République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Ammar Telidji
Laghouat

Faculté des Sciences et Sciences de l'Ingénierie
Département de Génie Informatique

Mémoire de Master

Filière: Informatique

Option: Réseaux, systèmes et applications répartis (**ReSar**)

Thème:

Simulation d'un nouvel algorithme de partage de ressources tolérant aux fautes dans les réseaux mobiles ad hoc

Présenté par:

Kerrache Chaker Abdelaziz

Soutenu devant le jury composé de:

M^{r}	X. Xxxxxxx	Université de Laghouat	(Président)
M^{r}	X. Xxxxxxx	Université de Laghouat	(Examinatrice)
M^{r}	X. Xxxxxxx	Université de Laghouat	(Examinateur)
M^{r}.	T. Allaoui	Université de Laghouat	(Rapporteur)

Année 2012

Je dédie ce mémoire
à
mes parents
mes frères et sœurs
et spécialement à
Youcef et Kaouthar

Remerciements

Je remercie Dieu de m'avoir donné la force physique et intellectuelle pour accomplir ce travail, et pour les richesses dont il me comble.

Ces quelques lignes ne pourront jamais exprimer la reconnaissance que j'éprouve envers tous ceux qui, de près ou de loin, ont contribué par leurs conseils, leurs encouragements ou leurs amitiés à l'aboutissement de ce travail.

Qu'il me soit permis de rendre un vibrant hommage à mon encadreur Mr Allaoui Tahar , qui est avant tout le frère de mon frère Allaoui Ismail ,ainsi que mon CO-encadreur Oubbati Omar Sami , pour moi un immense honneur de travailler avec eux durant mon projet fin d'étude, je les remercie également pour leurs disponibilité, leurs sens aigu de l'humanisme pédagogique, et enfin leurs rigueur intellectuelle et morale qui est pour moi plus qu'un exemple à suivre.

J'aimerais aussi témoigner ma reconnaissance à tous mes professeurs et enseignants qui m'ont permis, par leurs efforts, d'atteindre un tel niveau de formation.

Enfin je remercie les membres du jury qui ont bien voulu accepter, et ce malgré, leur lourdes et exaltantes responsabilités pour procéder à l'évaluation de ce travail.

Résumé

Ce mémoire présente un algorithme tolérant aux fautes au problème de la K-exclusion mutuelle dans les réseaux mobiles AD HOC.

Nous allons d'abord introduire le concept des systèmes répartis et les réseaux mobiles AD HOC avant de parler des problèmes classiques de ce genre de systèmes,puis nous étudions l'un de ces problèmes qui est le problème de partage d'une ressource, et sa généralisation en K ressources.
Nous expliquerons également la notion de la tolérance aux fautes qui permet de ne pas baisser les performances des algorithmes même en présence de pannes.

Nous présentons un nouvel algorithme traitant problème de la K-exclusion mutuelle, cet algorithme est basé sur le protocole de routage réactif AODV, ce qui permet de diminuer le nombre de messages échangés.
Cet algorithme sera amélioré par l'intégration d'un nouveau mécanisme de tolérance aux fautes.
Nous avons utilisé l'outil de simulation NS-2 afin d'étudier la performance de l'algorithme proposé, cette simulation nous a permi de spécifier les paramètres qui influent sur la performance de notre algorithme.

Mots-clés : Réseaux mobiles AD HOC, algorithmique répartie, Exclusion mutuelle, K-exclusion mutuelle, Routage, Simulation, NS-2.

Abstract

This thesis presents a faults tolerant algorithm of the K-mutual exclusion problem in mobile AD HOC networks.

We will first introduce the concept of distributed systems and mobile AD HOC networks , the classical problems of this type of systems, then we study one of these problems that is the problem of resource sharing, and its generalization to K resources.

We will also explain the notion of fault tolerance that allows for not to decrease the performance of the algorithms in the presence of failures.

We present a new algorithm of K-mutual exclusion problem, this algorithm is based on the AODV reactive routing protocol, which allows to reduce the number of exchanged messages.
This algorithm will be improved by the integration of a new mechanism of fault tolerance. We have used the NS-2 simulation tool to study the performance of the proposed algorithm, this simulation has to specify the parameters that affect the performance of our algorithm.

Key-words : Mobile AD HOC Network (MANET), Distributed algorithm, Mutual exclusion, K-mutual exclusion, Routing, Simulation, NS-2.

Table des matières

Liste des figures

Introduction générale

La croissance très rapide de la technologie sans fil ainsi que les unités de calcul portables a orienté les recherches vers le but essentiel des réseaux c'est-à-dire L'accès à l'information n'importe où, et n'importe quand.
Par conséquence les réseaux sans fil ont gagné un intérêt majeur et grâce aux avantages qu'ils offrent, ces réseaux occupent de plus en plus de place dans les différents genres de communications.

Les réseaux mobiles sans fil, peuvent être classés en deux grandes catégories : les réseaux mobiles avec infrastructure ou cellulaire, et les réseaux mobiles sans infrastructure ou les réseaux mobiles AD HOC.

Le concept des réseaux mobiles AD HOC essaie d'étendre les notions de la mobilité à toutes les composantes de l'environnement. Ici, contrairement aux réseaux basés sur la communication avec infrastructure (cellulaire), aucune administration centralisée n'est disponible, ce sont les hôtes mobiles eux-mêmes qui forment une infrastructure du réseau. Aucune supposition ou limitation n'est faite sur la taille du réseau mobile AD HOC, le réseau peut contenir des centaines d'unités mobiles.

L'inexistence d'administration centralisée dans les réseaux mobiles AD HOC avec la possibilité de partager des ressources entre les différents sites du réseau peut mener à des incohérences à cause de l'accès simultané par plusieurs nœuds à une même ressource.
Il est donc très important d'assurer l'accès exclusif à cette ressource, cette notion a donné naissance au problème de l'exclusion mutuelle dans les réseaux mobiles AD HOC.
Dans le cas où notre système est doté de plusieurs ressources (K ressources), nous parlerons donc de la généralisation de ce problème au problème de la K-exclusion mutuelle.

L'étude des performances d'une nouvelle solution apportée à ces problèmes nécessite une intervention directe dans un réseau réel, une chose qui est difficile voire même impossible, vue la difficulté d'extraction des données, ainsi que les anomalies pouvant accompagnées une nouvelle solution, nous utilisons des outils de simulation qui nous permettent de tester et évaluer les algorithmes proposés dans des conditions très proches de la réalité.

Les algorithmes proposés dans ces cadres doivent être bien conçus de façon qu'ils garantissent une fiabilité très élevée, mais malheureusement on ne peut pas évité d'éventuelles pannes, alors les chercheurs se sont penchés sur ce sérieux problème afin de garantir que les performances des systèmes ne diminuent pas même en présence de pannes, donnant naissance à la notion de tolérance aux fautes.

Dans ce mémoire, nous allons proposer et valider par la simulation, un nouvel algorithme de la K-exclusion mutuelle dans les réseaux mobiles AD HOC.nous allons introduire et intégrer un nouveau mécanisme de tolérance au pannes dans notre algo-

rithme de la K-exclusion mutuelle.

Le reste de ce mémoire est organisé comme suit :

Dans le *1er chapitre*, nous allons aborder des notions générales sur les systèmes distribués, les réseaux mobiles AD HOC et le problème de l'exclusion mutuelle et la K- exclusion mutuelle dans ce genre de réseaux ainsi que la notion de tolérance aux fautes.

Le *chapitre 2*, est consacré à l'étude de l'algorithme proposé dans le cadre de la K-exclusion mutuelle dans les réseaux mobiles AD HOC. Nous l'expliquons en donnant l'idée de base, le principe de fonctionnement nous présentons la réalisation de la simulation de cet algorithme et la discussion des différents résultats obtenus durant la simulation.

le *dernier chapitre*,est consacré à l'étude et à l'intégration d'un nouveau mécanisme de tolérance aux fautes dans notre algorithme de la K-exclusion mutuelle dans les réseaux mobiles AD HOC .

La *conclusion de ce mémoire* résume les travaux faits durant la réalisation de ce travail ainsi que des possibles améliorations futures.
Une présentation de l'outil de simulation NS-2 et un script TCL utilisé lors de la simulation sont mis à la disposition du lecteur à la fin de ce mémoire.

Notions générales

1

Sommaire

Dans ce chapitre, nous allons présenter le concept des systèmes distribués , les réseaux mobiles AD HOC . Nous introduisons également le concept du problème de partage de ressources dans ce type de réseaux,ainsi que la notion de tolérance aux fautes et en fin l'outil de simulation NS-2 .

1.1 Introduction

Le monde des réseaux sans fil est devenu l'un des axes de recherche les plus importants ces dernières années. L'évolution récente des moyens de communication sans fil a permis la manipulation de l'information à travers des unités de calcul mobiles. Les environnements mobiles offrent aujourd'hui une grande flexibilité d'emploi, en particulier, ils permettent la mise en réseau des sites dont le câblage serait difficile à réaliser .

1.2 Les systèmes répartis

Avec l'apparition des réseaux informatiques, les besoins en termes de calcul et de communication augmentent de jour en jour. Alors un système appelé centralisé, basé sur une seule machine fait son apparition, mais l'augmentation journalière des besoins a contribué à l'émergence de l'informatique dite répartie.
Avec L'informatique répartie Nous sommes en train de passer d'une architecture où une machine fournissait des services à un ensemble de machines (système centralisé), à une architecture où un ensemble de machines reliées par un réseau, composent un système qui fournit des services (système réparti).

1.2.1 Définition d'un système reparti

Ils existent plusieurs définitions d'un système réparti qui diffèrent d'un auteur à un autre. Tanenbaum [Tan94] a défini un système réparti comme un système qui s'exécute sur un ensemble de machines sans mémoire partagée dont l'utilisation se voit comme une seule et unique machine.
une autre définition dit qu'un système réparti est défini comme étant un ensemble de sites autonomes connectés par un réseau de communication, et équipés d'un logiciel dédié à la coordination des activités du système ainsi qu'au partage de ressources [Cou94].
Nous pouvons dire donc qu'un système réparti est un système composé de plusieurs machines autonomes qui communiquent entre elles par échange de messages.
Chaque ordinateur est une entité autonome capable de réaliser des tâches indépendamment des autres entités.
Au niveau de chaque site qui a une mémoire propre et une horloge locale, un ensemble de processus s'exécutent soit en collaboration soit en compétition. Pour des raisons de simplicité, on suppose qu'il y a un seul processus au niveau de chaque site, donc, dans tout ce qui suit, les termes : site,nœud et processus seront confondus, et ils représenteront la même chose.

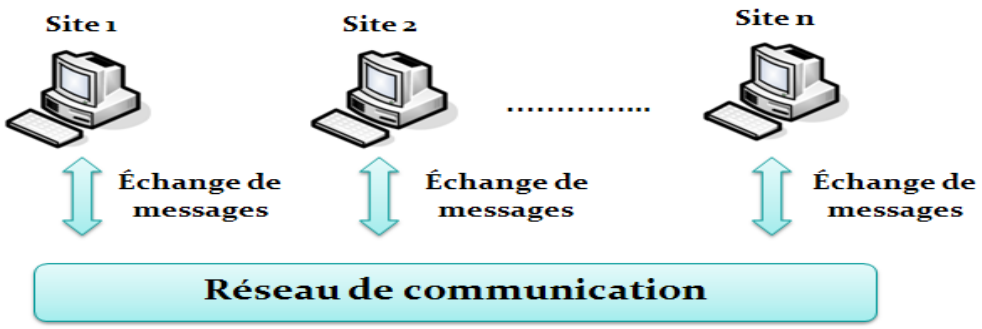

Figure 1.1 – *Structure d'un système réparti*

1.2.2 Les caractéristiques d'un système réparti

Un système réparti est caractérisé par :

- L'absence d'une mémoire commune et d'un état global.
- L'ensemble des machines doit posséder un mécanisme de communication inter-processus unique et global. Cette caractéristique permet la communication entre les différents processus.
- Les composants défectueux peuvent se relancer et rejoindre le système, après que la cause de l'échec a été réparée.
- La performance s'obtient en réduisant le nombre des messages transmis et en augmentant le parallélisme. On peut dire que la performance s'obtient en réglant le grain de parallélisme, c'est-à-dire l'intervalle de temps séparant deux communications successives.[Eveo4].

1.2.3 Les avantages et les inconvénients

1.2.3.1 Avantages

parmi Les avantages des systèmes repartis par rapport aux systèmes centralisés nous avons :

- **Sécurité :** les applications sont conçues selon une approche modulaire permettant d'isoler les données, et donc protéger les accès.

- **Accélération des calculs** : le système réparti permet de répartir les calculs sur les différents processus afin de les exécuter simultanément.ainsi Lorsqu'un site est surchargé, certaines tâches peuvent être déplacées ou allouées à un site moins chargé (la répartition des charges).

- **Transparence** : l'utilisation des ressources n'influe pas sur leurs états des sites (c'est-à-dire les sites utilisent les ressources sans savoir l'emplacement de ces ressources).

- **Flexibilité** : l'ajout ou la suppression d'un site est une opération simple, et n'influe pas sur le fonctionnement total du système.

- **Fiabilité** : l'un des buts des systèmes répartis est d'obtenir des systèmes plus fiables qu'un système centralisé. Si une machine tombe en panne, une autre machine devrait

prendre la relève, cette situation entraîne généralement la duplication des données partagées.

- **Accès distant** : un même service peut être utilisé par plusieurs sites, situés à des endroits différents.

- **Redondance** : des systèmes redondants permettent de palier une faute matérielle ou de choisir le service équivalent avec le temps de réponse le plus court.[Sopo8]

1.2.3.2 Inconvénients

parmi les inconvénients des systèmes répartis par rapport à la centralisation :

- **L'absence d'une horloge globale** : chaque nœud possède sa propre horloge pour dater les événements qui lui sont locaux. Par conséquent, si les horloges indépendantes de chaque nœud ne sont pas synchronisées, l'ordre des événements n'est pas déductible à partir des datations locales. Cette difficulté conduira à définir des datations logiques qui permettent de corriger ce problème.

- **La lenteur de la communication** : malgré le degré de fiabilité offert par les réseaux de communication, et les vitesses de transmission qu'ils proposent restent relativement lents par rapport à la vitesse de calcul sur les machines, ce qui implique un grand temps d'attente dans le cas d'un travail coopératif entre les processus, et donc on a une diminution concernant la performance du système.

- **La perte des messages** : parfois, les réseaux de communication ne sont pas fiables, quelques messages transportés par ces réseaux peuvent être perdus, et donc on doit ré-émettre ces messages.[Allo4]

Les systèmes répartis ont résolus beaucoup de problèmes, et avec l'évolution de la technologie du monde actuel, un nouveau mode de communication sans fil a fait son apparition, son point fort est la mobilité des sites donc rendant la mise en œuvre des réseaux plus facile sans utilisation du câblage. Les réseaux AD HOC viennent de montrer leurs efficacités.

1.3 Les réseaux sans fil

1.3.1 Introduction

les réseaux sans fil sont devenus l'axe de recherche le plus important ces dernières années. L'évolution récente des moyens de communication sans fil a permis la manipulation de l'information à travers des unités de calcul mobiles.

Les environnements mobiles offrent aujourd'hui une grande flexibilité d'emploi. En particulier, ils permettent la mise en réseau des sites dont le câblage serait très difficile à réaliser.
Un environnement mobile est un système composé de sites mobiles et qui permet à ses utilisateurs d'accéder à l'information indépendamment de leurs positions géographiques,

en utilisant les ondes radio pour communiquer.

Les réseaux mobiles sans fil peuvent être divisés en deux classes :

– Les réseaux avec infrastructure.

– Les réseaux sans infrastructure.

1.3.1.1 Les réseaux mobiles avec infrastructure

Dans cette classe de réseaux, nous avons des unités fixes appelées station de base (SB), ces stations, sont reliées entre elles par un réseau filaire. Chaque station de base a une interface de communication sans fil, et elle couvre une zone géographique limitée appelée cellule, c'est la raison pour laquelle on appelle ces réseaux les réseaux cellulaires.

Les autres unités dans ce réseau sont des unités mobiles (UM) ayant des liens sans fil avec les stations de base, une unité mobile n'appartient qu'à une seule SB à un moment donné, cette station offre tous les services de communication tant que l'UM est à l'intérieur de sa zone de couverture.

les communications entre UMs se fait par l'intermédiaire de leur même SB si elles appartiennent à la même cellule, Sinon par l'intermédiaire des deux SBs , les informations échangées entre les SBs doivent être relayées par le réseau filaire.[Lemoo] (Voir la figure 1.2).

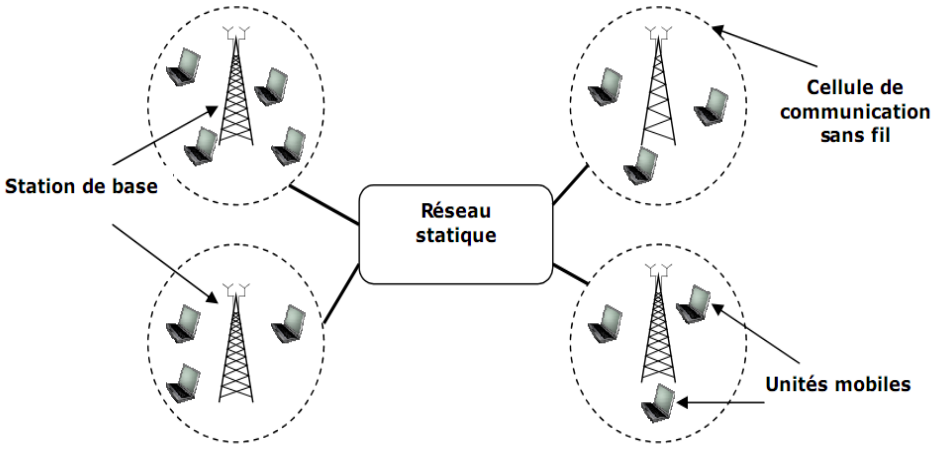

Figure 1.2 – *Le modèle des réseaux mobiles avec infrastructure.*

1.3.1.2 Les réseaux mobiles sans infrastructure

Ce modèle est représenté par les réseaux Ad Hoc et étend les notions de mobilité à tous les éléments qui composent le réseau.

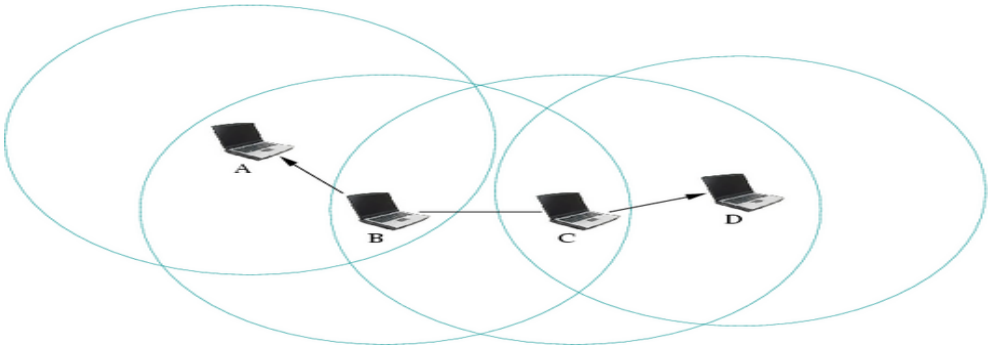

Figure 1.3 – *Le modèle des réseaux mobiles sans infrastructure.*

1.3.2 Les réseaux mobiles AD HOC

Le terme AD HOC en latin « qui va vers, ce vers quoi il doit aller », c'est-à-dire « formé dans un but précis », les réseaux AD HOC sont des réseaux sans fil capables de s'organiser sans infrastructure définie préalablement.

Les réseaux mobiles AD HOC sont connus sous le nom de MANET (Mobile Ad-hoc NETworks).

Un réseau mobile AD HOC est composé d'unités mobiles qui se déplacent dans un territoire quelconque et dont le seul moyen de communication est l'utilisation des interfaces sans fil, sans avoir besoin d'une infrastructure préexistante ou d'une administration centralisée.

L'absence de l'infrastructure rend le comportement des unités mobiles comme des nœuds intermédiaires (Routeur) qui participent à la découverte et à la maintenance des chemins pour les autres nœuds du réseau.

Pour assurer la communication entre les nœuds, ces derniers utilisent les signaux radio, le rayon de couverture du signal est dit porté de communication. Un nœud ne peut communiquer directement qu'avec les nœuds qui sont situés à l'intérieur de sa portée de communication, ces nœuds forment alors l'ensemble des voisins immédiats de ce nœud.[Mes98]

La figure 1.4 représente un réseau AD HOC de 7 unités mobiles et leurs liens de communication.

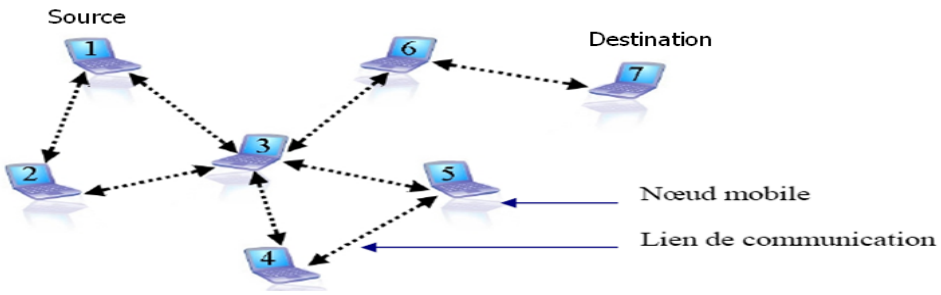

Figure 1.4 – *Un réseau AD HOC.*

1.3.2.1 Les caractéristiques des réseaux AD HOC

Les réseaux AD HOC présentent de nombreuses caractéristiques dont certaines leur sont bien spécifiques et les différencient des réseaux mobiles classiques.

- **Une topologie dynamique** : les unités mobiles du réseau, se déplacent d'une façon libre et arbitraire. Par conséquent, la topologie du réseau peut changer, à des instants imprévisibles, d'une manière rapide et aléatoire (voir figure 1.5).

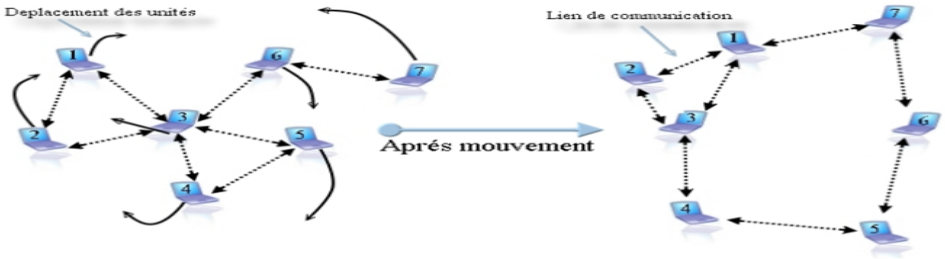

Figure 1.5 – *Topologie dynamique dans un réseau AD HOC.*

- **L'absence d'infrastructure** : les réseaux AD HOC se distinguent des autres réseaux mobiles par l'absence d'infrastructure préexistante et de tout genre d'administration centralisée. Les hôtes mobiles sont responsables d'établir et de maintenir la connectivité du réseau d'une manière continue.

- **Equivalence des nœuds du réseau** : dans un réseau classique, il existe une distinction nette entre les nœuds terminaux (stations, hôtes) qui supportent les applications et les nœuds internes (routeurs par exemple) du réseau, en charge de l'acheminement des données. Cette différence n'existe pas dans les réseaux AD HOC car tous les nœuds peuvent être amenés à assurer des fonctions de routage.

- **Rapidité de déploiement** : les réseaux AD HOC peuvent être facilement installés dans les endroits difficiles à câbler, ce qui élimine une bonne part du travail et du coût.

- **Communication par lien radio** : les communications entre les nœuds se font par l'utilisation d'une interface radio. Il est alors important d'adopter un protocole d'ac-

cès au médium qui permet de bien distribuer les ressources radio, et ceci en évitant le plus possible les collisions, et en réduisant les interférences.[Mes98]

1.3.2.2 Les avantages des réseaux AD HOC

- **Faciles à déployer** : il suffit de mettre en place plusieurs machines pour que le réseau existe. Ceci rend la construction d'un réseau AD HOC rapide et peu onéreuse.

- **Les nœuds sont mobiles** : l'absence de câblages autorise les nœuds à se déplacer librement.

- **Évolutifs** : pour ajouter un nœud à un réseau AD HOC préexistant, il suffit d'approcher le nouveau venu d'au moins l'un des membres du réseau. De même il suffit de l'en éloigner pour le retirer du réseau.[Mano7]

1.3.2.3 Les inconvénients des réseaux AD HOC

- **Une bande passante limitée** : une des caractéristiques primordiales des réseaux basés sur la communication sans fil est l'utilisation d'un médium de communication partagé. Ce partage fait que la bande passante réservée à un hôte soit modeste.

- **Des contraintes d'énergie** : les hôtes mobiles sont alimentés par des sources d'énergie autonomes donc restreintes, comme les batteries, par conséquent la durée de traitement est réduite. Donc le paramètre d'énergie doit être pris en considération dans tout contrôle fait par le système.

- **Une sécurité physique limitée** : les réseaux mobiles AD HOC sont plus touchés par le paramètre de sécurité que les réseaux filaires classiques. Cela se justifie entre autres par les vulnérabilités des liens radio aux attaques, ainsi que les contraintes et limitations physiques qui font que le contrôle des données transférées doit être minimisé.[Mes98]

1.3.2.4 Applications des réseaux AD HOC

Bien que les projets relatifs aux réseaux sans fil en général et aux réseaux AD HOC en particulier aient débuté dans un cadre militaire pur, leurs domaines d'applications s'étendent au-delà du cadre militaire. Les réseaux mobiles AD HOC sont faciles à déployer dans des bâtiments surtout les vieux bâtiments ou les sites classés (châteaux et monuments historiques).

D'une façon générale, les réseaux AD HOC sont utilisés dans toute application où le déploiement d'une infrastructure réseau filaire est difficile à mettre en place, ou lorsque la durée d'installation du câblage est importante.[Mano7]

Parmi les domaines d'applications des réseaux AD HOC :

- **Services d'urgence** : opération de recherche et de secours des personnes, tremblement de terre, Incendies, dans le but de remplacer l'infrastructure filaire.

- **Travail collaboratif et les communications dans des entreprises ou bâtiments** : dans le cadre d'une réunion par exemple .

- **Applications commerciales** : pour un paiement électronique distant (restaurant, aéroport, hôtel) ou pour l'accès mobile à Internet, ou service de guide en fonction de la position de l'utilisateur.

1.3.3 Le routage dans les réseaux AD HOC

1.3.3.1 Définition du routage

Le routage est l'acheminement des informations à la bonne destination à travers un réseau de connexion donné. Le problème de routage consiste à déterminer un chemin optimal des paquets à travers le réseau au sens d'un certain critère de performance.[Lemoo]

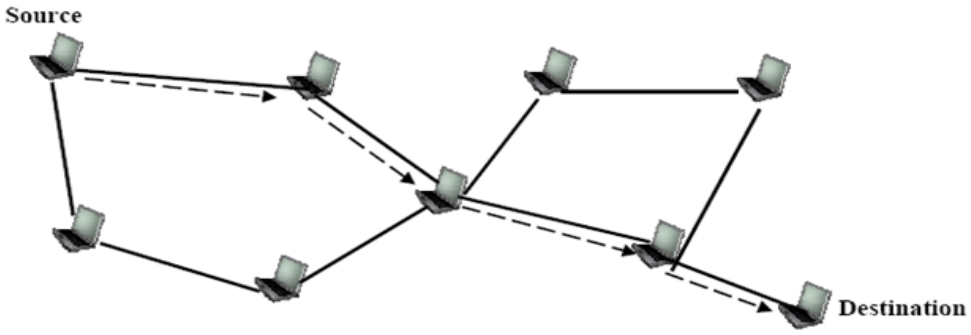

Figure 1.6 – *Le chemin utilisé dans le routage entre la source et la destination.*

1.3.3.2 Classification des protocoles de routage

Suivant la manière de création et de maintenance de routes lors de l'acheminement des données, les protocoles de routage peuvent être classés en trois catégories, les protocoles proactifs, les protocoles réactifs et les protocoles hybrides.[Lemoo]
Comme il est illustré dans la figure 2.1.

1.3.3.2.a Les protocoles de routage proactifs

Les protocoles de routage proactifs pour les réseaux mobiles AD HOC, sont basés sur la même philosophie des protocoles de routage utilisés dans les réseaux filaires conventionnels. Les méthodes utilisées exigent une mise à jour périodique des données de routage qui doivent être diffusées par les différents nœuds de routage du réseau.
OLSR (Optimized link state routing protocol) est l'un protocoles qui appartenant à cette classe.

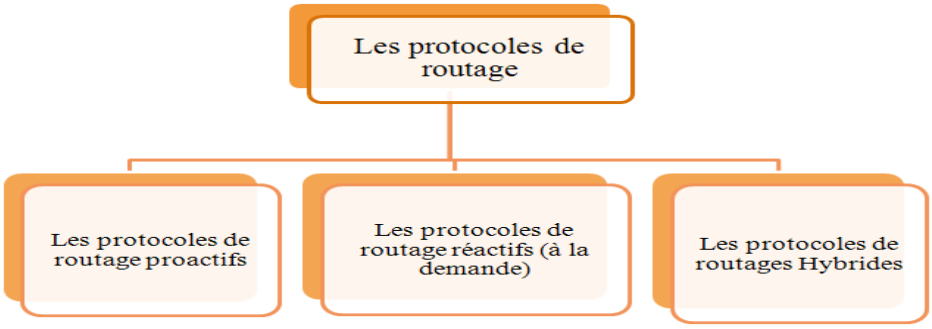

Figure 1.7 – *la classification des protocoles de routage.*

1.3.3.2.b Les protocoles de routage réactifs (à la demande)

Les protocoles de routage appartenant à cette catégorie, créent et maintiennent les routes selon les besoins. Lorsque le réseau a besoin d'une route, une procédure de découverte globale de routes est lancée, et cela dans le but d'obtenir une information spécifiée, inconnue au préalable.

AODV (Ad hoc On-demand Distance Vector), DSR (Dynamic Source Routing), sont les plus connus dans cette classe.

1.3.3.2.c Les protocoles de routage hybrides

Dans ce type de protocole, on peut garder la connaissance locale de la topologie jusqu'à un nombre prédéfini (à priori petit) de sauts par un échange périodique de trame de contrôle, autrement dit par une technique proactive. Les routes vers des nœuds plus lointains sont obtenues par schéma réactif, c'est-à-dire par l'utilisation de paquets, et de requêtes en diffusion.[Lemoo]

Le protocole ZRP (Zone Routinier Protocol) est le plus connu dans cette classe.

1.4 Problèmes classiques

Les réseaux AD HOC sont des systèmes répartis dont le support de communication est un réseau sans fil, il est évident de rencontrer les problèmes classiques des systèmes répartis tels que la datation des événements, l'élection, la détection de terminaison et le partage de ressources dans les réseaux AD HOC. Et à cause de la mobilité, on trouve également des problèmes propres aux réseaux AD HOC tels que le problème de routage.

1.5 Le problème de partage de ressources

Étant donné un système réparti de **N** processus partageant une ressource physique (une imprimante par exemple) ou logique (un fichier). Afin d'éviter des situations incohérentes, la ressource partagée ne peut être utilisée que par un seul processus à la fois, en d'autres termes, la ressource doit être utilisée en exclusion mutuelle (EM).

Une telle ressource est dite ressource critique, et les processus utilisant cette ressource exécutent une section du code appelée section critique (SC) qui manipule la ressource partagée. Pour garantir l'accès exclusif à la ressource critique, on doit gérer cet accès par un protocole d'acquisition et un protocole de libération. Un processus désirant entrer en SC doit appeler le protocole d'acquisition, ce protocole permet de retarder l'accès de ce processus à la SC si elle est utilisée par un autre site. À la fin de la SC, le processus exécute le protocole de libération pour informer les sites qui sont en attente que la SC est libre.[Allo7]. La forme générale d'un processus utilisant une ressource critique est :

Protocole d'acquisition
< Section Critique >
Protocole de libération

1.5.1 Notions de base

1. **Ressource critique**
   C'est une ressource partagée qui ne doit être accessible que par un seul site à la fois.
2. **Section critique**
   Les sites utilisant les ressources critiques exécutent une section du code manipulant la ressource appelée section critique (SC) : un seul processus au plus doit être en section critique afin de garantir une utilisation correcte de la ressource.

1.5.2 Le problème de l'exclusion mutuelle dans les systèmes repartis

Le problème de l'exclusion mutuelle était très bien traité dans les systèmes informatiques centralisés, plusieurs algorithmes et mécanismes assurant l'exclusion mutuelle ont été proposés, on peut citer par exemple les sémaphores de Dijkstra [Dij65] et les moniteurs. Par ailleurs, avec l'évolution des systèmes informatiques on a vu ce problème se transformé de point de vue centralisé à un point de vue réparti.

La première définition du problème de l'exclusion mutuelle a été donnée par Dijkstra [Dij65]. On peut résumer ses propos dans les cinq assertions suivantes :

1. À tout moment, il n'y a qu'un site du système qui puisse exécuter 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 faire d'hypothèse sur la vitesse des participants.
4. En dehors de la section critique, tout site peut quitter le système sans pour autant bloquer les autres.
5. Si plus d'un site désire entrer en section critique, on doit pouvoir décider en un temps fini de l'identité du site qui accédera à celle-ci.

Cette définition n'a pas évolué jusqu'à aujourd'hui, et on la retrouve dans les principales propriétés des algorithmes d'exclusion mutuelle.

1.5.3 Propriétés d'un algorithme d'exclusion mutuelle

Une solution n'est considérée correcte que si elle respecte les propriétés suivantes :

- **Propriété de sûreté** (*safety*) : à tout instant, un seul site au plus exécute la SC.
- **Propriété de vivacité** (*liveness*) : tout site demandeur de la ressource critique doit pouvoir l'acquérir au bout d'un temps fini.

  Un algorithme qui assure ces deux propriétés assure également l'absence de deux problèmes, l'interblocage et la famine :

- **Interblocage** (*Deadlock*) : est une situation du système où il y a plusieurs sites à l'état Demandeur et aucun ne peut accéder à la SC.
- **Famine** (*Starvation*) : aura lieu si un site qui se trouve à l'état Demandeur ne passe jamais à l'état Dedans.

1.5.4 Les classes de solutions d'exclusion mutuelle

Les algorithmes d'exclusion mutuelle sont classés en deux grandes familles : les algorithmes basés sur les permissions et ceux basés sur l'utilisation des jetons.

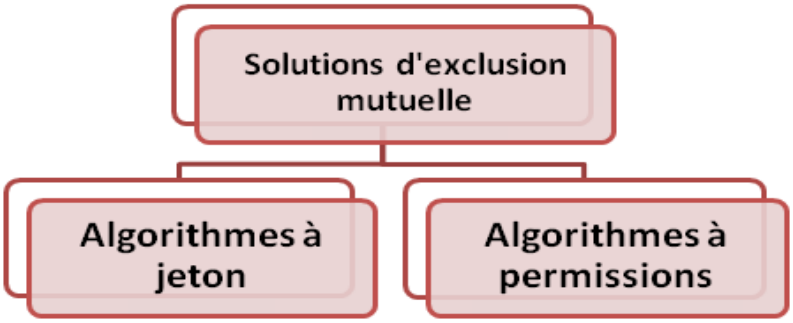

Figure 1.8 – *Les catégories des solutions d'exclusion mutuelle.*

1.5.4.1 Les algorithmes à permissions

Les algorithmes à permissions reposent sur une idée simple et naturelle : chaque site demande aux autres, s'il le désire, l'autorisation d'entrer en SC. Il ne pourra alors entrer en SC qu'à la seule condition d'avoir bien reçu toutes les permissions nécessaires.[Sopo8]

1.5.4.2 Les algorithmes à jeton

L'idée des algorithmes de cette famille est alors de considérer un témoin appelé "jeton" (token). Il peut être vu comme une super permission dont la seule possession permet d'exécuter une section critique.

1.6 Le problème de la K-exclusion mutuelle

1.6.1 Description du problème

Le problème de la K-exclusion mutuelle est une généralisation du problème de l'exclusion mutuelle simple, dans ce cas les sites partagent non seulement une ressource mais K exemplaires de la même ressource.

Cette disponibilité de ressources n'autorise pas l'accès simultané à une ressource, il faut toujours assurer l'accès exclusif à chaque exemplaire.

1.6.2 Résolution du problème

Les solutions du problème de la K-exclusion mutuelle sont basées sur les solutions utilisées pour résoudre le problème de l'exclusion mutuelle simple, on peut donc avoir deux types de solutions :

- **Solution utilisant des permissions** : le processus désirant entrer en SC doit demander des permissions à un ensemble de processus, la réception d'un nombre suffisant de ces permissions permet au processus d'utiliser la ressource.

  L'algorithme de Raymond est la première solution pour la K-exclusion mutuelle basée sur les permissions.

- **Solutions utilisant des jetons** : dans ce cas, il n'existe pas un seul jeton mais k jetons (k étant le nombre de ressources), seule la possession d'un jeton permet à un site demandeur l'accès à la SC.

  Parmi les solutions apportées basées sur les jetons on distingue l'algorithme de Srimani et Reddy [Sri89].

1.7 Le problème de l'exclusion mutuelle dans les réseaux AD HOC

Dans un environnement mobile, les unités mobiles peuvent partager les ressources présentes dans le système, tout en respectant l'exclusion mutuelle.

Les solutions proposées au problème de l'exclusion mutuelle dans les réseaux mobiles AD HOC peuvent être classées en deux grandes familles, les solutions utilisant des permissions [Wei08] et celles utilisant le jeton [WW01], ces solutions devront prendre en compte les caractéristiques des réseaux mobiles AD HOC.

1.8 Le problème de la K-exclusion mutuelle dans les réseaux AD HOC

Dans ce cas, nous avons K exemplaires d'une ressource critique, un processus ne peut utiliser qu'une seule ressource à un moment donné.

Quelques algorithmes sont proposés dans ce cadre comme [WK97].

1.9 La tolérance aux fautes

Il se peut que dans un système qui fonctionne correctement, d'avoir à un moment donné un problème qui va influencer négativement sur son comportement, c'est le problème de panne d'un site dans le système. Il existe plusieurs types de solutions permettant de résoudre ce problème.

1.9.1 Solutions

Les solutions suivantes sont plus adéquates à un système centralisé qu'un système réparti.

- **La prévention des fautes** : qui s'attache aux moyens permettant d'éviter l'occurrence de fautes dans le système.
- **L'élimination des fautes** : qui se focalise sur les techniques permettant de réduire la présence de fautes ou leurs impacts.
- **La prévision des fautes** : qui prédit l'occurrence des fautes (temps, nombre, impact) et leurs conséquences.

  Cependant dans un système réparti, ils existent plusieurs sites qui sont reliés par un réseau de communication, cela va permettre à un site de prendre les tâches d'un autre site qui a subit une faute
- **La tolérance aux fautes** : le système peut fonctionner en dépit des fautes.[AK08]

1.9.2 Les types de la tolérance aux fautes

La tolérance aux pannes permet l'assurance de la sûreté de fonctionnement, il existe deux types de tolérance : par mémoire stable, et par duplication.

1.9.2.1 Tolérance aux fautes par mémoire stable

Une mémoire stable est considérée comme étant un support persistant de stockage, dont le rôle principal est d'assurer une accessibilité et une protection des données contre les pannes pouvant affecter le système. Ainsi, suite à une panne, un état correct ayant été stocké antérieurement à cette panne sur la mémoire stable reste accessible, cela permet au système un retour à un état antérieur.

La tolérance aux fautes par sauvegarde périodique de points de reprise sur mémoire stable est très aisément mise en œuvre dans le cas d'une application séquentielle (et donc non-répartie).[Abd07]

1.9.2.2 Tolérance aux fautes par duplication

La tolérance aux pannes par duplication consiste à la création de copies multiples des composants sur des processeurs différents. Cette technique fournit un moyen de traiter les erreurs et de masquer ainsi le fait que des répliques ont défailli. Cependant, afin de restaurer le niveau de redondance des processus et d'être ainsi capables de tolérer d'autres défaillances, cette technique doit être complétée par un traitement de faute, qui consiste, entre autres, à la réallocation et l'initialisation de nouvelles réplique. [AK08]

Vue à la difficulté de l'implémentation réelle de notre algorithme, qui est dû au manque de souplesse de la variation des différents paramètres, et aux problèmes d'extraction de résultats, ce qui nous a motivé à faire appel à la simulation, nous allons utiliser l'outil NS-2, à cause des avantages qu'il offre, au niveau des systèmes répartis, et dans les réseaux mobiles AD HOC.

Conclusion

Dans ce chapitre nous avons présenté des notions de base concernant les systèmes repartis , les réseaux AD HOC , le problème de l'exclusion mutuelle et son extension à K-ressources, nous avons cité également les catégories de solutions apportées à ces problèmes.
La notion de la tolérance aux pannes a aussi fait l'objet de ce chapitre .

Dans le chapitre suivant, nous allons décrire et simuler notre algorithme, proposé pour la K-exclusion mutuelle dans les réseaux AD HOC.

2 Un nouvel algorithme basé sur le protocole de routage AODV

Sommaire

Ce chapitre est consacré à la description d'une nouvelle solution au problème la K-exclusion mutuelle dans les réseaux AD HOC.

Dans les MANETs, il existe un échange continue de messages de contrôle, tels que les messages utilisés par les protocoles de routage ainsi que les messages des différentes applications.
Ce nombre élevé de messages peut causer des problèmes tels que l'occupation de la bande passante ou l'augmentation du temps d'attente vu les mécanismes d'évitement de collision de ces message ...etc
donc Il serait très utile de trouver un moyen permettant de minimiser le nombre de messages échangés. et puisque l'échange de messages de contrôle est inévitable, nous avons pensé à intégrer des informations aux messages de contrôle, donc on peut exploiter ces messages de contrôle pour véhiculer des informations utilisées par les applications

2.1 L'idée de base

Le but de tous les algorithmes de partage de ressources est d'assurer la K-exclusion mutuelle par l'échange d'un nombre réduit de messages.dans les réseaux mobile AD HOC le protocole de routage AODV permet d'identifier le plus court chemin entre deux nœuds avant de réaliser une communication entre eux .
Notre idée consiste à exploiter cette caractéristique pour minimiser le nombre de messages échangés pour chaque entrée en SC, pour cela on va intégrer des informations concernant la disponibilité des ressources dans la table de routage de chaque site .

2.2 L'algorithme

L'idée de cet algorithme consiste à ajouter deux champs dynamiques dans la table de routage de chaque site :

– l'un pour l'existence de jetons libre que l'on appelle "jlibre" .

– et l'autre pour l'existence de jetons présent appelé "jpres" .

À chaque fois qu'il y a une modification au niveau de ces deux variables, le site doit modifier sa table de routage dans les deux champs, et par conséquent les autres sites doivent aussi modifier leurs tables de routage et faire des changements au niveau de se site et plus exactement au niveau des deux variables et cela grâce aux messages de contrôle utilisés pour le routage.
la forme de la table de routage sera :

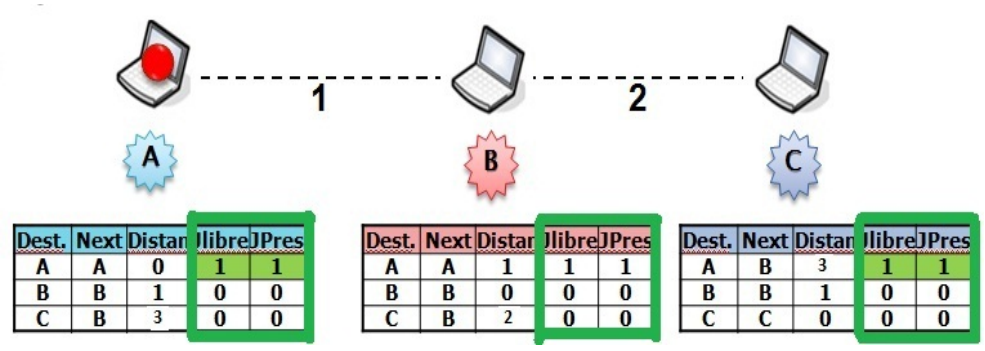

Figure 2.1 – *Format de la table de routage.*

la table de routage d'un site contient les identités et la distance des autres sites ainsi que l'information de disponibilité du jeton .
le site A possède un jeton libre donc la paire (jlibre , jpres) à comme valeur (1,1), les messages de contrôles vont acheminer cette information au autres sites donc les site B et C mettent à jour la ligne du site A dans les tables de routages.

Lorsqu'un site demandeur désire entrer en SC, il doit d'abord consulter sa table de routage, en essayant de trouver le site le plus proche parmi les sites détenant un jeton libre pour lui adresser la requête , ce mécanisme permet l'accès à la section critique avec un nombre réduit de messages.

2.2.1 Hypothèses

dans notre algorithme on suppose que :

- les nombres de nœuds et de ressources du système sont connus.
- Les canaux de communication sont fiables, et il n'y a pas de perte de messages.
- Les sites du système ne tombent pas en panne.
- Le réseau n'est pas partitionné.

2.2.2 Description de l'algorithme

Nous allons décrire les variables locales, les messages utilisés, et enfin les procédures de l'algorithme.

2.2.2.1 Les variables locales

Chaque nœud ***i*** dans le système maintient les variables locales suivantes :

- $Demandeur_i$: une variable booléenne initialisée à **Faux**, indiquant si ***i*** a demandé une ressource critique ou non.
- $Jeton_i$: une variable booléenne indiquant si ***i*** possède un jeton ou non, elle est initialisée à **Faux** au niveau des sites simples et à **Vrai** au niveau des sites D.
- $Suivant_i$: une file d'attente qui contient les identités des sites qui demandent la section critique, cette file d'attente sera envoyée avec le jeton.
- $AODV_i$(X,Y) : indique les deux champs pouvant être manipulés par notre algorithme :
  X :indique le jeton libre.
  Y :indique le jeton présent.

2.2.2.2 Les messages utilisés

Dans cet algorithme, on utilise trois types de messages :

- **Requête (Jeton, i)** : envoyé par le site ***i*** vers le site qui détient un jeton .
- **Accord (Jeton,file)** : envoyé par le site qui détient le jeton à un site demandeur pour lui permettre d'utiliser la ressource critique demandée.
- **Rejet ()** : envoyé par un ancien détenteur du jeton à un site demandeur pour lui informer qu'il ne détient plus le jeton.

Nous supposons qu'au début nous avons deux type de sites :

- : le site qui détient un jeton appelé **site D**.
- **et** le reste des sites du systèmes.

2.2.2.3 Initialisation des variables

Initialisation : des variables des sites D

$Demandeur_i$: Variable booléenne.
$Jeton_i$: Variable booléenne.
$Suivant_i$: file d'attente initialisé à NIL .

Début
$Demandeur_i \longleftarrow$ Faux;
$Suivant_i \longleftarrow$ NIL;
$Jeton_i \longleftarrow$ Vrai;
$AODV_i\_[\boldsymbol{i}]\_Jeton\_libre \longleftarrow 1$;
$AODV_i\_[\boldsymbol{i}]\_Jeton\_present \longleftarrow 1$;
Fin

Initialisation : des variables des sites simples

$Demandeur_i$: Variable booléenne.
$Jeton_i$: Variable booléenne.
$Suivant_i$: file d'attente initialisé à NIL .

Début
$Demandeur_i \longleftarrow$ Faux;
$Suivant_i \longleftarrow$ NIL;
$Jeton_i \longleftarrow$ faux;
$AODV_i\_[\boldsymbol{i}]\_Jeton\_libre \longleftarrow 0$;
$AODV_i\_[\boldsymbol{i}]\_Jeton\_present \longleftarrow 0$;
Fin

2.2.2.4 Les procédures de l'algorithme

Procédure N°1 : **Demander la Section Critique**

```
1  Début
2  |  Demandeur_i ⟵ Vrai;
3  |  Si (AODV_i_[i]_Jeton_libre = 0) Alors
4  |  |  detenant ⟵ Chercher le détenant adéquat(AODV_i);
5  |  |  Envoyer Requête (Jeton, i) à detenant;
6  |  Finsi
7  |  Attendre (Jeton = Vrai);
8  |  Modification dans la table de routage AODV_i (0,1);
9  |  < Entrer en Section Critique >
10 Fin
```

Quand un nœud qui ne possède pas de jeton désire entrer en SC, il devient demandeur et il cherche le détenant adéquat afin de lui envoyer une requête et attend la réception du jeton pour entrer en SC.

Procédure N°2 : **Réception de Accord (Jeton,File)**

```
1  Début
2  |  Jeton_i ⟵ Vrai;
3  |  Suivant_i ⟵ File;
4  Fin
```

Lorsqu'un site reçoit un message de type **Accord**, il va modifier sa variable booléenne Jeton à Vrai et il met à jour la file d'attente.

Procédure N°3 : **Réception de Requête (Jeton, j)**

```
1  Début
2      Si (AODV_i_[i]_Jeton_libre = 1) Alors
3          Jeton_i ⟵ Faux;
4          Envoyer Accord (Jeton,File) à j;
5          Modification dans la table de routage AODV_i (o,o);
6      Sinon
7          Si (AODV_i_[i]_Jeton_present = 1) Alors
8              Suivant_i ⟵ Suivant_i+{j};
9              File ⟵ Suivant_i ;
10         Sinon
11             Envoyer Rejet () à j;
12         Finsi
13     Finsi
14 Fin
```

Si un site reçoit un message de type **Requête**,il va envoyer le jeton vers le site demandeur s'il n'utilise pas la ressource,sinon il le met en attente . Dans le cas où le site ne détient pas de jeton, il va lui envoyer un message de type **Rejet** lui indiquant qu'il ne possède plus le jeton.

Procédure N°4 : **Réception de Rejet ()**

```
1  Début
2      AODV_i_[detenant]_Jeton_libre ⟵ o;
3      AODV_i_[detenant]_Jeton_present ⟵ o;
4      detenant ⟵ Chercher le détenant adéquat(AODV_i);
5      Envoyer Requête (Jeton, i) à detenant;
6  Fin
```

Lorsqu'un demandeur reçoit un message de type **Rejet**, il va rechercher un autre détenant adéquat en consultant toujours sa table de routage.

Procédure N°5 : **Libérer la Section Critique**

```
1  Début
2  |  Demandeur_i ⟵ Faux;
3  |  Si tete(Suivant_i) ≠ NIL Alors
4  |  |  j ⟵ tete(Suivant_i);
5  |  |  Suivant_i ⟵ Suivant_i-{j};
6  |  |  File ⟵ Suivant_i ;
7  |  |  Envoyer Accord (Jeton,File) à j;
8  |  |  Jeton_i ⟵ Faux;
9  |  Sinon
10 |  |  Modification dans la table de routage AODV_i (1,1);
11 |  Finsi
12 Fin
```

lorsqu'un site libère la section critique, il envoi le jeton vers le premier site demandeur dans la file d'attente,si la file d'attente est vide le jeton reste détenu par ce site donc une mise-à-jour est faite dans la table de routage.

Procédure N°6 : **Modification dans la table de routage** $AODV_i$ **(X,Y :entiers)**

```
1  Début
2  |  AODV_i_[i]_Jeton_libre ⟵ X;
3  |  AODV_i_[i]_Jeton_present ⟵ Y;
4  Fin
```

La modification dans la table de routage porte sur deux variables (l'existence de jeton libre et l'existence de jeton présent) selon les valeurs de X et Y.

Procédure N°7 : **Chercher le détenant adéquat(**$AODV_i$**)**

```
1  Début
2      X ⟵ ∞;
3      Pour Q=1 à nombre de nœuds faire
4          Si AODV_i_[Q]_Jeton_libre = 1 Alors
5              Min ⟵ AODV_i_[Q]_Distance;
6              Si Min < X Alors
7                  detenant ⟵ Q;
8                  X ⟵ Min;
9              Finsi
10         Finsi
11     Finpour
12     Si X = ∞ Alors
13         Pour Q=1 à nombre de nœuds faire
14             Si AODV_i_[Q]_Jeton_present = 1 Alors
15                 Min ⟵ AODV_i_[Q]_Distance;
16                 Si Min < X Alors
17                     detenant ⟵ Q;
18                     X ⟵ Min;
19                 Finsi
20             Finsi
21         Finpour
22     Finsi
23     Retourner(detenant);
24 Fin
```

Cette procédure représente la stratégie qui nous permet de trouver le détenant adéquat dans la table de routage avec le minimum de sauts.

2.3 Discussion des résultats de simulation

2.3.1 Les paramètres de simulation

le but de la simulation est d'étudier la performance de l'algorithme proposé , et pour spécifier les paramètres qui influent sur la performance de notre algorithme, pour cela nous allons changer des paramètres à chaque fois .
nous allons étudier deux cas :

- les détenteurs initiaux des jetons sont considérés demandeurs de la section critique dès le début "la courbe rouge".
- les détenteurs initiaux des jetons ne demandent pas la section critique des le début de simulation "la courbe bleue".
- **Scénario 1** : Variation du nombre de requêtes
  Dans ce scénario nous avons varié le nombre de requêtes entre 3 et 18.

- **Scénario 2** : Variation de la portée de communication
  Nous avons varié la portée de communication entre 50 et 500 mètres.

- **Scénario 3** : Variation du nombre de nœuds
  Ce scénario a connu la variation du nombre de nœuds entre 10 et 50.

- **Scénario 4** : Variation du nombre de ressources
  Cette variation porte sur le nombre de ressources qui se varient entre 3 et 15.

Nous pouvons illustrer les cinq scénarios dans la figure 2.2 ci-dessous.

	Nombre de requêtes	La portée de communication	Le nombre de nœuds	Le nombre de ressources
Scénario 1	3 ≤ D ≤ 18	200	50	5
Scénario 2	10	50 ≤ C≤ 500	50	5
Scénario 3	10	200	10 ≤ N ≤ 50	5
Scénario 4	10	200	50	3 ≤ R ≤ 15

Figure 2.2 – *Variation des paramètres de simulation.*

2.4 Résultats et interprétations

Notre algorithme a été validé par une simulation qui a utilisé les scénarios précédents. Cette simulation nous a permis de tirer un ensemble de résultats intéressants.
durant toute la simulation la courbe rouge est pour le cas où les détenteurs initiaux des jetons sont considérés demandeurs de la section critique dès le début,et la courbe bleue est pour le cas où les détenteurs initiaux des jetons ne demandent pas la section critique dès le début de simulation.

2.4.1 Variation du nombre de requêtes

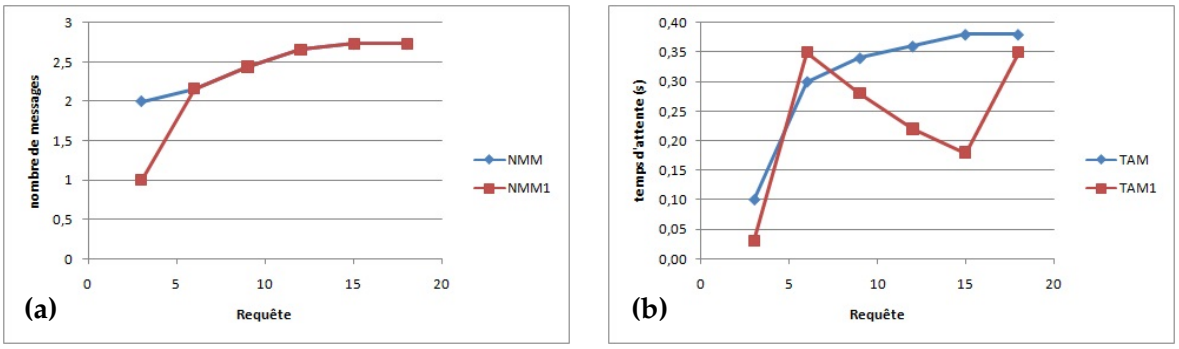

Figure 2.3 – *Influence du nombre de requêtes sur le NMM et le TAM.*

Dans la figure **(a)** pour la courbe rouge, on remarque une augmentation du NMM avec l'augmentation du nombre de requêtes, cela peut être justifié par le nombre de messages échangés par chaque site lors de demande .
On remarque que le NMM prend la valeur 1 avec 4 requêtes , c'est tout à fait logique car les sites demandeurs sont les mêmes sites qui détiennent les jetons , donc on aura pas un échange considérable de message.
En général , on peut dire que la courbe est stable (la valeur minimale = 1 et la valeur maximale est 2,7) ceci est obtenu grâce à la stratégie utilisée dans notre algorithme qui permet de choisir le détenteur le plus proche , et donc assure l'accès à la section critique par un échange réduit de messages.
dans la figure **(b)** on constate une augmentation du TAM jusqu'à stabilisation dans la dernière phase pour la courbe bleue , l'augmentation est plus rapide au début pour la courbe rouge parce que les détenteurs initiaux sont considérés demandeurs de la section critique la ré-augmentation de la courbe rouge est à cause du nombre de requêtes qui devient élevé.

2.4.2 Variation de la portée de communication

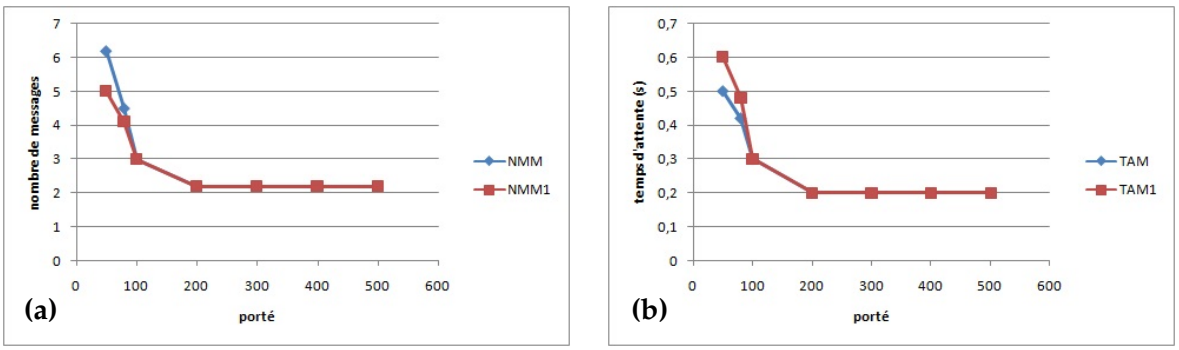

Figure 2.4 – *Influence de la portée de communication sur le NMM et le TAM.*

nous avons le même comportement pour les deux courbes, nous constatons bien que dans la figure **(a)** une diminution du NMM avec l'augmentation de la portée jusqu'à ce qu'il devient constant,et cela parce que les sites auront de plus en plus des voisins directe donc moins de messages transités.
la même chose dans la figure **(b)**, le résultat est clair, une diminution du TAM après chaque

augmentation de la portée, cela est dû à l'augmentation du nombre de voisins immédiats par l'augmentation de la portée.

2.4.3 Variation du nombre de ressources

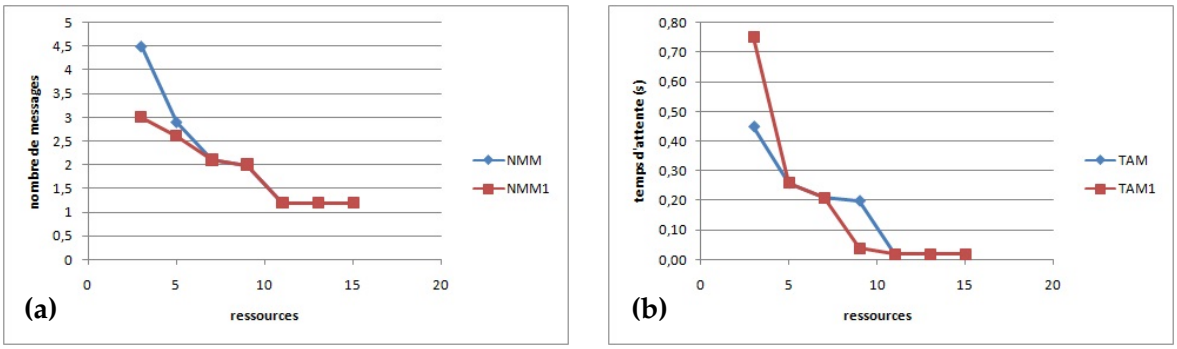

Figure 2.5 – *Influence du nombre de ressources sur le NMM et le TAM.*

Dans la figure **(a)**même comportement pour les deux courbes on remarque une diminution du NMM avec l'augmentation du nombre de ressources, car avec l'augmentation du nombre de ressources les sites auront plus de choix donc évidement ils vont choisir les ressources les plus proches donc moins de messages transités , la courbe se stabilise lorsque le nombre de ressources devient supérieur au nombre de demande.

dans la figure **(b)** on constate une diminution du TAM avec l'augmentation du nombre de ressources, cela est justifié par la disponibilité des ressources (qui dépasse même le nombre de demandes dans la dernière phase) qui permet des entrées en SC avec le minimum de temps d'attente.

2.4.4 Variation du nombre de sites

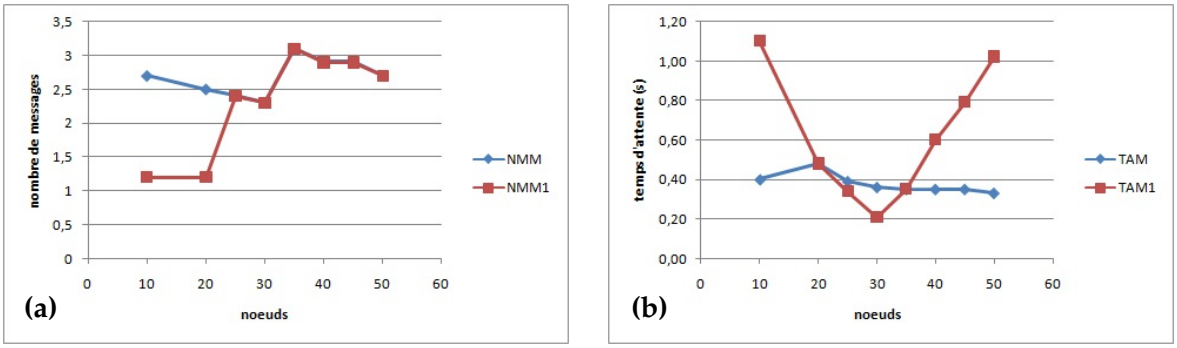

Figure 2.6 – *Influence du nombre de sites sur le NMM et le TAM.*

Dans la figure **(a)**,on remarque que lorsque les détenteurs initiaux ne sont pas demandeur de la section critique au début le comportement de la courbe est plus stable vu le taux élevé de requêtes reçues durant la section critique dans le cas inverse .

Dans la figure **(b)**, on constate que la demande immédiate des détenteurs initiaux a eu a effet négatif sur le TAM vu la durée que prend la section critique , la courbe est beaucoup plus stable dans le cas inverse.

2.5 Taux d'utilisation des jetons

Afin d'étudier la performance de notre algorithme, on veut vérifier le mouvement des jetons et le taux d'utilisation de chaque jeton, pour cela, nous avons identifié les jetons, et nous avons calculé le nombre de requêtes servies par chaque jeton.
Les résultats obtenus sont présentés dans les courbes suivantes.

Figure 2.7 – *10 requêtes.*

Figure 2.8 – *20 requêtes.*

Figure 2.9 – *10 requêtes.*

On remarque bien dans les courbes que le taux d'utilisation des jetons et équilibrée, et on remarque que le nombre de requêtes servies par chaque jeton est très proche des autres, cela peut être justifié par la disponibilité de l'information dans les tables de routage, et la méthode de choix judicieuse qui permet de choisir le jetons disponible le plus proche.

Conclusion

D'après ces courbes on peut dire qu'on est arrivé à une performance très acceptable surtout concernant le temps d'attente puisque le principe de recherche de jeton est basé en premier sur la recherche du jeton libre le plus proche sinon le jeton présent le plus proche et cela assure la minimisation du temps d'attente .
notons que le nombre de messages est le nombre effectif de messages c'est-à-dire le nombre de sauts entre source destination et non pas le nombre de message logique c'est-à-dire ce que les sites envoient .

3 Un algorithme tolérant aux fautes basé sur le protocole de routage AODV

Sommaire

Ce chapitre est dédié à la description d'un mécanisme de tolérance aux fautes et l'intégration de ce mécanisme dans l'algorithme pré-proposé pour la K-exclusion mutuelle dans les réseaux mobiles AD HOC.

3.1 INTRODUCTION

La méthode la plus utilisée dans la plupart des algorithmes de tolérances aux fautes est la duplication des données , le principe est simple , désigner un site particulier qui va recevoir un duplicata des données. Lors d'une panne, les données seront facilement récupérées.
Mais Les algorithmes utilisant cette méthode supportent une seule panne et la panne de plusieurs sites va causer l'arrêt du système.
Notre idée est basée sur la méthode de duplication mais en utilisant une hiérarchie de duplication pour tolérer plusieurs fautes.
on peut distinguer deux types de nœuds :

- Les nœuds simples.

- Les nœuds possesseurs des jetons (D) : ce sont les nœuds les plus importants car leurs panne va causer la pertes des jetons et des files d'attentes des demandeurs .

3.1.1 idée de base

le problème majeur d'une faute est la perte du jeton et de la file d'attente lorsque le site détenant un jeton tombe en panne.pour éviter cette perte, notre idée consiste à associer à chaque site détenant un jeton un site appelé le site sûr (S) , ce dernier garde une copie de la file d'attente , et peut même générer un jeton lorsque le site détenant le jeton tombe en panne, cette duplication permet d'éviter la perte, mais les nœuds sûrs (S) ne sont pas à l'abri des pannes, une panne d'un nœud S peut conduire le système à la perte des jetons et des files d'attentes, pour cela, on associe également à chaque nœud S un nœud back-to-back (B) ce dernier n'intervient que lorsqu'un site S tombe en panne.
le choix de S et B et fait selon deux critères :

- la distance par rapport au site D , évidement nous allons choisir les plus proches possible.
- les sites choisis ne doivent pas jouer le rôle de S et B pour un autre D .

3.1.2 Les cas possibles de pannes

- La panne du site détenant le jeton D.
- La panne d'un site sûr S.
- La panne d'un site back-to-back B.
- La panne du site détenant le jeton D et son site sûr S.
- La panne du site détenant le jeton D et son site sûr B.

3.1.3 Comportement de l'algorithme lors d'une panne

nous allons illustrer le comportement de l'algorithme lors de quelques cas de pannes

3.1.3.1 La panne du site détenant le jeton

Le nœud S doit :
- générer un jeton.
- Enlever l'identité du site défaillant de la file d'attente.
- Envoyer le jeton avec une copie de la file d'attente vers le premier site demandeur dans la file d'attente.
- Propager l'information au nœud B.

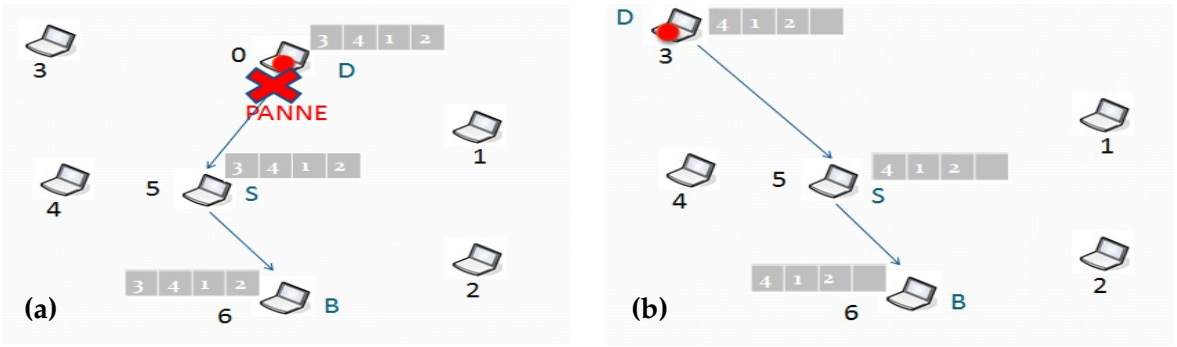

Figure 3.1 – *Comportement de l'algorithme lors d'une panne de site D .*

3.1.3.2 La panne d'un site sûr

Le nœud B du défaillant doit :
- choisir un site voisin, et le considère son back-to-back.
- Envoyer une copie de la file d'attente vers le nouveau site B.
- L'ancien site B devient un site S.

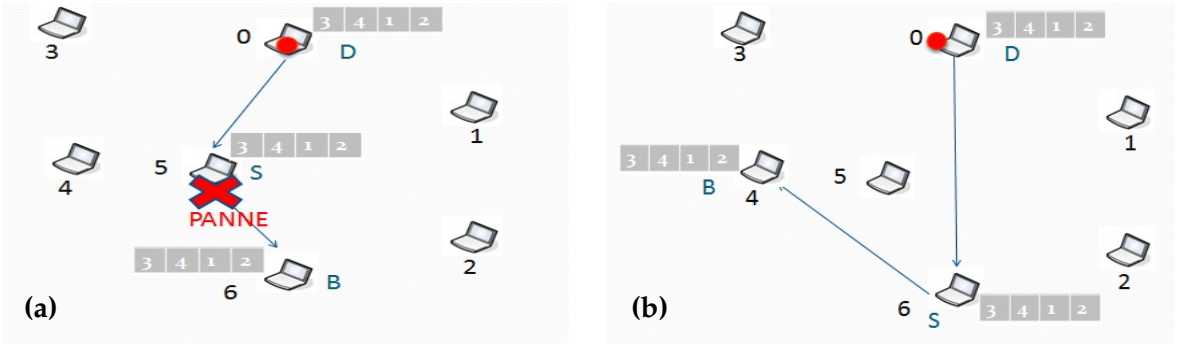

Figure 3.2 – *Comportement de l'algorithme lors d'une panne de site S .*

3.1.3.3 La panne du site détenant le jeton et son site sûr

Le nœud B du site S défaillant doit
- choisir un site voisin, et le considère son back-to-back.
- Enlever les identités des sites défaillants de la file d'attente, et envoyé une copie vers le nouveau site B.
- L'ancien site B devient un site S.
- Le nouveau site S génère un jeton, et l'envoie avec une copie de la file d'attente vers le premier site demandeur dans la file d'attente.

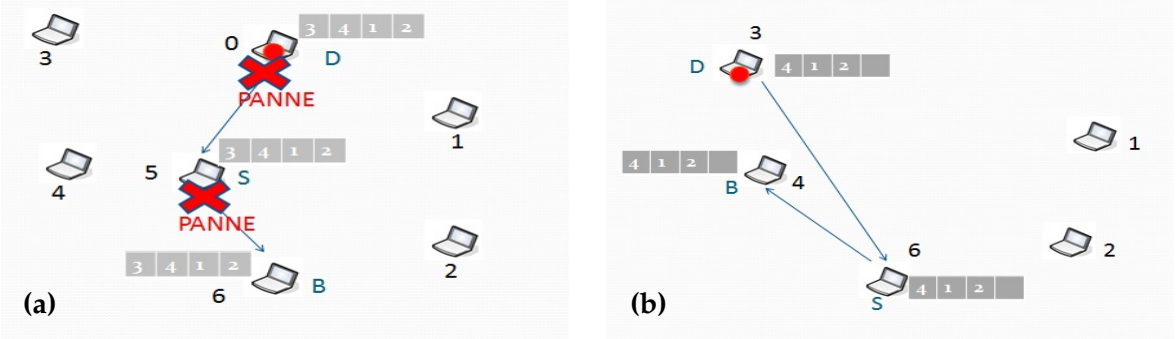

Figure 3.3 – *Comportement de l'algorithme lors d'une panne du site D et du site S .*

3.1.3.4 La panne d'un site back-to-back

Le site S doit :
- choisir un autre voisin et le considère son back-to-back.
- Envoyer une copie de la file d'attente vers le nouveau site B.

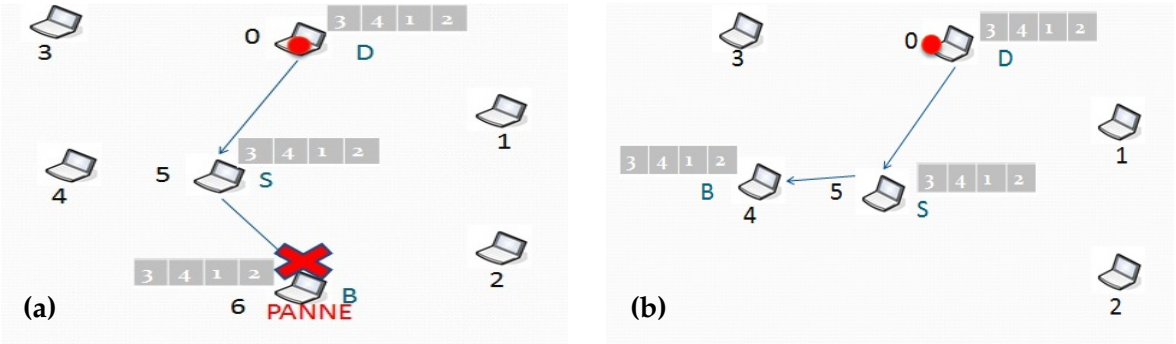

Figure 3.4 – *Comportement de l'algorithme lors d'une panne du site D et du site S .*

3.2 Intégration du mécanise de tolérance aux fautes dans l'algorithme de la K-exclusion mutuelle :

L'idée de ce nouvel algorithme consiste à ajouter deux autres champs dynamiques dans la table de routage de chaque site, l'un pour l'identité du site S et l'autre pour l'identité du site B qui sont propres pour chaque site.

puisque lors de l'enchainement de l'algorithme il y a la possibilité qu'un site qui détient le jeton joue le rôle d'un site S ou B nous allons utiliser trois variables locales propres à

chaque site idD_i , idS_i ,idB_i .
donc une site qui détient le jeton gère les deux champs de la table de routage et les sites jouant le rôle de S ou B gèrent les trois variables locales et dans le cas où un site joue les deux rôles il gère les champs de la table de routage et les trois variables.
donc le site S doit garder l'identité du site détenant le jeton idD_i et le site B idB_i .
les identités à garder dans le site B sont détenant le jeton idD_i ainsi que celui du site S idS_i.

3.2.1 Description de l'algorithme

Nous allons décrire les changements effectués au niveau des variables locales, les messages utilisés, et enfin les procédures de l'algorithme.

3.2.1.1 Les variables locales

Chaque nœud ***i*** dans le système maintient les variables locales suivantes :

- $Demandeur_i$: une variable booléenne initialisée à **Faux**, indiquant si ***i*** a demandé une ressource critique ou non.
- $Jeton_i$: une variable booléenne indiquant si ***i*** possède un jeton ou non, elle est initialisée à **Faux** au niveau des sites simples et à **Vrai** au niveau des sites D.
- $Suivant_i$: une file d'attente dont on va mettre les identités des sites qui demandes la section critique
- $tolerance_i$: une file d'attente utilisée lorsque le site joue le rôle S ou B
- idD_i : l'identité du site détenant le jeton D
- idS_i : l'identité du site S
- idD_i : l'identité du site B
- $AODV_i$(X,Y) : indique les deux champs pouvant être manipulés par notre algorithme :
  X :indique le jeton libre.
  Y :indique le jeton présent.

3.2.1.2 Les messages utilisés

Dans cet algorithme, on utilise trois types de messages :

- **Requête (Jeton, i)** : envoyé par le site ***i*** vers le site qui détient un jeton lors de la demande de la Section critique.

- **Accord (Jeton,file)** : envoyé par le site qui détient le jeton à un site demandeur pour lui permettre d'utiliser la ressource critique demandée.

- **Rejet ()** : envoyé par un ancien détenteur du jeton à un site demandeur pour lui informer qu'il ne détient plus le jeton.

- **Panne (i)** : envoyé par un site lors de détection d'une panne.

- **MAJtolerance (file)** : envoyé par un détenteur du jeton à son site S lorsque il reçoit une requête.

– **MAJSB (d,s,b)** : envoyé par l'un des sites D ,S ou B aux deux autres lors d'une mise à jour.

3.2.1.3 Initialisation des variables

Initialisation : des variables des sites D

$Demandeur_i$: Variable booléenne.
$Jeton_i$: Variable booléenne.
$Suivant_i$: file d'attente initialisé à NIL
$tolerance_i$: file d'attente initialisé à NIL
idD_i , idD_i , idD_i ,S,B : Variables entières.

Début
$Demandeur_i \longleftarrow$ Faux;
$Suivant_i \longleftarrow$ NIL;
$tolerance_i \longleftarrow$ NIL;
$Jeton_i \longleftarrow$ Vrai;
$S \longleftarrow choisirS()$;
$B \longleftarrow choisirB()$;
$AODV_i\_[i]\_S \longleftarrow S$;
$AODV_i\_[i]\_B \longleftarrow B$;
$AODV_i\_[i]\_Jeton\_libre \longleftarrow 1$;
$AODV_i\_[i]\_Jeton\_present \longleftarrow 1$;
Envoyer **MAJSB** (*i*,S,B) à S;
Envoyer **MAJSB** (*i*,S,B) à B;
Fin

Initialisation : des variables des sites simples

$Demandeur_i$: Variable booléenne.
$Jeton_i$: Variable booléenne.
$Suivant_i$: file d'attente initialisé à NIL
$tolerance_i$: file d'attente initialisé à NIL
idD_i , idD_i , idD_i ,S,B : Variables entières.

Début
$Demandeur_i \longleftarrow$ Faux;
$Suivant_i \longleftarrow$ NIL;
$tolerance_i \longleftarrow$ NIL;
$Jeton_i \longleftarrow$ Faux;
$AODV_i\_[i]\_S \longleftarrow$ -1;
$AODV_i\_[i]\_B \longleftarrow$ -1;
$AODV_i\_[i]\_Jeton\_libre \longleftarrow 0$;
$AODV_i\_[i]\_Jeton\_present \longleftarrow 0$;
Fin

3.2.1.4 Les procédures de l'algorithme

Procédure N°1 : **Demander la Section Critique**

```
1  Début
2  |  Demandeur_i ⟵ Vrai;
3  |  Si (AODV_i_[i]_Jeton_libre = 0) Alors
4  |  |  detenant ⟵ Chercher le détenant adéquat(AODV_i);
5  |  |  Envoyer Requête (Jeton, i) à detenant;
6  |  Finsi
7  |  Attendre (Jeton = Vrai);
8  |  Modification dans la table de routage AODV_i (0,1);
9  |  < Entrer en Section Critique >
10 Fin
```

Quand un nœud qui ne possède pas un jeton désire entrer en SC, il devient demandeur et il cherche le détenant adéquat afin de lui envoyer une requête et attend la réception du jeton pour entrer en SC.

Procédure N°2 : **Réception de Accord (Jeton,File)**

```
1  Début
2  |  Jeton_i ⟵ Vrai;
3  |  Suivant_i ⟵ File;
4  Fin
```

Lorsqu'un site reçoit un message de type **Accord**, il va modifier sa variable booléenne Jeton à Vrai et il met à jour la file d'attente.

Procédure N°3 : **Réception de Requête (Jeton, j)**

Début
  Si ($AODV_i$_[i]_$Jeton_libre = 1$) **Alors**
    $Jeton_i \longleftarrow$ Faux;
    Envoyer **MAJSB** (j,i,$AODV_i$_[i]_S) à $AODV_i$_[i]_S;
    Envoyer **MAJSB** (-1,-1,-1) à $AODV_i$_[i]_B;
    $idB_i \longleftarrow AODV_i$_[$i$]_$S$;
    $idS_i \longleftarrow i$;
    $idD_i \longleftarrow$ j ;
    Envoyer **Accord** (Jeton,$Suivant_i$) à j;
    Envoyer **MAJSB** (idD_i,idS_i,idB_i) à j;
    Modification dans la table de routage $AODV_i$ (o,o);
  Sinon
    Si ($AODV_i$_[i]_$Jeton_present = 1$) **Alors**
      $Suivant_i \longleftarrow Suivant_i + \{j\}$;
      Envoyer **MAJtolerace** ($Suivant_i$) à $AODV_i$_[i]_S;
    Sinon
      Envoyer **Rejet** () à j;
    Finsi
  Finsi
Fin

Si un site reçoit un message de type **Requête**,il va envoyer le jeton vers le site demandeur s'il n'utilise pas la ressource,sinon il le met en attente . Dans le cas où le site ne détient pas de jeton, il va lui envoyer un message de type **Rejet** lui indiquant qu'il ne possède plus le jeton.

Procédure N°4 : **Réception de Rejet ()**

Début
  detenant $\longleftarrow$ Chercher le détenant adéquat($AODV_i$);
  Envoyer **Requête** (Jeton, i) à *detenant*;
Fin

Lorsqu'un demandeur reçoit un message de type **Rejet**, il va rechercher un autre détenant adéquat en consultant toujours sa table de routage.

Procédure N°5 : **Libérer la Section Critique**

Début
  $Demandeur_i \longleftarrow$ Faux;
  Si $tete(Suivant_i) \neq NIL$ **Alors**
    $j \longleftarrow tete(Suivant_i)$;
    $Suivant_i \longleftarrow Suivant_i$-$\{j\}$;
    $Jeton_i \longleftarrow$ Faux;
    Envoyer **MAJSB** (j,$\boldsymbol{i}$,$AODV_i$_[$\boldsymbol{i}$]_S) à $AODV_i$_[$\boldsymbol{i}$]_S;
    Envoyer **MAJSB** (-1,-1,-1) à $AODV_i$_[$\boldsymbol{i}$]_B;
    $idB_i \longleftarrow AODV_i$_[$\boldsymbol{i}$]_$S$;
    $idS_i \longleftarrow \boldsymbol{i}$;
    $idD_i \longleftarrow$ j ;
    Envoyer **Accord** (Jeton,$Suivant_i$) à j;
    Envoyer **MAJSB** (idD_i,idS_i,idB_i) à j;
    Envoyer **MAJtolerance** ($Suivant_i$) à idB_i;
    Modification dans la table de routage $AODV_i(0,0)$;
  Sinon
    Modification dans la table de routage $AODV_i(1,1)$;
  Finsi
Fin

lorsqu'un site libère la section critique, il envoi le jeton vers le premier site demandeur dans la file d'attente,si la file d'attente est vide le jeton reste détenu par ce site donc une mise-à-jour est faite dans la table de routage.

Procédure N°6 : **Modification dans la table de routage** $AODV_i$ **(l,P :entiers)**

Début
  $AODV_i$_[$\boldsymbol{i}$]_$Jeton_libre \longleftarrow L$;
  $AODV_i$_[$\boldsymbol{i}$]_$Jeton_present \longleftarrow P$;
Fin

La modification dans la table de routage porte sur deux variables (existence de jeton libre et l'existence de jeton présent) selon les valeurs de L et P.

Procédure N°7 : Chercher le détenant adéquat($AODV_i$)

```
1  Début
2      X ⟵ ∞;
3      Pour Q=1 à nombre de nœuds faire
4          Si AODV_i_[Q]_Jeton_libre = 1 Alors
5              Min ⟵ AODV_i_[Q]_Distance;
6              Si Min < X Alors
7                  detenant ⟵ Q;
8                  X ⟵ Min;
9              Finsi
10         Finsi
11     Finpour
12     Si X = ∞ Alors
13         Pour Q=1 à nombre de nœuds faire
14             Si AODV_i_[Q]_Jeton_present = 1 Alors
15                 Min ⟵ AODV_i_[Q]_Distance;
16                 Si Min < X Alors
17                     detenant ⟵ Q;
18                     X ⟵ Min;
19                 Finsi
20             Finsi
21         Finpour
22     Finsi
23     Retourner(detenant);
24 Fin
```

Cette procédure représente la stratégie qui nous permet de trouver le détenant adéquat dans la table de routage avec le minimum de sauts.

Procédure N°8 : **MAJtolerance (file)**

```
1  Début
2      Si (idS_i = i) ) Alors
3          tolerance_i ⟵ file ;
4          Envoyer MAJtolerance (tolerance_i) à idB_i;
5      Sinon
6          Si (idB_i = i) Alors
7              tolerance_i ⟵ file ;
8          Finsi
9      Finsi
10 Fin
```

procédure appelé à chaque modification au niveau de la file d'attente des suivant et lors d'un nouveau choix des sites S ou B.

Procédure N°9 : **MAJSB (d,s,b)**

```
1  Début
2      Si AODV_i_[i]_Jeton_present ≠ 1 Alors
3          idD_i ⟵ d;
4          idS_i ⟵ s;
5          idB_i ⟵ b;
6          Si d = -1 Alors
7              AODV_i _[i]_B ⟵ -1;
8          Sinon
9              Si (i = s) Alors
10                 AODV_i_[i]_B ⟵ -1;
11                 AODV_i_[i]_S ⟵ -2;
12             Sinon
13                 AODV_i_[i]_B ⟵ -2;
14                 AODV_i_[i]_S ⟵ -1;
15             Finsi
16         Finsi
17     Sinon
18         AODV_i_[i]_B ⟵ b;
19         AODV_i_[i]_S ⟵ s;
20     Finsi
21 Fin
```

Cette procédure a comme rôle la maintenance des informations nécessaires à la tolérance aux fautes.

Procédure N°10 : choisirS()

```
1  Début
2      min1 ⟵ ∞;
3      mon_S ⟵ -1;
4      Pour Q=1 à nombre de nœuds faire
5          Si (AODV_i_[Q]_S = −2)et(AODV_i_[Q]_B = −2) Alors
6              Si AODV_i_[Q]_Jeton_libre = 1 Alors
7                  Min ⟵ AODV_i_[Q]_Distance;
8                  Si Min < min1 Alors
9                      mon_S ⟵ Q;
10                     min1 ⟵ Min;
11                 Finsi
12             Finsi
13         Finsi
14     Finpour
15     Si mon_S ≠ -1 Alors
16         Retourner(monS);
17     Sinon
18         choisirS();
19     Finsi
20 Fin
```

Procédure représentant la stratégie de choix du site S

Procédure N°11 : choisirB()

```
1  Début
2      min1 ⟵ ∞;
3      mon_B ⟵ -1;
4      Pour Q=1 à nombre de nœuds faire
5          Si (AODV_i_[Q]_S = −2)et(AODV_i_[Q]_B = −2) Alors
6              Si AODV_i_[Q]_Jeton_libre = 1 Alors
7                  Min ⟵ AODV_i_[Q]_Distance;
8                  Si Min < min1 Alors
9                      mon_B ⟵ Q;
10                     min1 ⟵ Min;
11                 Finsi
12             Finsi
13         Finsi
14     Finpour
15     Si mon_S ≠ -1 Alors
16         Retourner(monB);
17     Sinon
18         choisirB();
19     Finsi
20 Fin
```

Procédure représentant la stratégie de choix du site B

3.2.1.5 La détection d'une panne

"La détection de pannes est un aspect important de la programmation distribuée. il est en effet difficile de déterminer si un nœud est non fonctionnel ou simplement lent. Or, la distinction est importante pour une application distribuée : dans le premier cas une configuration est nécessaire, dans le second, une simple attente suffit souvent. Implémenter une détection de pannes efficace est un problème difficile, ce qui fait qu'en général, les programmes utilisent des approches simplistes comme des temps limites fixés.
Pour contrer ce problème, la notion de service de détection de panne a été proposée depuis plusieurs années. Un tel service peut utiliser des techniques complexes sans compliquer les applications. ces services ont bien suivi la technologie récente on est arrivé a détecter tout genre de pannes sauf les cassures brutales".[Wie12]
dans notre algorithme nous supposons que la panne à traiter est détectable facilement par exemple le déchargement de la batterie.

Procédure N°12 : demander l'intervention lors de détection d'une panne

```
1  Début
2    Si (AODV_i_[i]_Jeton_present = 1 )Ou(idS_i ≠ −1)Ou(idB_i ≠ −1 ) Alors
3      Si (idS_i ≠ −1)Ou(idB_i ≠ −1) Alors
4        Si (idS_i = i Alors
5          Destination1 ⟵ idB_i ;
6        Sinon
7          Destination1 ⟵ idS_i ;
8        Finsi
9      Finsi
10     Si (AODV_i_[i]_Jeton_present = 1) Alors
11       Destination ⟵ AODV_i_[i]_S;
12     Sinon
13       Si (idS_i = i) Alors
14         Destination ⟵ idB_i;
15       Sinon
16         Si (idB_i = i) Alors
17           Destination ⟵ idS_i;
18         Finsi
19       Finsi
20     Finsi
21     Si (Destination1 ≠ −1) Alors
22       Envoyer panne(i) à Destination1;
23     Finsi
24     Envoyer panne(i) à Destination;
25   Finsi
26 Fin
```

Cette procédure représente le comportement d'un site lors de détection de panne.

Procédure N°12 : réception d'une panne(Q)

```
1  Début
2      Si (idS_i = i) Alors
3          Si (idD_i = Q) Alors
4              Si (tolerance_i ≠ NIL) Alors
5                  j ⟵ tete(tolerance_i) ;
6                  tolerance_i ⟵ tolerance_i-{j};
7                  Envoyer Accord (Jeton,tolerance_i) à j;
8                  idD_i ⟵ j;
9                  Envoyer MAJtolerance (tolerance_i) à idB_i;
10                 Envoyer MAJSB (idD_i,idS_i,idB_i) à idD_i;
11             Sinon
12                 Jeton_i ⟵ Vrai;
13                 AODV_i_[i]_Jeton_libre ⟵ 1;
14                 AODV_i_[i]_Jeton_present ⟵ 1;
15                 AODV_i_[i]_S ⟵ choisir_S();
16                 AODV_i_[i]_B ⟵ idB_i idD_i ⟵ -1;
17                 idS_i ⟵ -1;
18                 idB_i ⟵ -1;
19                 Envoyer MAJSB (i,AODV_i_[i]_S,AODV_i_[i]_B) à AODV_i_[i]_S;
20                 Envoyer MAJSB (i,AODV_i_[i]_S,AODV_i_[i]_B) à AODV_i_[i]_B;
21             Finsi
22         Sinon
23             Si (idD_i = Q) Alors
24                 idB_i ⟵ choisir_B();
25                 Envoyer MAJSB (idD_i,idS_i,idB_i) à idB_i;
26                 Envoyer MAJSB (idD_i,idS_i,idB_i) à idD_i;
27             Finsi
28         Finsi
29     Sinon
30         Si (idS_i = Q) Alors
31             B ⟵ choisirB();
32             idS_i ⟵ idB_i;
33             idB_i ⟵ B;
34             Envoyer MAJSB (idD_i,idS_i,idB_i) à idB_i;
35             Envoyer MAJSB (idD_i,idS_i,idB_i) à idD_i;
36         Finsi
37     Finsi
38 Fin
```

Cette procédure est le cœur du mécanisme de tolérance aux pannes elle représente le comportement de notre algorithme lors d'une panne.

3.3 Discussion des résultats de simulations

3.3.1 Les paramètres de simulation

Nous avons changé à chaque fois le nombre de nœuds et de ressources, le nombre de requêtes, la portée de communication ainsi que la vitesse de déplacement et le nombre de pannes, ces changements permettent d'avoir différents scénarios.

- **Scénario 1** : Variation du nombre de requêtes
  Dans ce scénario nous avons varié le nombre de requêtes entre 3 et 18.

- **Scénario 2** : Variation de la portée de communication
  Nous avons varié la portée de communication entre 50 et 500 mètres.

- **Scénario 3** : Variation du nombre de nœuds
  Ce scénario a connu la variation du nombre de nœuds entre 10 et 50.

- **Scénario 4** : Variation du nombre de ressources
  Cette variation porte sur le nombre de ressources qui se varient entre 3 et 15.

- **Scénario 5** : variation du nombre de pannes
  Le nombre de panne pannes va être varier entre 2 et 10.

Nous pouvons illustrer les cinq scénarios dans la figure 3.5 ci-dessous.

	Nombre de requêtes	La portée de communication	Le nombre de nœuds	Le nombre de ressources	Le nombre de pannes
Scénario 1	3 ≤D ≤ 18	200	50	5	4
Scénario 2	10	50 ≤ C ≤ 500	50	5	4
Scénario 3	10	200	20 ≤ N ≤ 50	5	4
Scénario 4	10	200	50	3 ≤ R ≤ 15	4
Scénario 5	10	200	50	5	2 ≤ P≤ 10

Figure 3.5 – *Variation des paramètres de simulation.*

3.3.1.1 Variation du nombre de requêtes

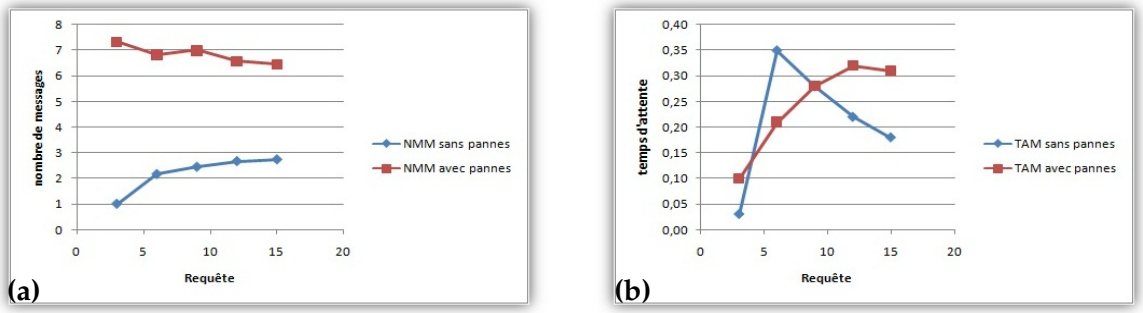

Figure 3.6 – *Influence du nombre de requêtes sur le NMM et le TAM.*

Dans les courbes **(a)**, c'est claire que le nombre de messages moyen est bien plus élevé pour l'algorithme avec tolérance aux pannes a cause des différentes mise-à-jours nécessaire a la tolérance.

Dans les courbes **(b)**, dans la première partie l'existence de panne a eu un effet positif pour certain sites puisque si un site détenant de jeton tombe en panne on transmet le jeton vers le demandeurs en attente , puis dans la deuxième phase nous avons des temps d'attente supérieure que l'algorithme sans tolérance et c'est tout a fait logique vu le temps que prend le re-fonctionnement après panne

3.3.1.2 Variation de la portée de communication

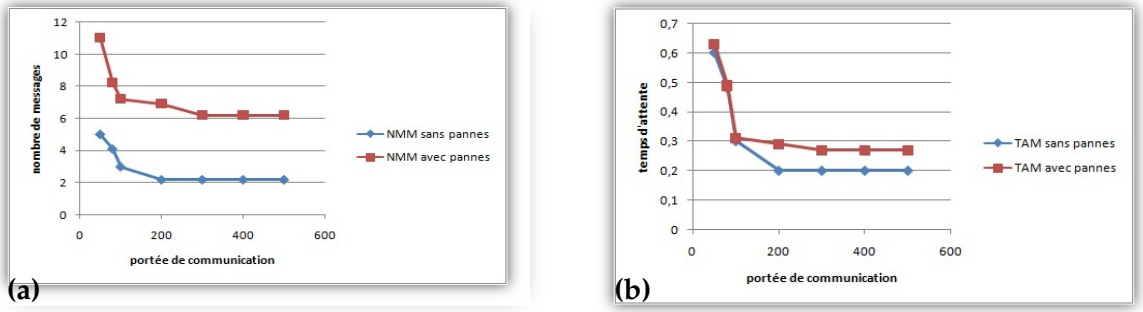

Figure 3.7 – *Influence de la portée de communication sur le NMM et le TAM.*

d'après la variation de la portée de communication, nous constatons bien que dans les courbes **(a)** une diminution du NMM toujours avec plus de messages pour l'algorithme avec tolérance aux pannes.
la même chose dans les courbe **(b)**, toujours avec une durée d'attente plus élevé pour l'algorithme avec tolérance aux pannes vu l'existence de pannes .

3.3.1.3 Variation du nombre de ressources

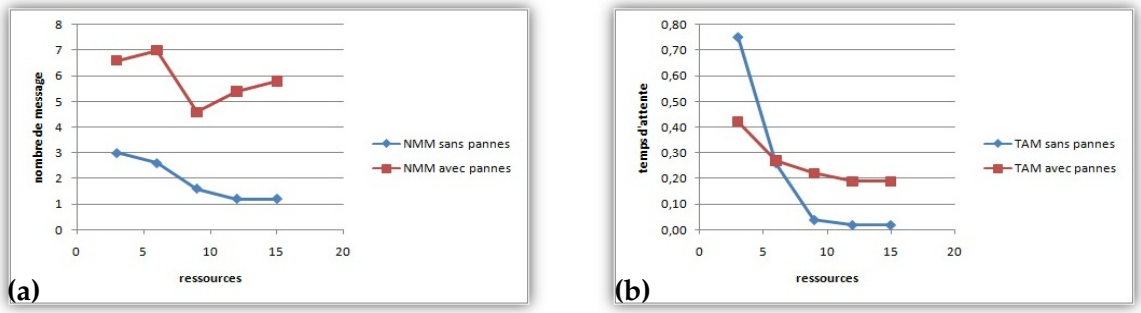

Figure 3.8 – *Influence du nombre de ressources sur le NMM et le TAM.*

dans les courbes **(a)** le comportement du NMM a une relation directe avec le nombre de panne des sites jouant le rôle de D ,S ou B.
(b), le temps d'attente est logiquement plus élevé pour l'algorithme avec tolérance aux pannes .

3.3.1.4 Variation du nombre de sites

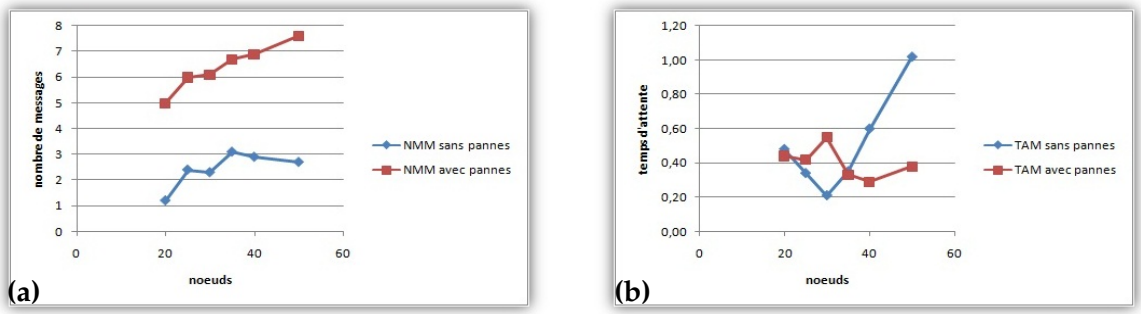

Figure 3.9 – *Influence du nombre de sites sur le NMM et le TAM.*

dans les courbes **(a)** on remarque l'augmentation du NMM avec le nombre de sites vu le nombre de mise a jours a chaque panne .
(b), le temps d'attente pour l'algorithme avec tolérance aux pannes a cause de la regénération immédiate après qu'un site D tombe en panne puisque la durée de la section critique est le paramètre le plus influant sur le temps d'attente.

3.3.1.5 Variation du nombre de pannes

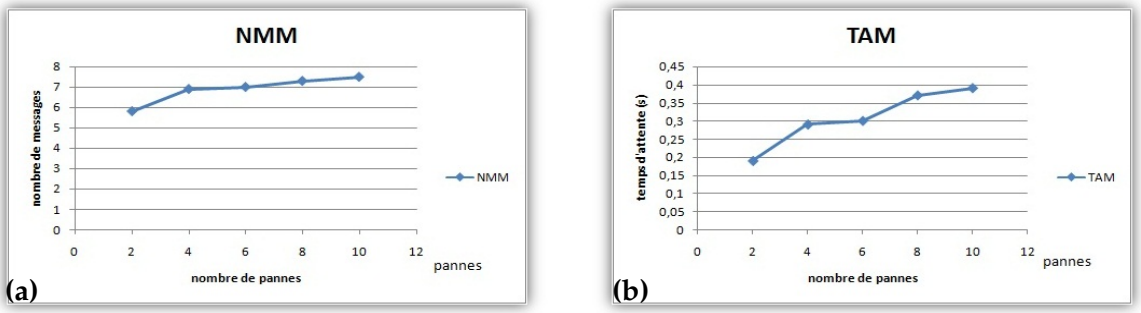

Figure 3.10 – *Influence du nombre de sites sur le NMM et le TAM.*

le nombre de messages et le temps d'attente augmentent logiquement avec le croisement du taux de pannes a cause des différentes mise à jours de tolérance au fautes.

Conclusion

D'après ces courbes on peut dire qu'on est arrivé à une performance très acceptable en terme de nombre de messages moyen et du temps d'attente moyen vu le gain de la tolérance aux fautes .

Conclusion et Perspectives

Dans ce mémoire nous avons proposé un algorithme tolérant aux fautes traitant le problème de la K exclusion mutuelle dans les réseaux mobiles Ad hoc Nous avons présenté des notions générales sur les systèmes repartis et les réseaux mobile Ad hoc ainsi que le problème de partage de ressources dans ces systèmes, la tolérance aux fautes et l'impacte de cette notion dans les algorithmes de partage de ressources a été également étudié.
Nous avons proposé et simuler un algorithme de la K exclusion mutuelle basé sur le protocole de routage AODV , et nous avons ensuite l'améliorer par l'ajout d'un nouveau mécanisme de tolérance aux fautes, ces algorithmes ont été validés par des simulations qui ont donné des résultats très acceptables.
Ce mémoire nous a permis de :

- Etudier et comprendre le problème de partage de ressources dans les Manets.
- Avoir une idée sur la tolérance aux fautes, et sur les avantages qu'elle offert pour les systèmes répartis.
- Se familiariser avec l'outil de simulation NS-2, et même avoir la possibilité de simuler notre propre algorithme.

Comme perspectives de ce travail, nous pensons à :

1. Réduire encore le temps d'attente en utilisant des nouvelles stratégies d'échanges de messages.
2. Utiliser le principe de nos algorithmes pour résoudre ce problème dans d'autres systèmes tels que : les VANETs . . . etc.

Bibliographie

[Abd07] Z. Abdelhafidi. Points de reprise dans les systèmes répartis etude basée sur la simulation des protocoles cic assurant la propriété rdt. *Thèse Magistère de l'université Amar Telidji-Laghouat Spécialité informatique*, pages 11–27, 2007. (Cité page 18.)

[AK08] A. Alliliche and M. Kebir. Etude comparative des protocoles de points de reprise de type cic. *Projet de fin d'étude, Université de Laghouat*, 2008. (Cité pages 17 et 18.)

[All04] Krouba M. Allaoui, T. Etude des systèmes répartis et réalisation d'un simulateur des algorithmes répartis d'exclusion mutuelle. *Projet de fin d'étude, Université de Laghouat*, 2004. (Cité page 7.)

[All07] T. Allaoui. Une nouvelle solution du problème de la k-exclusion mutuelle dans les systèmes répartis. *Mémoire de Magister, Université de Laghouat*, 2007. (Cité page 14.)

[Cou94] G. Coulouris. Distributed systems. *Concept and Design, 2nd Ed Addison Wesley Pubishers Ltd.*, 1994. (Cité page 5.)

[Dij65] W. Dijkstra. Solution of a problem in concurrent programming control. *Communications of the ACM*, 1965. (Cité page 14.)

[Dio07] B. Dioum. Effets de la mobilite sur les protocoles de routage dans les reseaux ad hoc. *Projet de fin d'étude, Université de Tizi Ouzou*, 2007. (Cité page 66.)

[Eve04] C. Evequoz. Programmation répartie. *Ecole d'ingénieur du Canton de Vaud*, 2004. (Cité page 6.)

[Lem00] T. Lemlouma. Le routage dans les réseaux mobiles ad hoc. *Mini projet, Université d'Alger USTHB*, 2000. (Cité pages 8, 12 et 13.)

[LK09] A. Lahag and Kouidri. Etude des performances des algorithmes de l'exclusion mutuelle dans les réseaux ad hoc {Simulation par NS2}. *Projet de fin d'étude, Université de Laghouat*, 2009. (Cité pages xi et 69.)

[Man07] N. Mansouri. Protocole de routage multi chemin avec équilibrage de charge dans les réseaux mobiles ad hoc. *Projet de fin d'étude, Ecole supérieur des communications de Tunis, Tunisie*, 2007. (Cité page 11.)

[Mes98] P. Meskauskas. Mobile ad hoc networking. *Seminar on Telecommunications Technology, Helsinki*, 1998. (Cité pages 9 et 11.)

[Sop08] J. Sopena. Algorithmes d'exclusion mutuelle : tolérance aux fautes et adaptation aux grilles. *Thèse de doctorat de l'Université Pierre et Marie Curie-Paris VI*, 2008. (Cité pages 7 et 15.)

[Sri89] P.K. Srimani. Another distributed algorithm for multiple entries to a critical section. *Information Processing Letters 41*, pages 51–57, 1989. (Cité page 16.)

[Tan94] S. Tanenbum. Distributed operating systems. *Prentice Hall*, 1994. (Cité page 5.)

[Wei08] W. Weigang. A fault tolerant mutual exclusion algorithm for mobile ad hoc networks. *Department of Computing, The Hong Kong Polytechnic University*, 2008. (Cité page 16.)

[Wie12] M. Wiesmann. Service de détection de pannes avec snmp. *résumé de cours Université japonaise JAIST*, 2012. (Cité page 48.)

[WK97] J.E. Walter and S. Kini. Mutual exclusion on multihop, mobile wireless networks. *Texas A and M University College Station*, 1997. (Cité page 17.)

[WW01] J. WALTER and L. WELCH. A mutual exclusion algorithm for ad hoc mobile networks. *Department of Computer Science,Texas USA*, 2001. (Cité page 16.)

Annexe : Script de simulation A

Sommaire

A.1 LE SCRIPT TCL (RÉSEAU AD HOC)

```
# ======================================================================
#                        Definition des options                        #
# ======================================================================
set val(chan)         Channel/WirelessChannel  ;# type de canal
set val(prop)         Propagation/TwoRayGround ;# model de propagation
set val(netif)        Phy/WirelessPhy          ;# type d'interface reseau
set val(mac)          Mac/802_11               ;# type d'interface mac
set val(ifq)          CMUPriQueue              ;# type de la 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                       ;# Nbre max des Packets -> file
set val(seed)         0.0                      ;# grain random
set val(adhocRouting) AODV                      ;# protocole de routage
set val(sc) "/home/chakinf/ns-allinone-2.34/ns-2.34/tcl/mobility/scene/scenar1"
set val(stop)         20                       ;# temps de simulation(duree)
#======================================================================
#                       Definition des procedures                      #
#======================================================================
#Phy/WirelessPhy set RXThresh_ 1.92278e-06; #10 meters
#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
# modification dans la portee de signal 200 meters
Phy/WirelessPhy set RXThresh_ 8.91754e-10
# ======================================================================
#            Creation d'instance de simulateur et topologie            #
# ======================================================================
set ns_ [new Simulator]                         ;# nouvelle simulation
$ns_ use-newtrace                               ;# nouveau fichier trace
set topo [new Topography]                       ;# Nouvelle Topologie
# ======================================================================
#              Creation des fichiers trace pour NS et NAM              #
# ======================================================================
set tracefd      [open adhoc.tr w]
set namtrace     [open adhoc.nam w]
set bw nbr_msg_tps.txt                  ;# Ficher .txt ( resultat 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 Resultats de la Simulation            ."
   puts  $mesure "_____________________________________________________"
   puts  $mesure "============= Informations de Simulation ============"
   puts  $mesure "                                                     ."
   puts  $mesure "Le Nombre de Sites = $nbr"
   puts  $mesure "Le Nombre de Racines  R = $nbracine"
   puts  $mesure "Le Nombre de Requetes = $nbrequete"
   puts  $mesure "                                                     ."
   puts  $mesure "====================================================="
```

```
    puts  $mesure "================================="
    puts  $mesure "     Resultas pour chaque site :"
    puts  $mesure "================================="
 }
# ============================================================================
#             Procedure de Terminaison de la simulation                      #
# ============================================================================
proc finish {} {
  global ns_ nf f0 nbr nbrequete tpatt nmsg p mesure  ;# Variables globales
  for {set i 0} {$i < $nbr} {incr i} {
    set nmsg [expr [$p($i) set nb_message_]+$nmsg]    ;# Somme des messages
  }
    puts  $mesure "======================================================"
    puts  $mesure "                  Le NMM et le TAM.                   "
    puts  $mesure "======================================================"
    puts  $mesure "Messages total = $nmsg"  ;# Calcul du nombre moyen des msg
    puts  $mesure "                                                      "
    puts  $mesure "Le NMM = $nmsg/$nbrequete = [expr $nmsg/$nbrequete]   "
    puts  $mesure "                                                      "
    set   tpatt [expr $tpatt/$nbrequete]   ;# Calcul du TAM
    puts  $mesure "Le TAM = $tpatt"        ;# Affichage du TAM
    $ns_ flush-trace                       ;# Ecriture dans le fichier TEXT
    close $mesure                          ;# Fermer le fichier des resultat
    puts "running nam..."                  ;# Affichage de "Running NAM ..."
    exec  nam adhoc.nam &                  ;# lancer le NAM automatiquement
    exit 0                                 ;# Sortie de la procedure
}
# ============================================================================
#     Procedure d'enregistrement des temps d'attente dans le fichier .txt    #
# ============================================================================
proc record {p n mutex hour} {
    global f0 nbrequete tpatt mesure       ;# Variables globales
    set ns [Simulator instance]            ;# Instancier la commande NS
    set bw0 [$ns now]                      ;# bwo est le temps actuelle
    set bw0 [expr $bw0- [$p set dem]]      ;# bwo := bwo - Temps de la demande
    puts $mesure "Temps d'attente pour le site N [$n node-addr] =
                  $bwo ==>  (( Date_Requete = $hour ** Duree SC = $mutex ))"
    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                                ;# Variables globales
    set ns_ [Simulator instance]               ;# Instancier la commande NS
    set nowe [$ns_ now]                        ;# Le temps actuelle
    set nbrk [$p set nbracine]                 ;# nbrk := nbracine ;
    $ns_ at $nowe "tester $p $n $mutex $hour"  ;# execution de tester
    $ns_ at $nowe "$p demander-sc"             ;# execution de demander-sc
    $ns_ trace-annotate "[$n node-addr] demande la SC "  ;# noeud demandeur
}
# ============================================================================
#    Procedure de test d'entrer et la liberation de la Section Critique      #
# ============================================================================
proc tester {p n mutex hour} {
    global nbr  mesure                         ;# Variables globales
    set ns_ [Simulator instance]               ;# definir commande ns_
    set time 0.01                              ;# Temps de test pour SC
    set now [$ns_ now]                         ;# Definir now
    set a [$p set sc]                          ;# Affectation a := sc ;
```

```
    if { $a == 1} {                                    ;# Condition entrer en SC
       $ns_ at $now "record $p $n $mutex $hour"        ;# Record pour le TAM
       $ns_ at $now "$n label \"<SC> \""               ;# libele <SC> au noeud
       $ns_ at [expr $now+$mutex] "$n label \" \""     ;# Effacer <SC>
       $ns_ at [expr $now+$mutex] "$p liberation-sc";# Appel de liberation-sc
       $ns_ at [expr $now+$mutex] "$p set sc -1"       ;# sc := -1 pour sortir
     } else {                                          ;# noeud veut entrer SC
       $ns_ at [expr $now+$time] "$p attendre"         ;# attendre MAJ sc <- 1
       $ns_ at [expr $now+$time] "tester $p $n $mutex $hour"
     }
}

# ==================================================================================
#                          Le programme Principale                                 #
# ==================================================================================
# ==================================================================================
# ==================================================================================
#                Lecture des donnees a partir d'un fichier texte                   #
# ==================================================================================
set f [open "les_demandes.txt" "r"]          ;# lire (les_demandes.txt)
set nbr [gets $f]                            ;# (nbr) <- premier ligne
set nbracine [gets $f]                       ;# (nbracine) <- deuxiemme ligne
set nbrequete [gets $f]                      ;# (nbrequete) <- troisiemme ligne

for {set j 0} {$j < $nbrequete} {incr j} { ;# Boucle pour lire les Scenarios
  for {set k 0} {$k < 2} {incr k} {
     set table($j,$k) [gets $f]              ;# Remplir par (site, heur)
   }
 }
close $f
# ==================================================================================
#              Declaration des couleurs selon les numeros                          #
# ==================================================================================
$ns_ color 0 Blue                ;# Le O est la Couleur Bleu (tous les messages)
# ==================================================================================
#                       Definition de la topologie                                 #
# ==================================================================================
$topo load_flatgrid $val(x) $val(y)
# ==================================================================================
#                      Creation du God ( voisinage )                               #
# ==================================================================================
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 $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace OFF
# ==================================================================================
#                        Creation des noeuds et des agents                         #
# ==================================================================================
for {set i 0} {$i < $nbr} {incr i} {
```

```
      set node_($i) [$ns_ node]
      $node_($i) random-motion 0
      $node_($i) color Black
      $god_ new_node $node_($i)
      set p($i) [new Agent/kemaodv]
      $p($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
}
# ============================================================================
#          La connexion des agents pour permettre la communication           #
# ============================================================================
for {set i 0} {$i < $nbr} {incr i} {
for {set j [expr $i+1]} {$j < $nbr} {incr j} {
$ns_ connect $p($i) $p($j)
}}
# ============================================================================
#               Coloration des noeuds a chaque couleur associe               #
# ============================================================================
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]

if {$a == -1} {
 if {$b > 0} {
   $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 Red"
}
}
# ============================================================================
#                       Appel des fichiers de mouvement                      #
# ============================================================================
if { $val(sc) == "" } {
  puts "*** NOTE: no connection pattern specified."
       set val(sc) "none"
} else {
  puts "Loading connection pattern..."
  source $val(sc)
}
# ============================================================================
#                Initialisation des variables globaux                        #
# ============================================================================
set nmsg1 0                                 ;# Initialisation de nmsg1
set nmsg 0                                  ;# Initialisation de nmsg
set tpatt 0                                 ;# Initialisation de tpatt
# ============================================================================
#              Fixation de la duree de la Section Critique                   #
# ============================================================================
```

```
set duree 1                                    ;# la duree de la Sc
# ============================================================================
#                  Initialisation des positions des noeuds                   #
# ============================================================================
for {set i 0} {$i < $nbr} {incr i} {
     $ns_ initial_node_pos $node_($i) 50       ;# se fait apres la definition
}                                              ;# du modele de mobilite

# ============================================================================
#              Affichage du Menu dans le fichier des resultats               #
# ============================================================================
$ns_ at 0.0 "Affichage"                      ;# Appel a la procedure Affichage
set ns_ [Simulator instance]
set nowe [$ns_ now]
# ============================================================================
#              Coloration des noeud selon leur etat actuelle                 #
# ============================================================================
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)"
 }
}
# ============================================================================
#                L'execution des scenarios de la simulation                  #
# ============================================================================
for {set j 0} {$j < $nbrequete} {incr j} { ;# lire les Scenarios de la table
 set site $table($j,0)                      ;# lire l'identificateur du site
 set heure $table($j,1)                     ;# lire l'heur de la demande
 $ns_ at $heure "$p($site) set sc 0"        ;# sc <- 0
 $ns_ at $heure "$p($site) set dem $heure" ;# affecter la demande a l'heur
 $ns_ at $heure "diffusion $p($site) $node_($site) $duree $heure"
 puts "$site"                               ;# Appel de diffusion pour
}                                           ;# afin d'executer les demandes
# ============================================================================
#              Finalisation de la simulation et debut du RUN                 #
# ============================================================================
$ns_ at [expr $val(stop) + 0.01] "finish";# Finir la Simulation apres(STOP)
puts "debut de la simulation ..."        ;# Ecrire"debut de la simulation.."
$ns_ run                                 ;# Debut du NAM
```

le Script TCL (Réseau AD HOC)

Annexe : L'outils de simulation B

Sommaire

Ce chapitre introductif gnagnagna.
Pas obligatoire !

B.1 Le simulateur NS-2

Pour résoudre les problèmes liés à un système donné, on fait appel à des algorithmes ou des protocoles, ces algorithmes doivent être testés et évalués pour les valider, et pour satisfaire leurs insuffisances, et cela à l'aide d'un outil de simulation.

B.1.1 Définition

NS-2 est un simulateur à événements discrets, écrit en C++. C'est le simulateur le plus célèbre dans le domaine de la simulation des réseaux. Il fournit un environnement permettant de réaliser des simulations entre des protocoles IP, TCP, UDP, routage et des protocoles multicast dans les réseaux filaires ainsi que dans les réseaux sans fil avec un support de mobilité des nœuds.[Dio07]

B.1.2 Les étapes de simulation

La simulation avec NS-2 passe en général par trois phases :

- **Définition de la topologie du réseau** : on crée les nœuds, avec les caractéristiques de chacun, ainsi que les liens entre les nœuds. On peut définir sur chaque lien, le délai de transmission, la bande passante, le fait qu'il soit simplexe ou duplexe et le type de file d'attente se trouvant à son extrémité.

L'exemple suivant montre comment définir une topologie ***Anneau*** :

```
# ======================================================================
#                        Creation d'une tolpologie                     #
# ======================================================================
for {set i 0} {$i < 7} {incr i} {     ;# Boucle de creation de 7 noeuds
   set n($i) [$ns node]               ;# La creation d'un noeud
 }
for {set i 0} {$i < 7} {incr i} {     ;# Boucle pour creer les liens
                                      ;# Creation d'un lien bidirectionel
   $ns duplex-link $n($i) $n([expr ($i+1)%7]) 1Mb 10ms RED
 }                                    ;# Et les caracteristiques associees
```

Création d'une topologie anneau

Afin d'obtenir la topologie B.1 suivante :

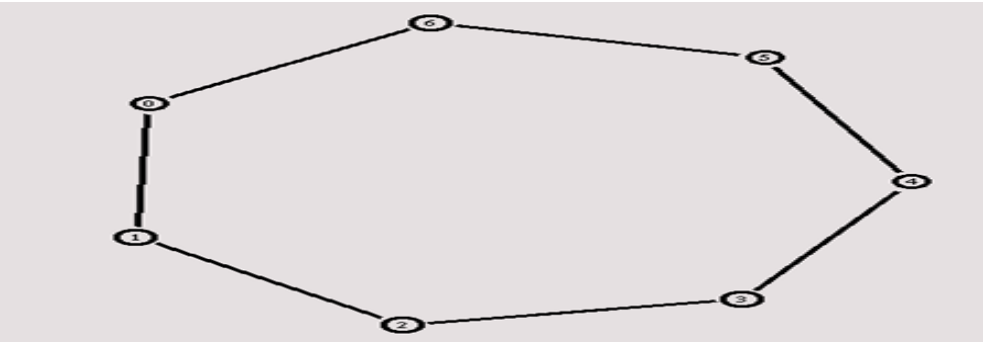

Figure B.1 – *Exemple de Création d'une topologie.*

- **Spécification du scénario de la simulation** : l'utilisateur spécifie les différents agents de la communication qui vont agir pendant la simulation. Il spécifie aussi la succession des différentes opérations (à l'instant t_1, envoi des données, à l'instant t_2 arrêt d'émission).

- **Exploitation des résultats** : cette dernière phase consiste en un recueil des statistiques de la simulation. Ces dernières peuvent être exploitées directement par NS-2 en faisant appel aux outils qui l'accompagnent, ou bien elles seront archivées pour une utilisation ultérieure par d'autres outils de traitements statistiques.

B.1.3 Comment réaliser une simulation ?

La simulation nécessite des données qui caractérisent l'environnement, tels que la surface du réseau, la topologie utilisée, le protocole à simuler ... etc. Généralement, ces données ne sont pas définies en NS. Pour cela, l'utilisateur doit définir les informations (données) en utilisant le langage C++.

Afin de réaliser un algorithme, nous devons créer deux fichiers source écrits en langage C++ (***.h, *.cc***).

Le premier (***.h***) est un fichier d'en-tête, qui contient la structure des messages échangés entre les sites, le deuxième (***.cc***) contient les fonctions nécessaires de l'algorithme (envoi et réception des messages, ...).

La compilation de ces fichiers nous permet d'avoir un fichier de type objet (***.o***), ce dernier doit être intégré dans le simulateur NS-2, en lui ajoutant à la variable ***OBJ_CC*** du fichier (***makefile***), enfin, on recompile le noyau de NS par la commande ***make*** écrite dans un terminal.

On peut résumer les étapes de réalisation d'un algorithme dans la figure B.2.

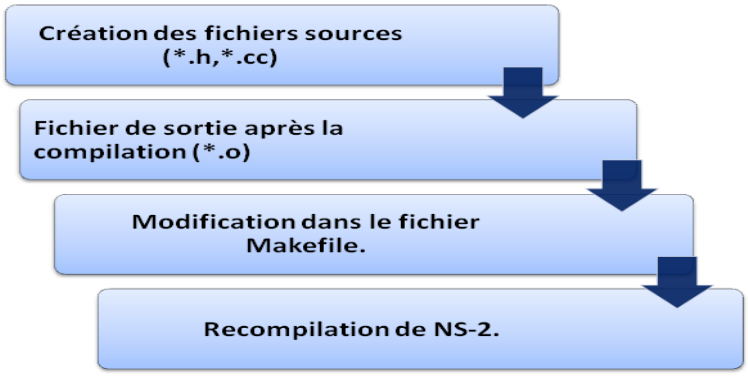

Figure B.2 – *Les étapes de simulation.*

Pour tester et utiliser ces programmes, nous avons créer les fichiers (***.tcl***) qui contient la topologie du réseau, ensuite on appel les fonctions du code C++, et enfin la visualisation de simulation.

La sortie standard est un fichier d'animation (***.nam***) pour visualiser la simulation, et un fichier de trace (***.tr***) qui contient toutes les informations obtenues après la simulation, ces informations sont très générales et nécessitent des filtres pour bien les exploiter. C'est la raison pour laquelle, nous avons utilisé un fichier texte (***.txt***) qui ne contient que les informations importantes pour notre simulation.

Nous pouvons résumer les étapes de réalisation d'une simulation par la figure B.3 :

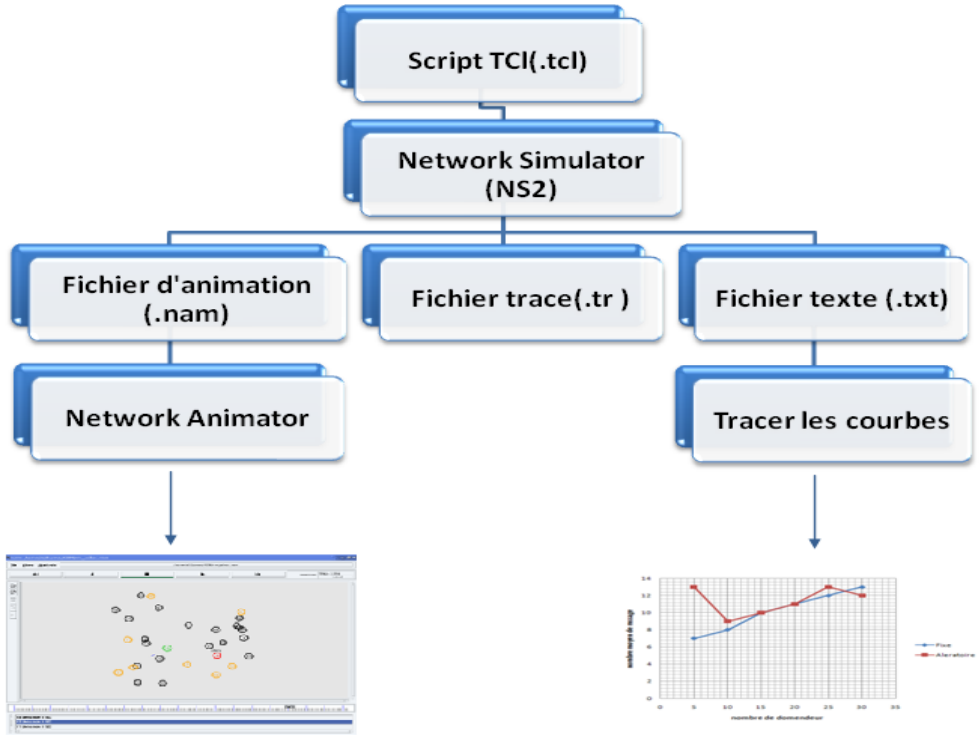

Figure B.3 – *Réalisation d'une simulation.[LK09]*

B.1.4 Les paramètres de simulation

Pour réaliser notre simulation, nous avons défini plusieurs paramètres, certains paramètres sont fixes et ne changent pas durant toute la simulation, d'autre sont variables et leur changement permet d'obtenir à chaque fois un nouveau scénario de simulation.

La variation de ces paramètres nous permet d'identifier ceux ayant une influence sur la performance de notre algorithme.

B.1.4.1 Les paramètres fixes

- **Surface de réseau** : C'est l'intervalle maximal utilisé par les nœuds pendant leurs mouvements, nous avons choisi une surface de 500m*500m.

- **La durée de la SC** : C'est la durée d'exécution d'une SC par un site, nous avons fixé cette valeur à 1 seconde.

- **Protocole de routage** : le protocole de routage permet de définir les chemins entre les nœuds pour échanger des messages.nous choisissons le Protocole réactif ***AODV***.

- **Modèle de propagation** : Dans NS-2, trois modèles de propagation sont implémentés : ***Free Space***, ***Shadowing***, ***Two-ray-ground***. Dans notre simulation, le modèle ***Two-ray-ground*** est choisi comme modèle de propagation, car ce modèle est devenu

un standard dans la recherche sur les réseaux mobiles.

– **Modèle de mobilité** : NS-2 offre quelques modèles de mobilité basés sur la génération aléatoire des positions des nœuds. Parmi ces modèles, on peut citer ***Random-WayPoint***, ***Random Walk***, ***Random Direction model***, nous avons choisi le modèle ***Random-WayPoint***, car dans ce modèle, les nœuds sont distribués uniformément dans l'espace de simulation, leurs positions initiales sont aléatoire ainsi que leurs déplacements.

B.1.5 Évaluation de performances

Le but des différents scénarios était d'identifier les paramètres ayant une influence sur la performance de l'algorithme, cette performance est exprimée par deux valeurs importantes :

– **Le nombre de messages moyen (NMM)** : Chaque entrée en section critique nécessite un échange d'un ensemble de messages, le nombre moyen de messages échangés par entrée en section critique est calculé par la formule suivante :

$$NMM = \frac{\text{Nombre de messages total}}{\text{Nombre d'entrée en Section Critique}}$$

Cette valeur est considérée comme un facteur important de performance des algorithmes de l'exclusion mutuelle.

– **Le temps d'attente moyen (TAM)** : Un site qui demande une section critique doit attendre un intervalle de temps appelé le temps d'attente, le temps d'attente moyen est calculé par la formule suivante :

$$TAM = \frac{\sum_i \text{Temps } d'attente_i}{\text{Nombre d'entrée en Section Critique}}$$

Le temps d'attente moyen est également l'un des facteurs importants pour l'évaluation de la performance des algorithmes.

B.1.6 Les étapes d'un scénario

Pour réaliser un scénario donné, nous devons introduire les différentes informations nécessaires pour la simulation dans l'ordre suivant :

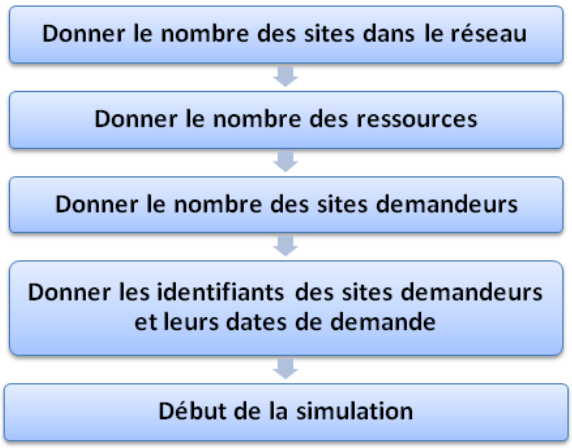

Figure B.4 – *Les étapes de réalisation d'un scénario.*

Acronymes

AODV	Ad hoc On-Demand Distance Vector
DSR	Dynamic Source Routing
EM	Exclusion Mutuelle
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
K-EM	K-Exclusion Mutuelle
MAC	Medium Access Control
MANET	Mobile Ad-hoc Networks
MSG	MeSsaGe
NbJeton	Nombre de Jeton
NAM	Network AniMator
NMM	Nombre de Messages Moyen
NS-2	Network Simulator 2
OLSR	Optimized Link State Routing protocol
RDM	Random Direction Model
RED	Random Early Detection
RWM	Random Waypoint Model
SB	Station de Base
SC	Section Critique
TAM	Temps d'Attente Moyen
TCL	Tools Command Language
TCP	Transport Control Protocol
TPATT	TemPs d'ATTente
UM	Unités Mobiles
ZRP	Zone Routing Protocol