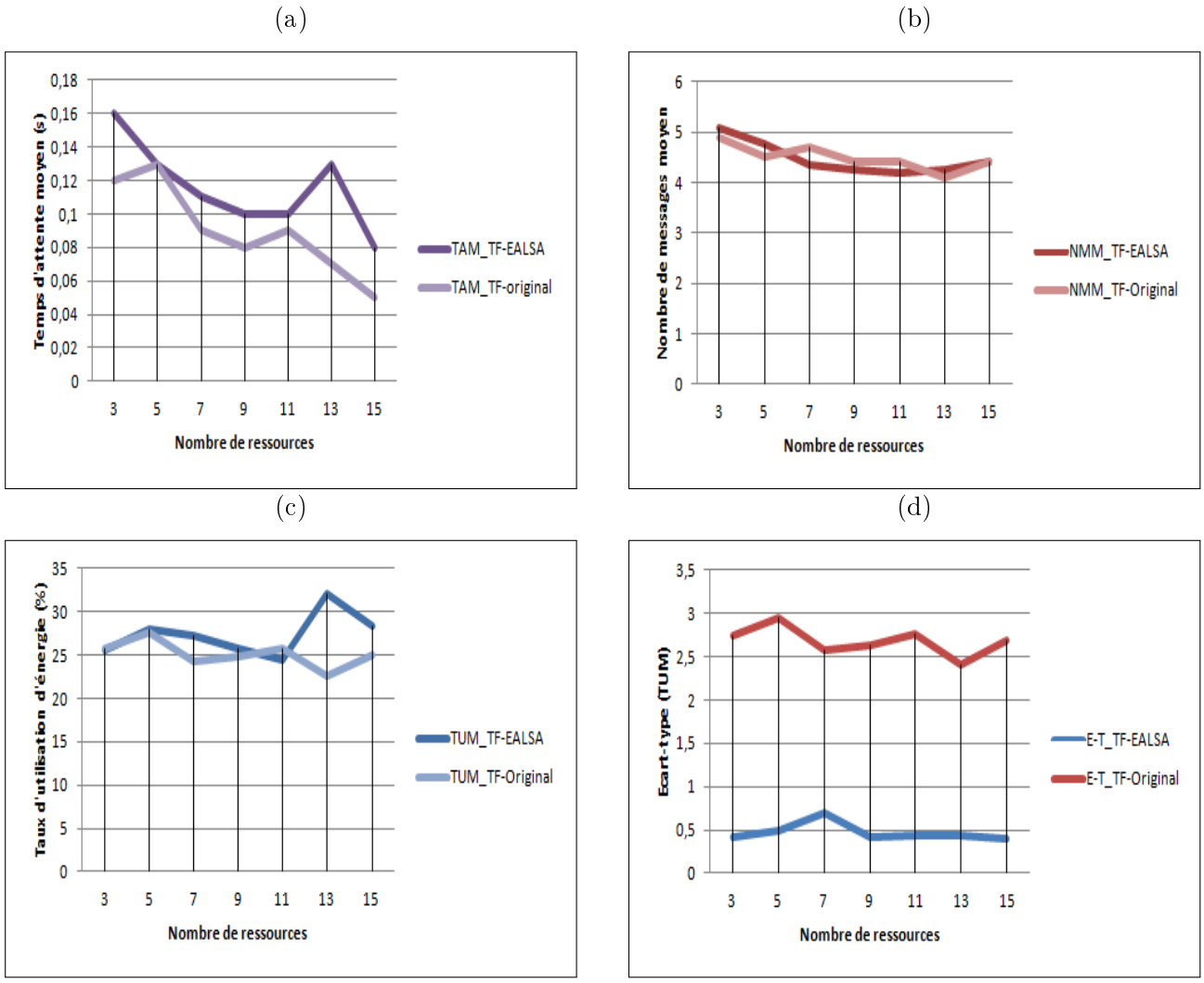


FIGURE 3.4 – Influence du nombre de ressources sur TAM, NMM, TUM, et l'écart-type du TUM

Variation du nombre de requêtes Généralement le TAM augmente à cause de l'augmentation du nombre de requêtes, et la même chose pour le TUM on justifie ça par l'augmentation du nombre de requêtes et les messages ajoutés lors des pannes .
Dans la figure 3.5d on trouve que pour notre algorithme le partage de la charge entre les noeuds est mieux que pour l'ancienne approche c'est toujours à cause de la nouvel technique utilisée pour sélectionner un leader ou bien un noeud S ou B.

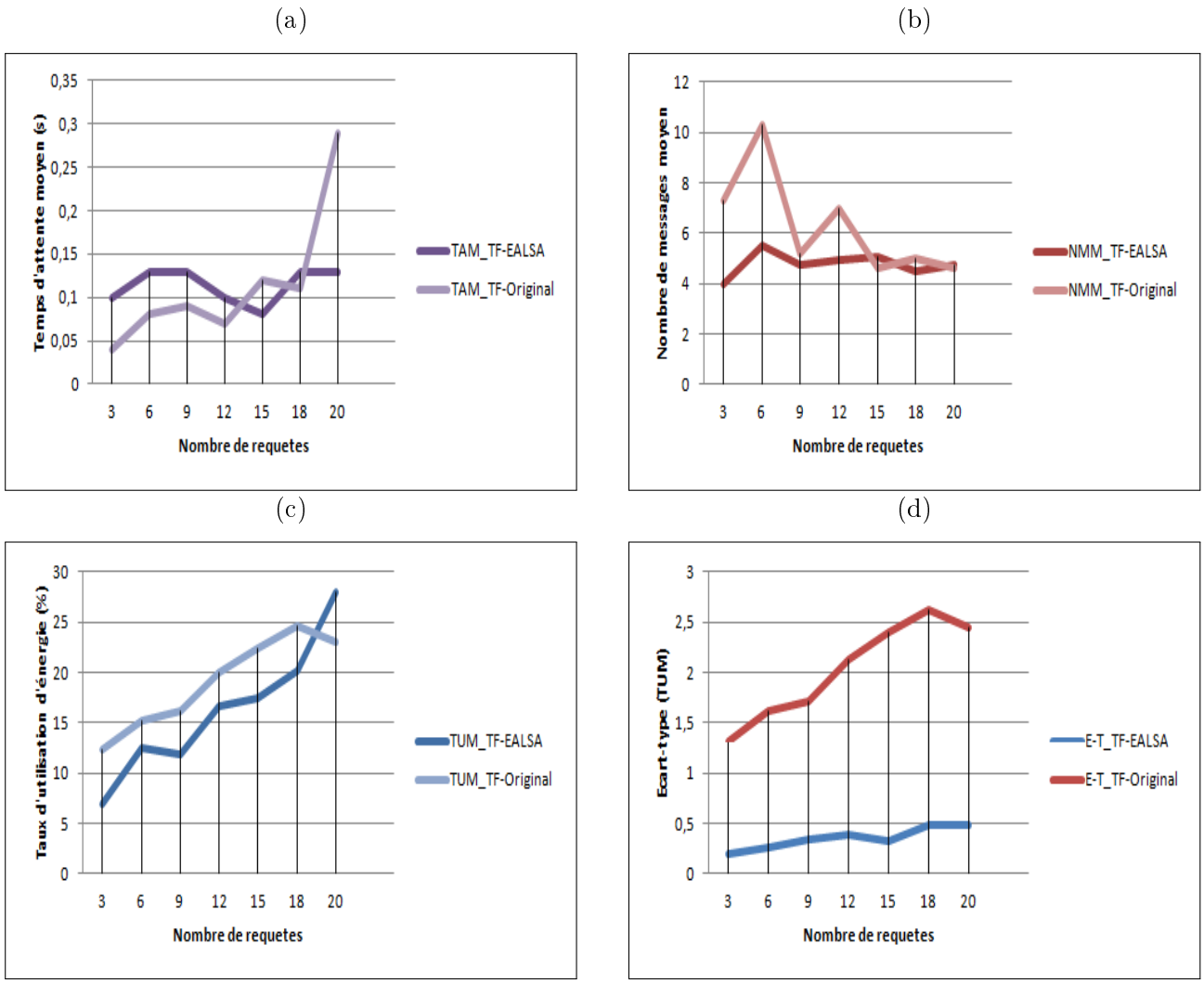


FIGURE 3.5 – Influence du nombre de requêtes sur TAM, NMM, TUM, et l'écart-type du TUM

Variation du nombre de noeuds Le TAM augmente avec l'augmentation du nombre de noeuds pour les deux courbes à cause de la mise à jour effectuée lors des pannes mais pour l'ancien algorithme le TAM est très élevé, on le justifie par la pannes de plusieurs leaders.
Le TUM augmente avec l'augmentation du nombre de noeuds parce qu'on aura plus de noeuds qui vont participer dans la délivrance de paquets. Dans la figure 3.6d on trouve que la technique utilisée a une influence positive sur l'équilibrage de charge si en comparent avec l'ancien algorithme.

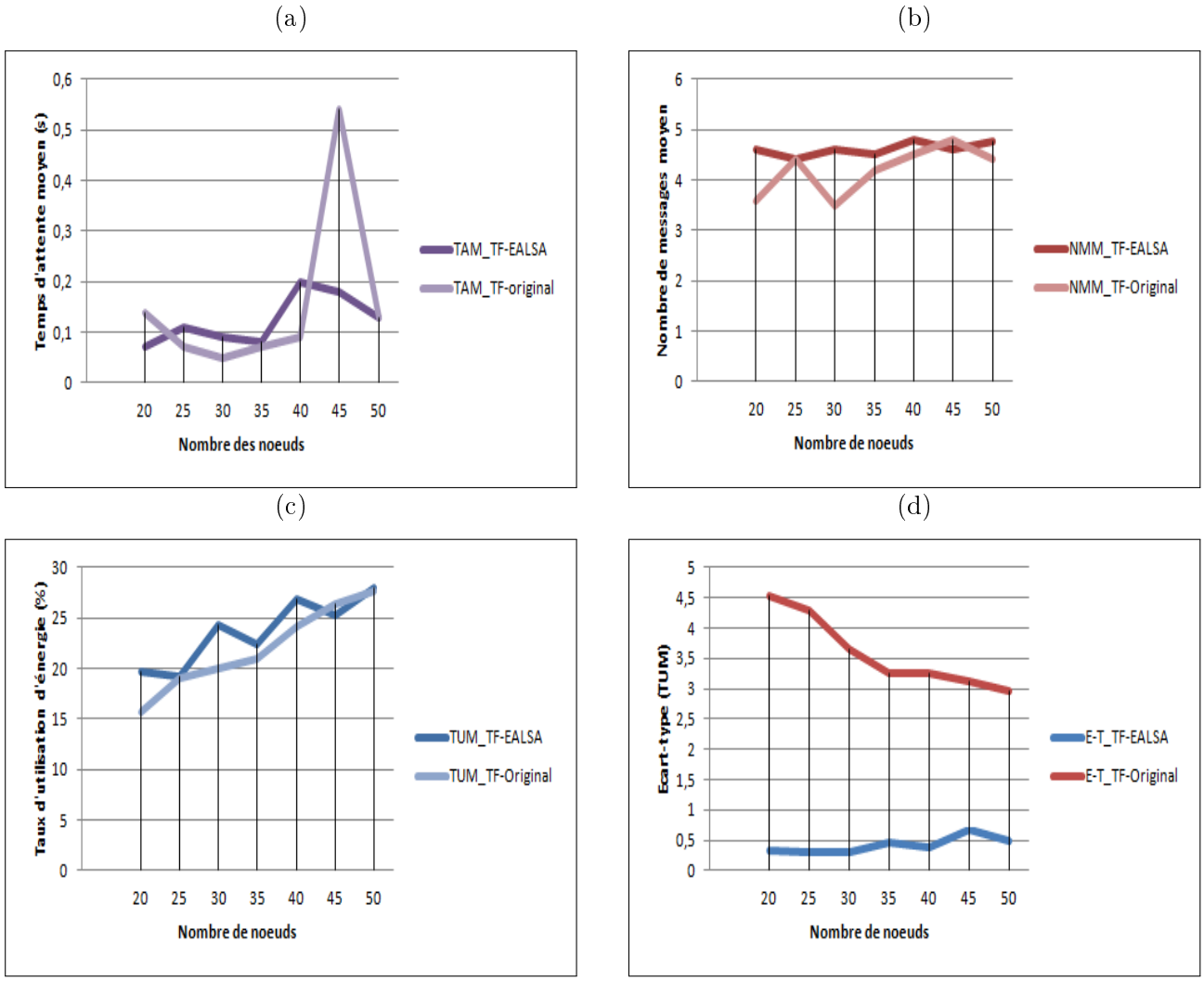
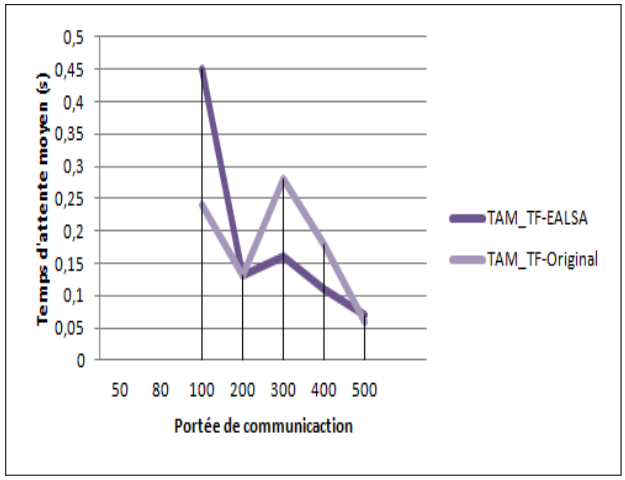


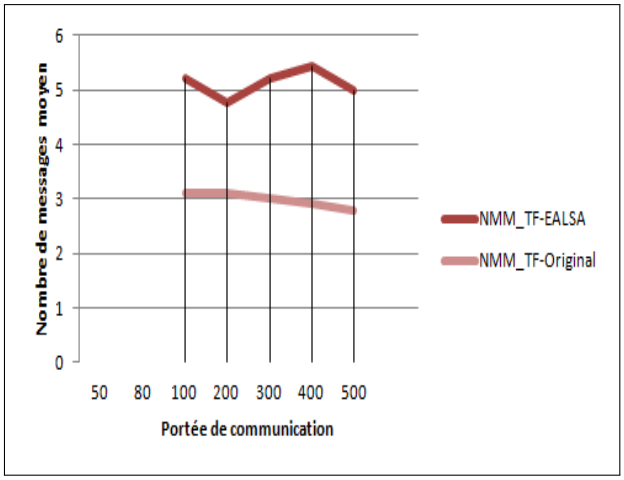
FIGURE 3.6 – Influence du nombre de noeuds sur TAM, NMM, TUM, et l'écart-type du TUM

Variation de la portée de communication Dans la figure 3.7a on voie que TAM pour notre algorithme diminue avec l'augmentation de la portée de communication parce les noeuds auront plus de voisins directs, et on justifie l'augmentation du NMM par la panne de plusieurs leaders. Pour l'équilibrage de charge notre approche est toujours la meilleure c'est toujours à cause des techniques utilisées pour sélectionner un leader ou bien un noeud S ou B.

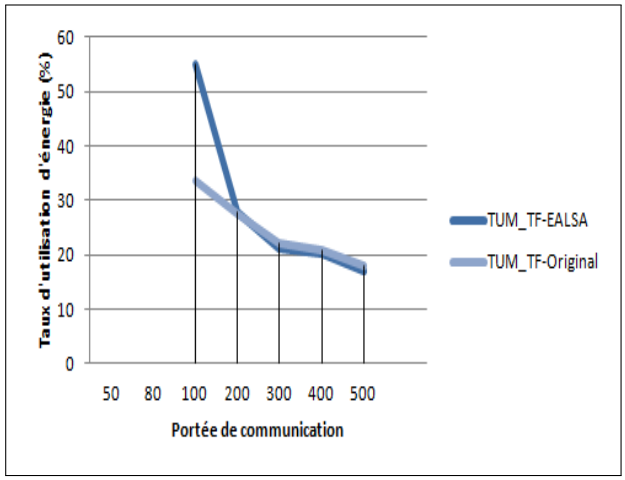
(a)



(b)



(c)



(d)

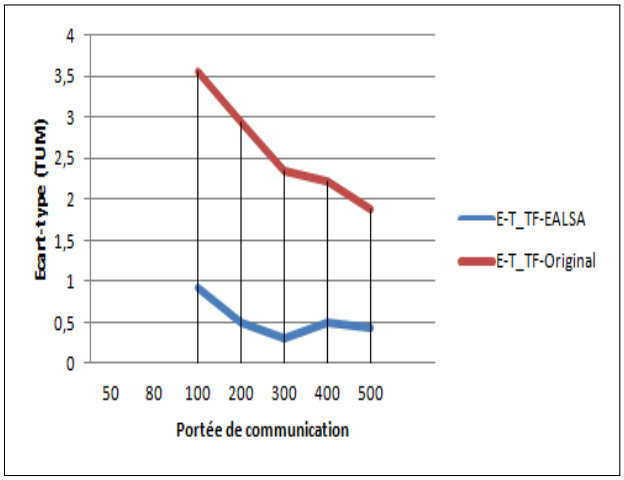
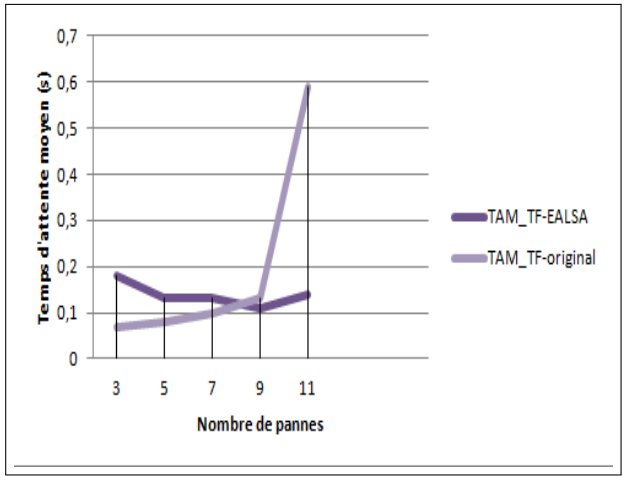


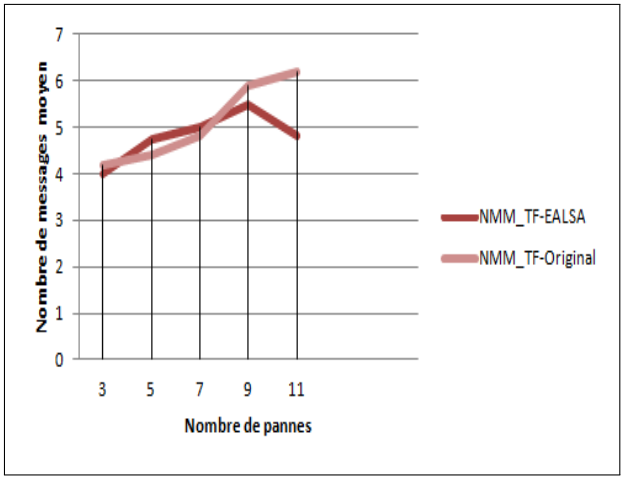
FIGURE 3.7 – Influence de la portée de communication sur TAM, NMM, TUM, et l'écart-type du TUM

Variation du nombre de pannes On trouve que le nombre de panne n'avait pas une influence sur le TAM, on justifie ça par la panne n'a pas atteint que les noeud D.
On constate un bon résultat pour l'équilibrage de charge parce que le choix est basé sur l'énergie et la distance en même temps .

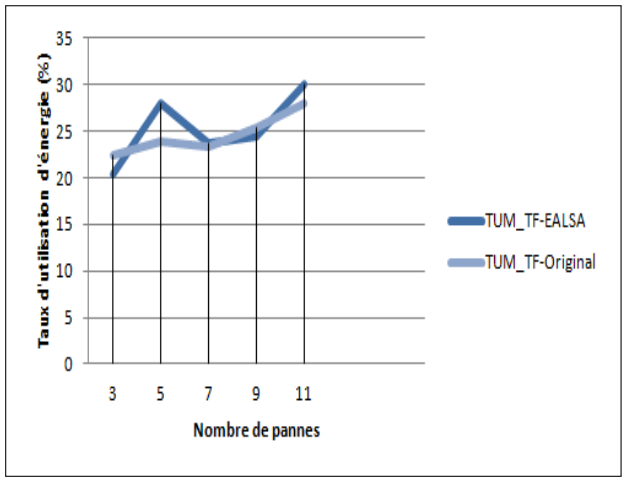
(a)



(b)



(c)



(d)

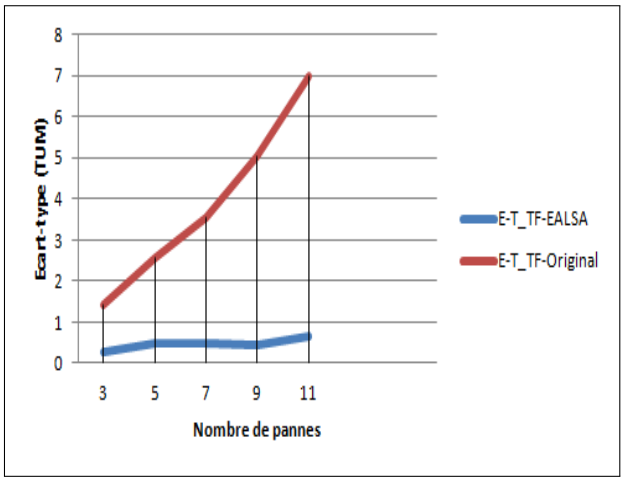


FIGURE 3.8 – Influence du nombre de pannes sur TAM, NMM, TUM, et l'écart-type du TUM

3.5 Conclusion

D'après les résultats de simulation des deux algorithmes, on déclare que notre algorithme ne traite pas que le problème de partage de ressources, il est tolérant aux pannes et assure l'équilibrage de charge entre les noeuds afin de diminuer le risque de la perte des jetons par l'échec des noeuds causé par le déchargement de la batterie.

Conclusion générale

Dans ce mémoire on a parlé sur les principaux concepts des réseaux mobiles ad hoc, on a aussi expliqué le problème de partage de ressources ainsi que la notion de l'exclusion mutuelle. L'objectif de ce mémoire était d'ajouter une amélioration à l'algorithme proposé dans [1] pour le rendre performant en terme de l'équilibrage de charges entre les noeuds.

On a simulé les anciens algorithmes[1] et les nouveaux algorithmes en utilisant plusieurs scénarios pour étudier ses performances, et on a fait une comparaison entres les résultats obtenus, donc on peut dire qu'on a arrivé à des performances assez acceptables.

Ce modeste travail est loin d'être parfait, mais quand même il m'a permis de comprendre le problème de partage de ressources en générale soit dans les systèmes distribués ou bien dans les MANETs. il m'a permis aussi de s'habituer à manipuler le fameux simulateur des réseaux $NS2$.

Bibliographie

[1] Kerrache Chaker Abdelaziz. Simulation d'un nouvel algorithme de partage de ressources tolérant aux fautes dans les réseaux mobiles ad hoc. *Thèse Master de l'université Amar Telidji-Laghouat Spécialité informatique*, 2012.

[2] Marwan Al-Jemeli, Fawnizu Azmadi Hussin, and Brahim Belhaouari Samir. Energy efficient and high throughput composite routing metric for mobile wireless sensor networks. *International Review on Computers and Software (IRECOS)*, 8(9) :2269–2277, 2013.

[3] Youssef BADDI. Introduction au simulateur rśeau NS2. `http://y-baddi.developpez.com/tutoriels/ns2/`, consulté le 28/05/2017.

[4] Stefano Basagni, Marco Conti, Silvia Giordano, and Ivan Stojmenovic. *Mobile ad hoc networking*. John Wiley and Sons, 2004.

[5] Edsger W Dijkstra. Solution of a problem in concurrent programming control. In *Pioneers and Their Contributions to Software Engineering*, pages 289–294. Springer, 2001.

[6] Kevin Fall and Kannan Varadhan. The ns manual (formerly ns notes and documentation). *The VINT project*, 47, 2005.

[7] M Handy and D Timmermann. Simulation of mobile wireless networks with accurate modelling of non-linear battery effects. In *Proc. Int'l. Conf. Applied Simulation and Modeling*, pages 532–537, 2003.

[8] Teerawat Issariyakul and Ekram Hossain. *Introduction to network simulator NS2*. Springer Science & Business Media, 2011.

[9] Swati Gupta Jai Shree Mehta, Shilpa Nupur. An overview of manet : Concepts, architecture and issues. *International Journal of Research in Management, Science and Technology*, 3(2) :98–101, 2015.

[10] Julien Sopena. *Algorithmes d'exclusion mutuelle : tolérance aux fautes et adaptation aux grilles*. PhD thesis, Paris 6, 2008.

[11] Pradip K Srimani and Rachamallu LN Reddy. Another distributed algorithm for multiple entries to a critical section. *Information Processing Letters*, 41(1) :51–57, 1992.

[12] Jun-Zhao Sun. Mobile ad hoc networking : an essential technology for pervasive computing. In *Info-tech and Info-net, 2001. Proceedings. ICII 2001-Beijing. 2001 International Conferences on*, volume 3, pages 316–321. IEEE, 2001.

[13] ALLAOUI Tahar. *Une nouvelle solution du problème de la K-Exclusion Mutuelle dans les systèmes répartis*. PhD thesis, Université Amar Telidji de Laghouat, 2007.

[14] Andrew S Tanenbaum. *Distributed operating systems*. Pearson Education India, 1995.

[15] JE Walter and Savita Kini. Mutual exclusion on multihop, mobile wireless networks. *Texas A&M Univ., College Station, TX 77843-3112, TR97*, 14, 1997.

[16] Jennifer E Walter, Jennifer L Welch, and Nitin H Vaidya. A mutual exclusion algorithm for ad hoc mobile networks. *Wireless Networks*, 7(6) :585–600, 2001.

[17] wikipedia. Tolŕance aux pannes. `https://fr.wikipedia.org/wiki/Tol\'rance_aux_pannes`, consulté le 07/05/2017.

[18] Weigang Wu, Jiannong Cao, and Jin Yang. A fault tolerant mutual exclusion algorithm for mobile ad hoc networks. *Pervasive and Mobile Computing*, 4(1) :139–160, 2008.

Annexe

.1 Annexe

.1.1 Le simulateur NS2

.1.2 Présentation du simulateur réseaux NS2

NS est un outil logiciel de simulation de réseaux informatiques. Il est essentiellement élaboré avec les idées de la conception par objets, de la réutilisation du code et de modularité. Il est aujourd'hui un standard de référence en ce domaine, plusieurs laboratoires de recherche recommandent son utilisation pour tester les nouveaux protocoles.
Le simulateur NS actuel est particulièrement bien adapté aux réseaux à commutation de paquets et à la réalisation de simulations de grande taille (le test du passage à l'échelle). Il contient les fonctionnalités nécessaires à l'étude des algorithmes de routage unicast ou multicast, des protocoles de transport, de session, de réservation, des services intégrés, des protocoles d'application comme FTP. A titre d'exemple la liste des principaux composants actuellement disponibles dans NS par catégorie est :

- application : Web, ftp, telnet, générateur de trafic (CBR...) ;
- transport : TCP, UDP, RTP, SRM ;
- routage unicast : Statique, dynamique (vecteur distance) ;
- routage multicast : DVMRP, PIM ;
- gestion de file d'attente : RED, DropTail, Token bucket.[3]

.1.3 L'architecture de base

La figure 9 montre l'architecture de base de $NS2$. $NS2$ fournit aux utilisateurs une commande exécutable ns qui prend comme argument d'entrée le nom d'un fichier Tcl de script de simulation. Dans la plupart des cas, un fichier de trace est créée et utilisé pour tracer un graphe et /ou pour créer une animation. NS_2 se compose de deux langages clés : C ++ et Object-oriented Tool Command Language (OTcl). Bien que le C ++ définisse le mécanisme interne (c'est-à-dire un backend) des objets de simulation, l'OTcl configure la simulation en assemblant et configurant les objets ainsi que dans la planification d'évènements discrets (c'est-à-dire un frontend). Le C ++ et l'OTcl sont liés ensemble à l'aide de TclCL

Après la simulation, $NS2$ produit des résultats de simulation basés sur des textes ou des animations. Pour interpréter ces résultats graphiquement et de façon interactive, des outils tels que NAM (Network AniMator) et XGraph sont utilisés. Pour analyser un comportement particulier du réseau, les utilisateurs peuvent extraire un sous-ensemble pertinent de données textuelles et les transformer en une présentation plus concevable.[8]

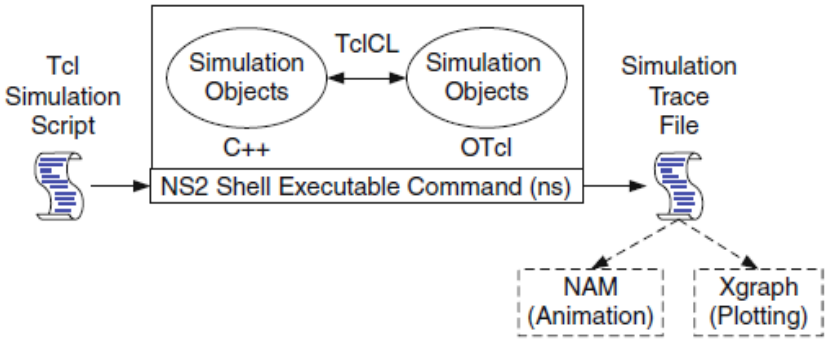


FIGURE 9 – L'architecture de simulation avec NS2

.1.4 L'outil de visualisation NAM

$NS2$ ne permet pas de visualiser le résultat des expérimentations. Il permet uniquement de stocker une trace de la simulation, de sorte qu'elle puisse être exploitée par un autre logiciel, comme NAM.
NAM est un outil de visualisation qui présente deux intérêts principaux : représenter la topologie d'un réseau décrit avec NS2, et afficher temporellement les résultats d'une trace d'exécution NS-2. Par exemple, il est capable de représenter des paquets TCP ou UDP, la rupture d'un lien entre noeuds, ou encore de représenter les paquets rejetés d'une file d'attente pleine. Ce logiciel est souvent appelé directement depuis les scripts TCL pour NS2, pour visualiser directement le résultat de la simulation.[3]

.1.5 Le script tcl

```
||==================================================================================================||
                                    Definition des options
||==================================================================================================||
set val(chan)                          Channel/WirelessChannel       ;# type de canal
set val(prop)                          Propagation/TwoRayGround      ;# model du propagation radio
set val(netif)                         Phy/WirelessPhy               ;# type d'interface reseau
set val(mac)                           Mac/802_11                    ;# type de mac
set val(ifq)                           CMUPriQueue                   ;# type interface de file d'attente
set val(ll)                            LL                            ;# link layer type
set val(ant)                           Antenna/OmniAntenna           ;# model d'antenne
set val(x)                             500                           ;# X dimension du topology
set val(y)                             500                           ;# Y dimension du topology
set val(ifqlen)                        50                            ;# max de packet dans la file
set val(seed)                          0.0                           ;# grain random
set val(adhocRouting)                  AODV                          ;# protocole de routage
set val(sc) "/home/Bouchra/..../ns-2.35/tcl/mobility/scene/modele_mobilité" ;# fichier du model de mobilite
set val(stop)                         30                            ;# duree de simulation
||==================================================================================================||
                                    Programme Principal
||==================================================================================================||
# Les portees
#Phy/WirelessPhy set RXThresh_ 7.69113e-08 ;#50 meters
#Phy/WirelessPhy set RXThresh_ 3.00435e-08 ;#80 meters
#Phy/WirelessPhy set RXThresh_ 1.42681e-08 ;#100 meters
Phy/WirelessPhy set RXThresh_ 8.91754e-10 ;#200 meters
#Phy/WirelessPhy set RXThresh_ 1.76149e-10 ;#300 meters
#Phy/WirelessPhy set RXThresh_ 5.57346e-11 ;#400 meters
#Phy/WirelessPhy set RXThresh_ 2.282e-11 ;#500 meters
||**************************************************************************************************||
                        creation d'instance de simulateur et topo
||**************************************************************************************************||
set ns_ [new Simulator]                                           ;# nouvelle simulation
$ns_ use-newtrace                                                 ;# nouveau fichier trace
set topo [new Topography]                                         ;# nouvelle topologie
||**************************************************************************************************||
                              Creation des fichier trace pour ns et nam
||**************************************************************************************************||
set tracefd  [open adhoc.tr w]
set namtrace [open adhoc.nam w]
set bw nbr_msg_tps.txt                                      ;# un ficher contient les resultats de simulation
set mesure [open $bw w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
||==================================================================================================||
                         Procedure d'affichage concernant la simulation
||==================================================================================================||
proc Affichage {} {
global  mesure nbr nbracine nbrequete                             ;# Declaration des variables globales
puts  $mesure ""
puts  $mesure "Les Résultats de la Simulation (Energy-Aware Leader Selection Algorithm)"
puts  $mesure "______________________________________________________________________________"
puts  $mesure "-----------Les paramètres de Simulation ----------------------------------------"
puts  $mesure "Le Nombre de Noeuds = $nbr"
puts  $mesure "Le Nombre de Leaders initiaux  L = $nbracine"
puts  $mesure "Le Nombre de Requetes = $nbrequete"
puts  $mesure "------------------------------------------------------------------------------"
puts  $mesure "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
puts  $mesure "    Résultas pour chaque noeud :"
puts  $mesure "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
 }
||==================================================================================================||
                  Procedure de Terminaison de la simulation
||==================================================================================================||
proc finish {} {
global ns_ nf f0 nbr nbrequete tpatt nmsg p mesure Ttaux nb          ;# les variables globales
for {set i 0} {$i < $nbr} {incr i} {                                 ;# compter le nombre de message total
set nm 0
set ttl 0
set nmsg [expr [$p($i) set nb_message_]+$nmsg]                       ;# la somme des messages de chaque noeud
set nm [$p($i) set nb_message_]
set panne [$p($i) set Enpanne]                                       ;# savoir si le noeud est en panne ou non
if {$panne == 0} {                                                   ;# Si le noeud n'est pas en panne
set re [expr 6.0-[$p($i) set EneResiduelle]]
set taux($i) [expr [expr $re*100]/6]                                 ;# on calcule le taux d'utilisation d'energie

puts $mesure "le taux d'utilisati on d'energie pour le noeud $i = $taux($i)"

set Ttaux [expr $Ttaux+$taux($i)]                                    ;# la somme des taux d'utilisation d'energie
```

```
set nb [expr $nb+1]                                    ;# la somme des noeuds qui ne sont en panne
  }
  }
set Ttaux [expr $Ttaux/$nb]                            ;# la somme des taux d'utilisation d'energie
for {set j 0} {$j < $nbr} {incr j} {              ;# calculer l'ecart-type de taux d'utilisation d'energie
set pn [$p($j) set Enpanne]
if {$pn == 0} {
set tot [expr $taux($j)-$Ttaux]
set caree [expr pow($tot,2)]
set ttl [expr $ttl+$caree]
  }
    }
set ttl [expr $ttl/$nb]
set ttl [expr {sqrt($ttl)}]                       ;# calculer l'ecart-type de taux d'utilisation d'energie
puts $mesure "______________________________________________________________________________"
puts $mesure "Le nombre de message moyen*Le temps d'attente moyen*Le taux d'utilisation d'énergie moyen*Ecart-type"
puts $mesure "______________________________________________________________________________"
puts $mesure "Le nombre total des messages = $nmsg" ;# Ecriture des resultat dans un fichier TEXT
puts $mesure "                                               "
puts $mesure "Le nombre de message moyen = $nmsg/$nbrequete = [expr $nmsg/$nbrequete]"     ;# Calcul du NMM
set tpatt [expr $tpatt/$nbrequete]                          ;# Calcul du temps d'attente moyen
puts $mesure "Le temps d'attente moyen = $tpatt"              ;# Affichage du temps d'attente moyen
puts $mesure "Le taux d'utilisation d'energie moyen = $Ttaux %"
puts $mesure "l'Ecart-type = $ttl"
puts $mesure "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
$ns_ flush-trace                                    ;# Pour Confirmer l'ecriture dans le fichier TEXT
close $mesure                                       ;# Fermer le fichier text des resultat
puts "running nam..."                               ;# Affichage de " Running NAM ... "
exec nam adhoc.nam &                                ;# lancer le NAM automatiquement
exit 0                                              ;# Sortie de la procedure
  }
||==========================================================================================||
                    Procedure d'enregistrement des temps d'attente dans le fichier text
||==========================================================================================||
proc record {p n mutex hour} {
global f0 nbrequete tpatt mesure                    ;# Declaration des variables globales
set ns [Simulator instance]                         ;# Instancier la commande NS
set bw0 [$ns now]                                   ;# Definir bw0 comme etant le temps actuelle
set bw0 [expr $bw0- [$p set dem]]             ;# Faire l'affectation bw0 := bw0 - Temps de la demande
puts $mesure "Temps d'attente pour le noeud N° [$n node-addr] =  $bw0 ==>  ( Date_Requete = $hour )"
puts $mesure "______________________________________________________________________________"
set tpatt [expr $tpatt+$bw0]                        ;# le temps d'attente
  }
||==========================================================================================||
                    Procedure de Diffusion des demandes d'entrer en Section Critique
||==========================================================================================||
proc diffusion {p n mutex hour} {
global  nbr                                         ;# Declaration des variables globales
set ns_ [Simulator instance]                        ;# Instancier la commande NS
set nowe [$ns_ now]                                 ;# Definir nowe comme etant le temps actuelle
$ns_ at $nowe "tester $p $n $mutex $hour"           ;# et executer actuellement la procedure tester
$ns_ at $nowe "$p demander-sc"                      ;# et executer la procedure demander-sc
$ns_ trace-annotate "[$n node-addr] demande la SC "    ;# Tracer le noeud qui veut entrer en Section Critique
  }
proc diffusion1 {p n hour} {
global  nbr                                         ;# Declaration des variables globales
set ns_ [Simulator instance]                        ;# Instancier la commande NS
set nowe [$ns_ now]                                 ;# Definir nowe comme etant le temps actuelle
$ns_ at $nowe "tester1 $p $n $hour"                 ;# et executer actuellement la procedure tester1
$ns_ at $nowe "$p demander-sp"                      ;# et executer la procedure demander-sp
$ns_ trace-annotate "[$n node-addr] tombe en panne "
 ;# Tracer le noeud qui va tomber en panne
  }
||==========================================================================================||
                    Procedure de test d'entrer et la liberation de la Section Critique
||==========================================================================================||
proc tester {p n mutex hour} {
global nbr  mesure                                  ;# Declaration des variables globales
set ns_ [Simulator instance]                        ;# Instancier la commande NS
set time 0.01                          ;# Le temps d'attente pour entrer en Section Critique
set now [$ns_ now]                                  ;# Definir now comme etant le temps actuelle
set a [$p set sc]                                   ;# Affectation a := sc ;
set e [$p set nbjeton]                              ;#
if { $a == 1} {                                     ;# Condition pour entrer en Section Critique
$ns_ at $now "record $p $n $mutex $hour"        ;# Appeler la procedure RECORD pour calculer le temps d'attente
$ns_ at $now "$n label \"<Leader> \""               ;# Afficher le libele <Leader> a cote du noeud
set f [$p set sp]                                   ;# Affectation f := sp ;
$ns_ at [expr $now+$mutex] "$n label \" \""         ;# puis l'Effacer apres un temps de duree MUTEX
$ns_ at [expr $now+$mutex] "$p liberation-sc"       ;# Puis appeler la procedure liberation-sc
$ns_ at [expr $now+$mutex] "$p set sc -1"        ;# affecter a sc := -1 comme etant il est sorti de la SC
```

```
   } else {                                                ;# Si le noeud desire entrer en Section Critique
$ns_ at [expr $now+$time] "$p attendre"                    ;# il doit attendre pour mettre a jour la variable SC
$ns_ at [expr $now+$time] "tester $p $n $mutex $hour"
 ;# rappeler la procedure TESTER (( Recursif )) pour entrer
  }
  }
||=========================================================================================================||
                              Procedure de tomber en panne
||=========================================================================================================||
proc tester1 {p n hour} {
global nbr  mesure                                         ;# Declaration des variables globales
set ns_ [Simulator instance]                               ;# Instancier la commande NS
set time 0.01                                          ;# Le temps d'attente pour entrer en Section Critique
set now [$ns_ now]                                         ;# Definir now comme etant le temps actuelle
set a [$p set sp]                                          ;# Affectation a := sp ;
if { $a == 1} {                                            ;# Condition pour tomber en panne
$ns_ at $now "$n label \"\""                               ;# Afficher le libele a cote du noeud
   } else {                                                ;# Si le noeud desire entrer en Section Critique
$ns_ at [expr $now+$time] "$p attendre-panne"              ;# il doit attendre pour mettre a jour la variable SC
$ns_ at [expr $now+$time] "tester1 $p $n $hour"            ;# rappeler la procedure TESTER (( Recursif ))
  }
  }
||*********************************************************************************************************||
                                  Le programme Principale
||*********************************************************************************************************||
||=========================================================================================================||
                          Lecture des donnees a partir d'un fichier texte
||=========================================================================================================||
set f [open "les_pannes.txt" "r"]                           ;# Ouvrir le fichier
set nbr [gets $f]                                           ;# Affecter la premier ligne a (( nbr ))
set nbpanne [gets $f]                                       ;# Affecter la troisiemme ligne a (( nbpanne ))
for {set j 0} {$j < $nbpanne} {incr j} {
;# Boucle Imbriqee pour lire le reste du Scenario de Simulation
for {set k 0} {$k < 2} {incr k} {
set table1($j,$k) [gets $f]                             ;# Remplir la table par les valeurs (( site, heur, duree ))
   }
 }
close $f
set f [open "les_demandes.txt" "r"]                       ;# Ouvrir le fichier (( les_demandes.txt )) pour le lire
set nbr [gets $f]                                         ;# Affecter la premier ligne a (( nbr ))
set nbracine [gets $f]                                    ;# Affecter la deuxiemme ligne a (( nbrk ))
set nbrequete [gets $f]                                   ;# Affecter la troisiemme ligne a (( nbrequete ))
for {set j 0} {$j < $nbrequete} {incr j} {       ;# Boucle Imbriquee pour lire le reste du Scenario de Simulation
for {set k 0} {$k < 2} {incr k} {
set table($j,$k) [gets $f]                       ;# Remplir la table par les valeurs (( site, heur, duree ))
   }
 }
close $f
||=========================================================================================================||
                              Declaration des couleurs selon les numeros
||=========================================================================================================||
$ns_ color 0 Blue                                             ;# Le 0 est la Couleur Bleu (( Demande ))
$ns_ color 1 red                                              ;# Le 1 est la Couleur Rouge (( Requete ))
$ns_ color 2 limegreen                                        ;# Le 2 est la Couleur Vert (( Liberation ))
$ns_ color 3 brown                                            ;# Le 3 est la Couleur Marron (( Accord ))
$ns_ color 4 magenta                                          ;# Le 4 est la Couleur Violet (( Aide ))
$ns_ color 5 Black                                            ;# Le 5 est la couleur Noir (( Search ))
||*********************************************************************************************************||
 #Definition de topologie
||*********************************************************************************************************||
$topo load_flatgrid $val(x) $val(y)
||*********************************************************************************************************||
 #Creation de God
||*********************************************************************************************************||
set god_ [create-god $nbr]
||*********************************************************************************************************||
#Configuration globale d'un noeud
||*********************************************************************************************************||
$ns_ node-config -adhocRouting $val(adhocRouting) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -channelType $val(chan) \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
```

```
                -macTrace OFF
||*****************************************************************************************************||
# Creation des noeuds selon la valeur [$val(nn)] et attaches aux canal,agent.
||*****************************************************************************************************||
for {set i 0} {$i < $nbr} {incr i} {
$ns_ node-config -energyModel "EnergyModel" \                    ;# definir le model d'energie pour chaque noeud
-initialEnergy 6 \                                               ;# l'energie initiale en joule
-rxPower 0.5 \                                                   ;# la puissance de d'emission en watt
-txPower 1 \                                                     ;# la puissance de reception en watt
if {$i == 0 || $i == 34 || $i == 46 || $i == 4 || $i == 17 } {
$ns_ node-config -energyModel "EnergyModel" \                    ;# definir le model d'energie pour chaque noeud
-initialEnergy 0.8 \                                               ;# l'energie initiale en joule
-rxPower 0.5 \                                                   ;# la puissance de d'emission en watt
-txPower 1 \
}
set node_($i) [$ns_ node]
$node_($i) random-motion 0
$node_($i) color Black
$god_ new_node $node_($i)
set p($i) [new Agent/TF-EALSA]
$p($i) set nb_noeud_ $nbr
$p($i) set nbracine $nbracine
$ns_ at 0.0 "$p($i) initialisation"
$ns_ attach-agent $node_($i) $p($i)
$p($i) set packetSize_ 1024
}
for {set i 0} {$i < $nbr} {incr i} {
for {set j [expr $i+1]} {$j < $nbr} {incr j} {
$ns_ connect $p($i) $p($j)
}
}
proc colorRd {p node_ } {
set ns_ [Simulator instance]
set nowe [$ns_ now]
$ns_ at $nowe "$node_ color red"
}
proc colorBl {p node_ } {
set ns_ [Simulator instance]
set nowe [$ns_ now]
$ns_ at $nowe "$node_ color blue"
}
proc colorBk {p node_ } {
set ns_ [Simulator instance]
set nowe [$ns_ now]
$ns_ at $nowe "$node_ color black"
}
proc colorLm {p node_ } {
set ns_ [Simulator instance]
set nowe [$ns_ now]
$ns_ at $nowe "$node_ color limegreen"
}
proc colorOr {p node_ } {
set ns_ [Simulator instance]
set nowe [$ns_ now]
$ns_ at $nowe "$node_ color Orange"
}
proc coloration {p node_ } {
set ns_ [Simulator instance]
set nowe [$ns_ now]
set a [$p set father]
set b [$p set nbjeton]
set c [$p set sc]
set d [$p set numrequest]
set e [$p set sp]
if {$b == 1} {
$ns_ at $nowe "$node_ color Orange"
} else {
$ns_ at $nowe "$node_ color Black"
}
if {$d == 1} {
$ns_ at $nowe "$node_ color Limegreen"
}
if {$c == 1} {
$ns_ at $nowe "$node_ color Blue"
}
if {$e == 1} {
$ns_ at $nowe "$node_ color Red"
}
}
||*****************************************************************************************************||
```

```
#  Appel des fichiers des mouvements
||*********************************************************************************************************||
if { $val(sc) == "" } {
puts "*** NOTE: no connection pattern specified."
set val(sc) "none"
} else {
puts "Loading connection pattern..."
source $val(sc)
}
||*********************************************************************************************************||
# pour compter le nbr de messsges echanges
||*********************************************************************************************************||
set nmsg1 0                         ;# initialisation de nmsg1
set nmsg 0                          ;# initialisation de nmsg
set tpatt 0                         ;# initialisation de tpatt
set Ttaux 0                         ;# initialisation de Ttaux
set nb 0                            ;# initialisation de nb
||*********************************************************************************************************||
#duree de la sc
||*********************************************************************************************************||
set duree 1    ;# la duree de la Sc
||*********************************************************************************************************||
#procedure pour colorer les nouds leaders
||*********************************************************************************************************||
for {set i 0} {$i < $nbr} {incr i} {
$ns_ initial_node_pos $node_($i) 60  ;# La fonction doit etre appele apres la definition de modele de mobilite
}
||*********************************************************************************************************||
#initialistion
||*********************************************************************************************************||
$ns_ at 0.0 "Affichage"
set ns_ [Simulator instance]
set nowe [$ns_ now]
for {set i 0} {$i < $nbr} {incr i} {
set temps 0.0
for {set j 0} {$j < 2000} {incr j} {
set temps [expr $temps+0.01]
$ns_ at $temps "coloration $p($i) $node_($i)"
}
}
||*********************************************************************************************************||
#charger les noeuds qui vont tomber en panne
||*********************************************************************************************************||
for {set j 0} {$j < $nbpanne} {incr j} {
set noeud $table1($j,0)
set inst $table1($j,1)
$ns_ at $inst "diffusion1 $p($noeud) $node_($noeud) $inst"
puts "panne au niveau de :"
puts "$noeud"
}
||*********************************************************************************************************||
#charger les noeuds qui demandent l'entrer a la  sc
||*********************************************************************************************************||
for {set j 0} {$j < $nbrequete} {incr j} {
set site $table($j,0)
set heure $table($j,1)
$ns_ at $heure "$p($site) set sc 0"
$ns_ at $heure "$p($site) set dem $heure"
$ns_ at $heure "diffusion $p($site) $node_($site) $duree $heure"
puts "$site"
}
for {set i 0} {$i < $nbr} {incr i} {
$ns_ at $val(stop) "$p($i) Finir" ;# récuperer l'energie des noeuds a la fin de simulation
}
$ns_ at [expr $val(stop) + 0.01] "finish"
puts "Debut de la simulation ..."
$ns_ run
```