

RAPPORT DE PROJET DE FIN D'ETUDES

Filière

Ingénieur en Télécommunications

Option

Ingénierie des réseaux

Routage avec QoS temps réel dans les réseaux de capteurs

Elaboré par :

DHIB Eya

Encadré par :

Mr TABBENE Nabil

Mlle AFIF Meriem

Année universitaire : 2006 / 2007

*À ma mère
À mon père
À mon oncle Omed
À ma tante Najet
À mes frères et mes sœurs
À toute ma famille
À tous ceux qui me sont chères
Je dédie ce travail.*

Remerciement

Au terme de ce travail, je voudrais adresser mes sincères remerciements à Monsieur Nabil TABBANE et Madame Meriem AFIF pour leurs encadrements et leurs encouragements tout au long de ce projet.

Je tiens, également, à remercier les membres du jury, qui ont accepté d'évaluer ce modeste travail.

Je souhaite témoigner ma gratitude à tous ceux qui ont contribué, de près ou de loin à l'élaboration de ce travail.

 Eya DHIB

Table des matières

Introduction générale.....	7
Chapitre 1 Généralités sur les réseaux de capteurs.....	10
1.1. Introduction	10
1.3. Composant d'un nœud capteur.....	10
1.4. Architecture des réseaux de capteurs	11
1.5. Caractéristiques des réseaux de capteurs.....	12
1.6. Contraintes et facteurs conceptuelles des réseaux de capteurs.....	13
1.7. Les modèles de transmission des données dans les réseaux de capteurs	14
1.8. Applications des réseaux de capteurs	15
1.9. Conclusion.....	16
Chapitre 2 Le routage dans les réseaux de capteurs.....	17
2.1. Introduction	17
2.2. Contraintes de routage dans les réseaux de capteurs sans fil	17
2.3. Classification des protocoles de routage dans les réseaux de capteurs	17
2.4. Les protocoles de routage pour les réseaux de capteurs.....	22
2.5. Conclusion.....	25
Chapitre 3 Protocoles de routage avec QoS : SPEED, MMSPEED et E2MSPEED.....	26
3.1. Introduction	26
3.2. Le protocole SPEED.....	26
3.2.1. Paquet de localisation	27
3.2.2. Estimation de délai	28
3.2.3. <i>Stateless Non-deterministic Geographic Forwarding</i> (SNGF).....	28
3.2.4. Procédure de reroutage pour délai	30
3.2.5. Le protocole de la couche MAC: DCF	36
3.3. MMSPEED : Multi Path Multi SPEED.....	36
3.3.1. Garantir une QoS temporelle	37
3.3.2. Garantir une QoS en fonction du PDR	39
3.3.3. Le protocole de la couche MAC : EDCF.....	42
3.4. Le protocole E3MSPEED : Enhanced Mac MMSPEED	43
3.4.1. Contribution.....	43
3.4.2. Le protocole de la couche MAC : EDCA	43
3.5. Conclusion	48
Chapitre 4 Évaluation des performances des protocoles : SPEED, MMSPEED et E3MSPEED	49
4.1. Paramétrage de la simulation.....	49
4.2. Méthodologie de la simulation	49
4.3. SPEED et MMSPEED.....	50
4.4. MMSPEED et E3MSPEED.....	59
4.5. Conclusion.....	61
Conclusion générale	64

Bibliographie.....	66
Annexe	68
1. Présentation du simulateur JSIM (<i>Java SIMulator</i>)	68
2. Installation du simulateur JSIM sous la plateforme WINDOWS.....	68
2.1. Préparation d'environnement.....	68
2.2. Installation de JSIM.....	69
3. Implémentation du protocole MMSPEED.....	71
4. Implémentation du protocole SPEED.....	71
5. Technique de simulation.....	71
5.1. Pré-simulation.....	71
5.2. Simulation.....	72
5.3. Post-simulation	73
5.4. Exploitation.....	73

Liste des figures

Figure 1.1	Les composants d'un nœud capteur.....	11
Figure 1.2	Architecture d'un réseau de capteurs.....	12
Figure 2.1	Classification des protocoles de routage pour les réseaux de capteurs	18
Figure 2.2	Routage plat	19
Figure 2.3	Routage hiérarchique	20
Figure 2.4	Routage basé sur la localisation.....	21
Figure 3.1	Architecture fonctionnelle du protocole SPEED.....	27
Figure 3.2	Paquet de localisation	27
Figure 3.3	Calcul de la vitesse de livraison.....	28
Figure 3.4	Choix de nœud de prochain saut.....	29
Figure 3.5	Paquet « Delay Back pressure »	30
Figure 3.6	Procédure de re-routage pour délai « Delay back pressure rerouting » (cas 1)	32
Figure 3.7	Procédure de re-routage pour délai « Delay back pressure rerouting » (cas 2)	34
Figure 3.8	Mécanisme de <i>Void Avoidance</i>	35
Figure 3.9	Modélisation de la couche réseau du protocole MMSPEED.....	37
Figure 3.10	Différentiation de la qualité de trafic.....	39
Figure 3.11	Sélection des prochains sauts selon la contrainte qualité de trafic	41
Figure 3.12	Différentiation de services au niveau de la couche MAC EDCA.....	46
Figure 3.13	Accès EDCA.....	47
Figure 4.1	Délai de bout en bout : SPEED et MMSPEED	50
Figure 4.2	Taux de livraison des paquets avec contrainte qualité de trafic : SPEED (1000 m/s) et MMSPEED.....	52
Figure 4.3	Taux de livraison des paquets avec contrainte qualité de trafic : SPEED (250m/s) et MMSPEED.....	53
Figure 4.4	Taux de livraison des paquets avec contraintes délai de bout en bout et qualité de trafic pour le protocole MMSPEED.....	54
Figure 4.5	Taux de livraison des paquets avec contraintes délai de bout en bout et qualité de trafic pour le protocole SPEED	55
Figure 4.6	Paquet de contrôle du trafic prioritaire : SPEED et MMSPEED.....	56
Figure 4.7	Paquet de contrôle du trafic moins prioritaire	57
Figure 4.8	Délai de bout en bout : MMSPEED et E3MSPEED	59
Figure 4.9	Taux de livraison des paquets avec contraintes délai de bout en bout et qualité de trafic pour le protocole E3MSPEED	60
Figure 4.10	Paquet de contrôle du trafic prioritaire : MMSPEED et E3MSPEED.....	61
Figure 4.11	Paquet de contrôle du trafic prioritaire : MMSPEED et E3MSPEED.....	61

Liste des tableaux

Tableau 3.1 Le passage de priorité d'utilisateur (802.11d) aux catégories d'accès (802.11e)..	44
Tableau 3.2 Paramètres d'accès pour chaque AC en mode EDCA	45
Tableau 3.3 Valeur d'accès d'après le <i>Draft IEEE 9.0</i>	46
Tableau 4.1 Paramètres du contexte de la simulation	49
Tableau 4.2 Paramètres du protocole EDCA	59

Acronymes

AC	Access Category
ACK	ACKnowledgment
AIFS	Arbitration Inter-Frame Spacing
APTEEN	Adaptatif Threshold sensitive Energy Efficient sensor Network protocol
BE	Best effort
BI	Backoff Interval
BK	Background
CADR	Constrained Anisotropic Diffusion Routing
CW	Contention Window
DCF	Distributed Coordination Function
DIFS	DCF IFS
dPDR	desired Packet Delivery Ratio
E3MSPEED	Enhanced MAC MMSPEED
EDCA	Enhanced Distribution Channel Access
EDCF	Enhanced Distributed Coordination Function
EWMA	Exponential Weighted Moving Average
GAF	Geographic adaptive fidelity
GEAR	Geographic and Energy-Aware Routing
ID	Identification
IEEE	Institute of Electrical and Electronics Engineers
IFS	Inter Frame Spaceng
JSIM	Java SIMulator
MAC	Medium Access Control
MECN	Minimum Energy Communication Network
MMSPEED	Multi path Multi SPEED
OTCL	Object Tool Command Language
PDR	Packet Delivery Ratio
PEGASIS	Power Efficient GATHERing in Sensor Information Systems
PF	Persistence Factor
QoS	Quality of Service
RP	Reaching Probability

RP	Reaching Probability
SAR	Sequential Assignment Routing
SIFS	Short IFS
SMECN	Small Minimum Energy Communication Network
SNGF	Stateless Non-deterministic Geographic Forwarding)
SPEED	SPEED
SPIN	Sensor Protocols for Information via Negotiation
TC	Traffic Category
TEEN	Threshold sensitive Energy Efficient sensor Network protocol
TRP	Total Reaching probability
TXOP	Transmission Opportunity
VI	Video
VO	Voice

Introduction générale

Depuis quelques décennies, le besoin d'observer, de surveiller à distance et de récupérer les données d'un environnement complexe et distribué s'accroît rapidement surtout avec les récentes avancées des domaines de réseaux et des applications sans fils. Dans ce constat et pour répondre à ces attentes, une nouvelle branche s'est créée donnant ainsi l'apparition des réseaux de capteurs. Ces réseaux forment un type particulier des réseaux Ad Hoc, dans lesquels les nœuds sont des capteurs. Dans ce type des réseaux, les capteurs échangent l'information sur l'environnement afin d'établir une vue globale de la région surveillée. Cette information sera, ensuite, délivrée à l'utilisateur externe à travers le nœud passerelle « *Sink* ».

Les réseaux de capteurs se distinguent par leur densité des nœuds importante, leur autonomie énergétique limitée, leur topologie mobile, etc. Ces contraintes ont posé plusieurs défis pour la conception et la gestion des réseaux de capteurs. L'un de ces défis était la mise en revue et le développement des nouveaux protocoles de routage adéquats à la nature particulière des réseaux de capteurs. Notre projet s'intéresse plus particulièrement à l'étude des protocoles de routage basé sur la QoS temps réel.

Dans le premier chapitre, nous présentons une description générale des réseaux de capteurs, leurs caractéristiques, leur architecture ainsi que leurs domaines d'applications. Le deuxième chapitre synthétise tous les protocoles de routage dédiés à ce genre de réseaux. Le troisième chapitre se consacre pour l'étude des protocoles de routage basés sur la QoS à savoir le protocole SPEED et MMSPEED (*Multi path Multi SPEED*). Nous avons essayé d'améliorer l'offre de la QoS en proposant notre nouveau protocole E3MSPEED (*Enhanced MAC MMSPEED*). Dans le dernier chapitre, nous exposons les résultats de nos simulations relatives aux protocoles de routage mentionnés et nous comparons les performances de chacun en termes des deux contraintes de délai de bout en bout et le taux des paquets délivrés avec succès.

Chapitre 1

Généralités sur les réseaux de capteurs

1.1. Introduction

Grâce aux avancées technologiques de plusieurs secteurs des technologies de l'information, il devient aujourd'hui envisageable de produire en masse des systèmes d'une taille extrêmement réduite (de quelques centimètres) et embarquant des unités de calcul et de communication sans fil pour un coût réduit. Ayant ces caractéristiques, les nœuds capteurs sont capables de générer et d'échanger des données d'une manière autonome et complètement transparente pour les utilisateurs.

Les réseaux de capteurs représentent actuellement un nouveau domaine, en plein développement, émergeant des innovations des technologies de communication.

L'objectif de ce chapitre est de faire une description synthétique des réseaux de capteurs, leurs architectures, leurs caractéristiques et contraintes ainsi que leurs domaines d'applications variés.

1.2. Définition du réseau de capteurs

Les réseaux de capteurs sont souvent composés d'un nombre très important de nœuds, appelés capteurs. Ces nœuds sont des entités capables d'opérer en toute autonomie afin de collecter, traiter et envoyer les données relatives à leur environnement. Déployés de façon à couvrir un territoire donné, les capteurs communiquent par radio afin de concentrer l'information sur une station collectrice située au cœur ou en bordure du territoire, appelée nœud « *Sink* » [1,2,3]. Ces entités doivent être équipées d'une batterie qui est une source d'énergie non durable, ce qui présente la contrainte la plus gênante pour la survie d'un tel réseau.

Les réseaux de capteur ne définissent pas une architecture préexistante puisque les capteurs peuvent se trouver fixés à un endroit précis ou mobile, formant ainsi un réseau sans infrastructure prédéfinie. Tel réseau ne se limite pas à un domaine particulier mais il peut s'adresser à une diversité de secteurs [1,2,3,4] comme la biologie, la chimie, l'environnement, ainsi que la surveillance sismique et même la télésurveillance personnelle...

1.3. Composant d'un nœud capteur

Un nœud capteur est constitué des principales composantes suivantes :

- **Unité de captage (*Sensing Unit*)** : elle permet de capturer le phénomène observé et le convertir depuis un signal analogique en un numérique. Il sera, ensuite, fourni à l'unité de calcul.
- **Unité de traitement (*Processing Unit*)** : elle est chargée d'exécuter les protocoles de communication qui permettent aux nœuds capteurs de collaborer avec les autres nœuds pour accomplir la requête en question.
- **unité de transmission (*Transceiver Unit*)**: elle est chargée d'effectuer toutes les émissions et les réceptions de données.
- **Unité de contrôle d'énergie (*Power Unit*)** : elle effectue des opérations de contrôle de l'énergie restante et de mesure de la durée de vie du nœud capteur.
- **système de localisation (*Location Finding System*)** : il fournit des informations sur la localisation requise par les techniques de routage.
- **mobilisateur (*Mobilizer*)** : il est appelé si le nœud capteur doit être déplacé pour accomplir la requête à traiter.

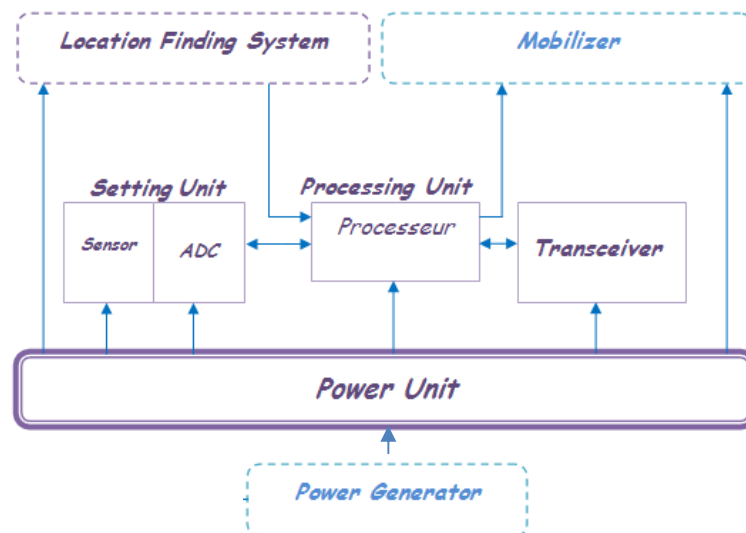


Figure 1.1 Les composants d'un nœud capteur

1.4. Architecture des réseaux de capteurs

Les nœuds de capteurs sont habituellement dispersés dans un champ de capteur « *sensor field* ». Ils forment ainsi un réseau sans fil à multi-saut comme (c.f figure I.1). Chaque nœud a pour mission la collecte des données et leurs routages vers le nœud de contrôle « *Sink* ». À son tour, le nœud de contrôle transmet les informations reçues à partir d'Internet ou par satellite à l'utilisateur final « *task manager* ».

Il existe deux types d'architectures pour les réseaux de capteurs : les réseaux de capteurs plats et les réseaux de capteurs hiérarchiques. [2,3]

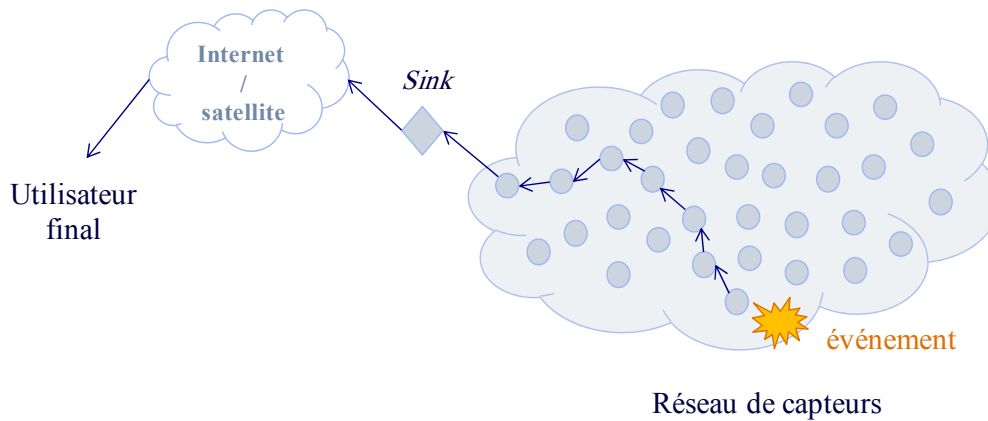


Figure 1.2 Architecture d'un réseau de capteurs

1.4.1. Les réseaux de capteurs sans fil plats :

Un réseau de capteurs sans fil plat est un réseau homogène, où tous les nœuds sont identiques en termes de batterie et des fonctions, excepté le « *Sink* » [3]. Ce dernier joue le rôle d'une passerelle et est responsable de la transmission de l'information collectée à l'utilisateur final (voir figure 2.2).

1.4.2. Les réseaux de capteurs hiérarchiques :

Une architecture hiérarchique a été proposée pour réduire le coût et la complexité de la plus part des nœuds capteurs. Elle consiste à introduire un ensemble de nœuds plus coûteux et plus puissants, en créant une infrastructure qui décharge la majorité des nœuds simples à faible coût de plusieurs fonctions du réseau. L'architecture hiérarchique est composée de plusieurs couches : une couche de capteur, une couche de transmission et une couche de point d'accès (voir figure 2.3).

1.5. Caractéristiques des réseaux de capteurs

Les principaux caractéristiques des réseaux de capteurs se résument dans ce qui suit :

1.5.1. Densité « importante » des nœuds

Les réseaux de capteurs se composent généralement d'un nombre très important des nœuds pour garantir une couverture totale de la zone surveillée. Ceci engendre un niveau de surveillance élevé et assure une transmission plus fiable des données sur l'état du champ de capteur.

1.5.2. Topologie dynamique

La topologie des réseaux de capteurs instable est le résultat des trois facteurs essentiels suivants:

- **La mobilité des nœuds** : les nœuds capteurs peuvent être attachés à des objets mobiles qui se déplacent librement et arbitrairement, introduisant ainsi une topologie instable du réseau.
- **La défaillance des nœuds** : du fait de l'autonomie énergétique limitée des nœuds, la topologie du réseau n'est pas fixée (les nœuds « morts » sont, d'un point de vue logique, simplement supprimés).
- **L'ajout de nouveaux nœuds** : de nouveaux nœuds peuvent facilement être rajoutés. Il suffit de placer un nouveau capteur qui soit dans la portée de communication d'au moins un autre nœud capteur du réseau déjà existant.

1.5.3. Auto organisation

L'auto organisation s'avère très nécessaire pour ce type de réseau afin de garantir sa maintenance. Vu les différentes raisons résultant une topologie instable du réseau de capteur, ce dernier devra être capable de s'auto organiser pour continuer ses applications.

1.5.4. La tolérance de fautes

Le réseau doit être capable de maintenir ses fonctionnalités sans interruptions en cas de défaillance d'un ou plusieurs de ses capteurs. Cette défaillance peut être causée par une perte d'énergie, ou par dommage physique ou interférence de l'environnement. Le degré de tolérance dépend du degré de criticité de l'application et des données échangées [4].

1.5.5. Scalabilité

Les réseaux de capteurs peuvent contenir des centaines voire des milliers de nœuds capteurs [4]. Un nombre aussi important engendre beaucoup de transmissions inter nodales et nécessite que le nœud « *Sink* » soit équipé d'une mémoire importante pour stocker les formations reçues.

1.6. Contraintes et facteurs conceptuelles des réseaux de capteurs

L'architecture des réseaux de capteurs peut être influencée par certains facteurs pouvant agir sur son bon fonctionnement :

1.6.1. Pannes fréquentes

Pour que les paquets de données atteignent la destination finale « *Sink* », ils vont passer par un grand nombre de nœuds. À cause de sa topologie dynamique et la probabilité de rencontrer un nœud capteur en panne ou mort, le réseau de capteurs ne peut plus garantir des taux de

livraison importants. De plus, la communication radio peut être bruitée, ce qui peut endommager l'information transmise

1.6.2. L'environnement

Les nœuds capteurs doivent être conçus d'une manière à résister aux différentes et sévères conditions de l'environnement : forte chaleur, pluie,...

1.6.3. La topologie dynamique

La mobilité des nœuds, la possibilité d'étendre le réseau par ajout de nouveaux nœuds et la suppression de nœuds défaillants changent fréquemment la topologie des réseaux. Ceci nécessite une maintenance permanente pour assurer le bon fonctionnement des réseaux de capteurs.

1.6.4. La consommation d'énergie

Un capteur, de part sa taille, est limitée en énergie. Dans la plupart des cas, le remplacement de la batterie est impossible, ce qui veut dire que la durée de vie d'un capteur dépend grandement de la durée de vie de sa batterie. Les recherches actuelles se concentrent principalement sur les moyens d'optimiser la consommation d'énergie par les nœuds capteurs.

1.7. Les modèles de transmission des données dans les réseaux de capteurs

La transmission de données dans les réseaux de capteurs peut se faire suivant plusieurs modèles dont on distingue trois essentielles :

- Modèle *event driven*
- Modèle *query driven*
- Modèle *continuous*

1.7.1. Le modèle *driven event*

La génération et la transmission des paquets DATA est commandé par la réalisation d'un événement. La plupart des applications *driven event* [5] sont des applications intolérantes aux délais (temps réel), critiques, interactives et de non bout en bout. En fait, au lieu d'avoir un nœud émetteur et un autre récepteur de l'information, on trouve un nœud récepteur (le nœud de contrôle « *sink* ») et un groupe de nœuds capteurs, se trouvant proche de l'événement, qui sont tous des émetteur de la même information.

La réussite de ces applications, pour ce modèle, repose essentiellement sur la détection de l'événement et la rapidité des prises des réactions nécessaires pour assurer l'aspect temps réel des applications.

L'inconvénient majeur de ce modèle est la redondance des données. En fait, les nœuds excités par le même événement envoient la même information au nœud de contrôle « *sink* ». Pour cela, un protocole de routage basé sur la négociation des données est recommandé.

1.7.2. Le modèle *query driven*

Le modèle *query driven* est semblable au modèle *driven event* sauf que la collecte des informations sur l'état de l'environnement est initiée par des interrogations envoyées par le nœud de contrôle « *sink* », alors que, pour le modèle précédant, elle est déclenchée suite à un événement détecté. La plupart des applications *query driven* sont des applications interactives, critiques, de bout en bout et leur tolérance aux délais dépend de l'urgence de l'interrogation [5].

Notons que le modèle *query driven* peut être utilisé pour contrôler et reconfigurer les nœuds. Par exemple, le « *sink* » peut envoyer des commandes au lieu des interrogations pour modifier le programme d'un nœud capteur, modifier son taux de trafic ou son rôle. Seul le nœud capteur jouant le rôle de « *sink* » est autorisé d'émettre des demandes d'interrogations ou des commandes et ce pour assurer l'ordre et l'hierarchie de réseau de capteur.

1.7.3. Le modèle *continuous*

Dans le modèle continu, les nœuds capteurs envoient les informations d'une manière continue au nœud « *sink* » suivant un volume de trafic prédéterminé.

1.8. Applications des réseaux de capteurs

Les réseaux de capteurs promettent beaucoup de domaines de nouvelles applications à savoir :

1.8.1. Applications militaires [4]: Les nœuds capteur devraient fournir les services suivants:

- surveillance du champ de bataille
- Reconnaissance des forces d'opposition
- repérage des cibles
- évaluation des dommages de la bataille
- détection et reconnaissance d'attaque nucléaire, biologique et chimique.

1.8.2. Applications de santé[4]: Certaines applications de santé des réseaux de capteurs sont:

- fourniture d'interfaces pour les handicapés;
- repérage et surveillance des médecins et des patients dans les hôpitaux;

- télésurveillance des données physiologiques humaines.

1.8.3. Applications environnementales [4]: Ces applications incluent :

- le repérage des mouvements des oiseaux des petits animaux et des insectes;
- la surveillance des conditions environnementales qui affectent les récoltes et le bétail;
- l'exploration planétaire;
- alertes des catastrophes (incendie, séisme...)
- la détection d'inondation;
- l'étude de pollution.

1.9. Conclusion

La flexibilité, la tolérance de fautes, le prix réduit et les caractéristiques rapides de déploiement des réseaux de capteurs offrent des possibilités infinies de développement dans tous les domaines d'application. Ceci nous permet de penser que les réseaux de capteurs feront bientôt partie intégrante de nos vies et satisferont sûrement les plus grands projets.

Cependant, la réalisation des réseaux de capteurs doit satisfaire quelques contraintes parmi lesquelles on peut citer : la consommation d'énergie, le changement de topologie, la densité importante des réseaux... Ces contraintes exigent que de nouveaux protocoles de routage soient mises au point.

Nous nous intéressons, dans la suite de ce projet, à la présentation des protocoles de routages déployés pour les réseaux de capteurs.

Chapitre 2

Le routage dans les réseaux de capteurs

2.1. Introduction

Les caractéristiques des réseaux des capteurs, comme la densité importante des nœuds, leurs autonomies énergétiques limitées et la topologie qu'ils forment, exigent des protocoles de routage spécifiques, différents de ceux déployés dans les réseaux usuels. De ce fait, le développement de nouveaux protocoles de routage s'avère indispensable. Ces protocoles doivent tenir compte de l'aspect fonctionnel de ces réseaux tout en optimisant les calculs nécessaires pour choisir la route la plus optimale.

Dans ce chapitre, nous décrivons les différentes familles de protocoles de routage déployés par les réseaux de capteurs.

2.2. Contraintes de routage dans les réseaux de capteurs sans fil

Le routage dans les réseaux de capteurs diffère de celui des réseaux Ad Hoc dans les points suivants :

- il n'est pas possible d'établir un système d'adressage global pour le grand nombre de nœuds.
- les applications des réseaux de capteurs exigent l'écoulement de données mesurées depuis des sources multiples vers la destination finale « *Sink* ».
- les différents capteurs peuvent générer produire les mêmes données à proximité d'un phénomène (problème de la redondance des données).
- les nœuds capteurs exigent ainsi une gestion soigneuse des ressources.

En raison de ces différences, de nouveaux protocoles de routage ont été proposés dans les réseaux de capteurs.

2.3. Classification des protocoles de routage dans les réseaux de capteurs

Les protocoles de routage dans les réseaux peuvent être classés selon deux concepts [6]:

- la structure de réseau,
- le type de protocole.

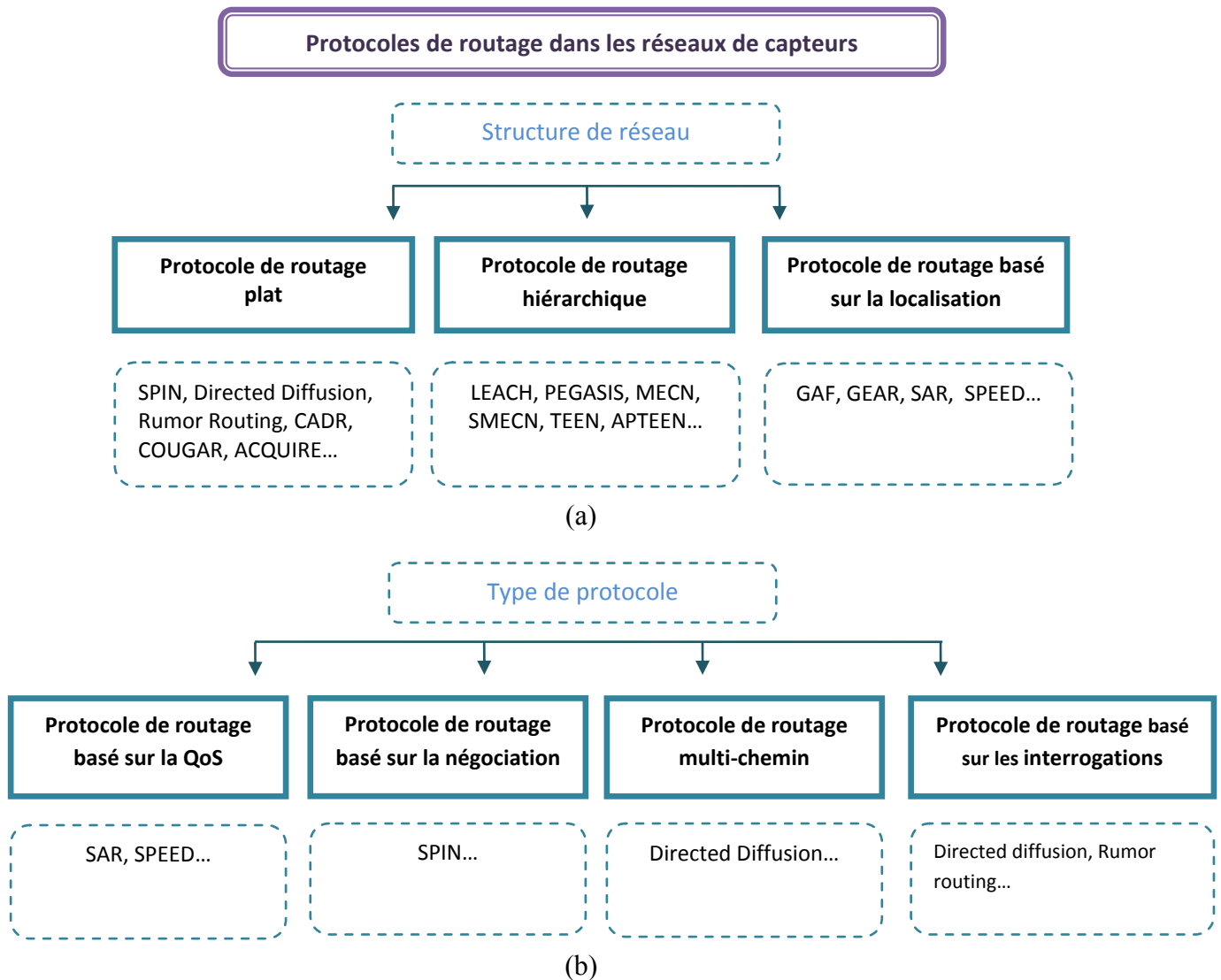


Figure 2.1 Classification des protocoles de routage pour les réseaux de capteurs

2.3.1. Protocole de routage multi-chemin: il se base sur l'adoption de plus qu'un chemin menant vers la destination, et ce, pour avoir des chemins de secours si jamais le chemin principal serait rompu.

2.3.2. Protocole de routage basé sur la négociation des données: En détectant le même phénomène, les nœuds capteurs inondent le réseau par les mêmes paquets de données. Ce problème de redondance peut être résolu en employant des protocoles de routage basés sur la négociation. En effet, avant de transmettre, les nœuds capteurs négocient entre eux leurs données en échangeant des paquets de signalisation spéciales, appelés *meta-DATA*. Ces paquets permettent de vérifier si les nœuds voisins disposent des mêmes données à transmettre [6]. Cette procédure garantit que seules les informations utiles seront transmises et élimine la redondance des données.

2.3.3. Protocole de routage basé sur les interrogations: la collecte des informations sur l'état de l'environnement est initiée par des interrogations envoyées par le nœud « *Sink* ». (voir paragraphe 1.7.2).

2.3.4. Protocole de routage basé sur la QoS: Ce type de protocoles tend à satisfaire certaines métriques, pendant la transmission des données vers la destination finale. Parmi ces métriques, nous citons : le délai de bout en bout, la gigue, PDR (*Paquet Delivery Ratio*), énergie consommée...

Nous nous intéressons, dans la suite, à traiter des protocoles adoptant ce type de routage et dégager leurs performances en la garantie de la QoS temps réel.

On peut adopter une autre classification des protocoles de routage pour les réseaux de capteurs en se basant sur la structure de réseau déployée. Ainsi, on distingue les protocoles de Routage hiérarchique, plat et par localisation.

2.3.5. Les protocoles de routage plat (flat based-routing) :

Ces protocoles considèrent que tous les nœuds sont identiques, c'est à dire ont les mêmes fonctions à exécuter sauf le nœud de contrôle « *sink* » qui est chargé de collecter toutes les informations issues des différents nœuds capteurs pour les transmettre vers l'utilisateur final. La décision d'un nœud de router des paquets vers un autre dépendra de sa position et pourra être remise en cause au cours du temps.

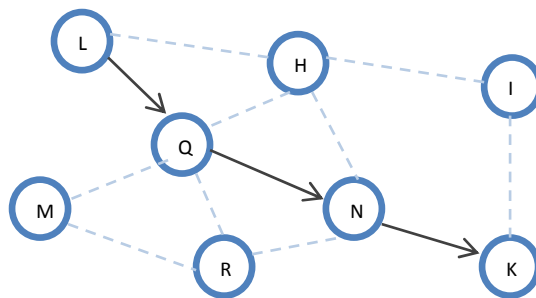


Figure 2.2 Routage plat

2.3.6. Les protocoles de routage hiérarchique :

Ces protocoles fonctionnent en confiant des rôles différents aux nœuds du réseau. Certains nœuds sont sélectionnés pour exécuter des fonctions particulières. Un nœud peut être, par exemple, une passerelle pour un ensemble de nœuds. Dans ce cas, le routage devient plus simple, puisqu'il s'agit de passer par les passerelles pour atteindre le nœud destination qui lui est directement attaché.

Un exemple est donné par la figure 2.3 : Pour que les paquets générés par le nœud F atteignent le nœud L, ils doivent passer par les passerelles P, S et R.

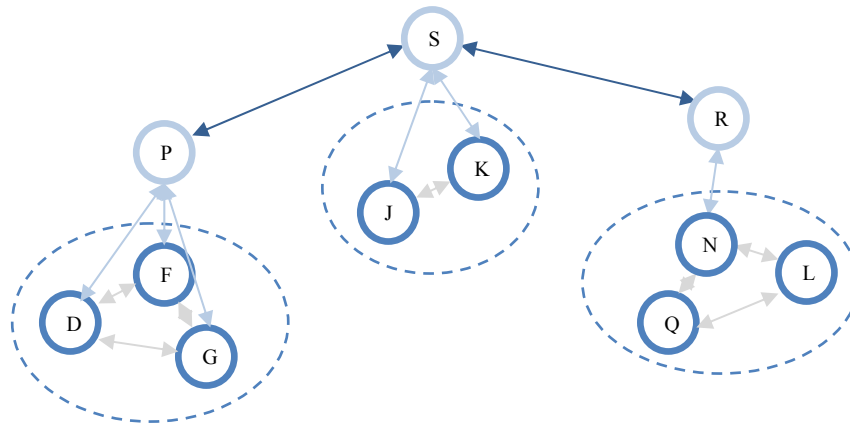


Figure 2.3 Routage hiérarchique

Le principe des protocoles de routage hiérarchique est basé essentiellement sur les nœuds passerelles. En fait, les nœuds ordinaires savent que si le destinataire n'est pas dans leur voisinage direct, il suffit d'envoyer la requête à la passerelle qui la prendra en charge. À son tour, elle transmettra cette requête vers le nœud ciblé.

Ce type de routage présente de nombreux avantages pour les réseaux dont leurs nœuds sont sédentaires et disposent de suffisamment d'énergie [6].

2.3.7. Les protocoles de routage avec localisation géographique :

Un routage est dit géographique lorsque les décisions de routage sont basées sur la position des nœuds. Les pré requis pour effectuer un routage géographique dans un réseau ad hoc [6,7] sont :

- Tous les nœuds possèdent un moyen de localisation, soit un système natif comme le GPS (*Global Position System*), soit un système logiciel comme un protocole de localisation.
- Un nœud source connaît toujours la position du nœud destinataire. Pour ce faire, soit tous les nœuds connaissent les positions initiales de tous les nœuds, soit un service de localisation doit être utilisé.

On distingue trois principales décisions de routage géographique, qui dépendent, soit de la progression, soit de la distance, soit de la direction. Dans ce qui suit, nous expliquons ces prises de décision par l'étude de l'exemple donnée par la figure 2.4.

Nous supposons que le nœud source S désire transmettre des informations vers le nœud D.

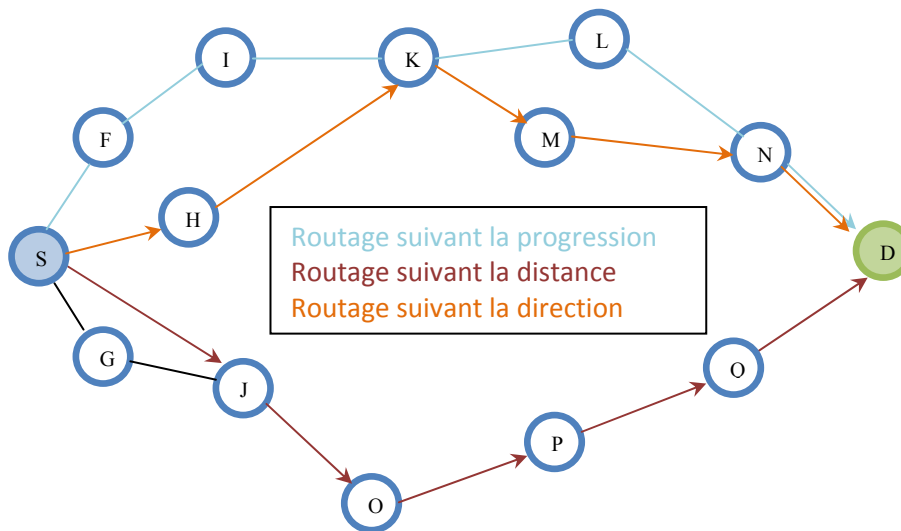


Figure 2.4 Routage basé sur la localisation

- **Suivant la progression** : La progression d'un nœud correspond à sa projection sur l'axe (SD). Le nœud dont la projection est la plus proche de la destination est choisi comme prochain nœud. Suivant la figure 2.4, le chemin résultant sera celui défini par les nœuds S F I K L N et D. Cette approche est communément appelée *Most Forward within Radius* (MFR). Si la progression minimale est considérée, la méthode est appelée *Nearest Forward Progress* (NFP).
- **Suivant la distance** : Le nœud le plus proche de la destination D en termes de distance est choisi comme prochain nœud. Dans notre cas de figure, c'est le nœud J qui est sélectionné. La route de S à D passe par les nœuds J O P Q. Dans ces approches, on cherche à minimiser le nombre de sauts.
- **Suivant la direction** : Le nœud voisin le plus proche de la droite (SD) en direction de D est choisi. Le nœud H est pris comme prochain nœud et le chemin entre S et D est S H K M N D. On minimise dans ce cas la distance réelle parcourue par les messages entre les nœuds S et D. Une variante de cette approche est de considérer tous les nœuds en direction de la destination appartenant à un cône dirigé vers D [7].

2.4. Les protocoles de routage pour les réseaux de capteurs

2.4.1. Propagation et discussion (flooding and gossiping):

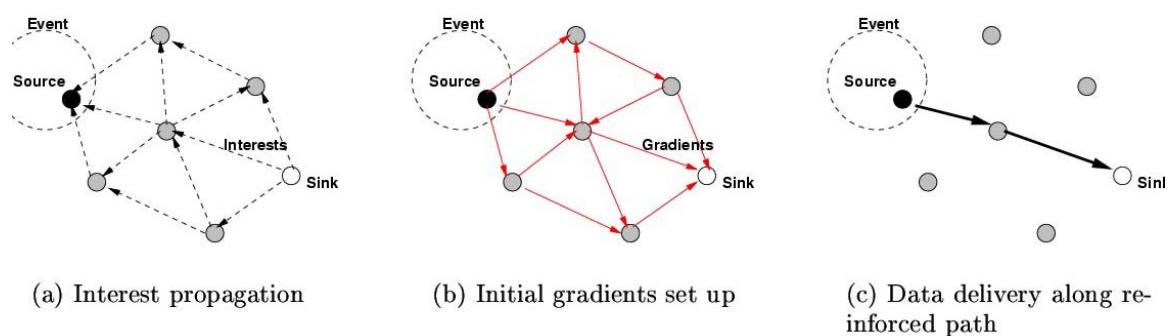
Pour le *flooding*, chaque capteur recevant un paquet de données le renvoie à tous ses voisins et ce processus continue jusqu'à ce que le paquet arrive à la destination ou le nombre maximum de sauts pour le paquet est atteint. D'autre part, le *gossiping* est une version légèrement améliorée du *flooding* où le nœud récepteur envoie le paquet à un voisin choisi aléatoirement.

2.4.2. Sensor Protocols for Information via Negotiation (SPIN):

L'idée derrière le SPIN est d'échanger des informations sur les données à envoyer en utilisant des paquets de signalisations spéciales nommées *meta-DATA*. Ceci permet d'éviter le problème des données redondantes. Chaque nœud, s'intéressant à la donnée référencée par ce paquet *meta-DATA*, peut les récupérer en envoyant un paquet de requête.

2.4.3. Diffusion dirigée (Directed Diffusion) :

Le protocole *Directed Diffusion* est simple dans son principe. Le nœud « *Sink* » commence à envoyer, vers tous les nœuds, un message *Interest* pour démarrer une application bien déterminée. Ce paquet sera acquitté par un autre appelé *gradient*. Un gradient est un lien de réponse de la part du voisin recevant l'intérêt. En utilisant les intérêts et les gradients, plusieurs chemins peuvent être établis entre le « *Sink* » et la source. L'un de ces chemins est sélectionné par renforcement. Si ce chemin échoue un nouveau ou un alternatif doit être identifié.



2.4.4. Routage par rumeur (Rumor routing):

L'idée est de transmettre les requêtes aux nœuds qui ont observé un événement particulier. Quand un nœud détecte un événement, il l'ajoute à sa table locale et génère un paquet nommé *agent*. Ce paquet parcourt le réseau afin de propager des informations sur les événements

locaux aux nœuds distants. Quand un nœud génère une requête pour un événement, les nœuds qui connaissent l'itinéraire, peuvent répondre en se référant la table d'événement.

2.4.5. CADR (Constrained Anisotropic Diffusion Routing):

CADR est réalisé en activant seulement les capteurs qui sont proches d'un événement particulier et en ajustant dynamiquement les chemins de données. Avec CADR, chaque nœud évalue un objectif (information/coût) et conduit les données suivant le gradient local et les besoins de l'utilisateur final.

2.4.6. COUGAR :

C'est un protocole qui considère le réseau comme un énorme système de base de données répartie. L'idée principale est d'employer des requêtes déclaratives afin d'abstraire le traitement des requêtes des fonctions de la couche réseau et d'utiliser l'agrégation de données pour économiser de l'énergie. L'abstraction est soutenue par une nouvelle couche de requête entre les couches réseau et application.

2.4.7. ACQUIRE :

L'approche considère le réseau comme une base de données répartie et bien adaptée pour les requêtes complexes. Une requête est expédiée par le *sink* et chaque nœud recevant la requête peut répondre partiellement en employant l'information dans son cache ou recueille l'information de ses voisins accessible puis la transmet à un autre capteur. Une fois la requête complètement résolue, elle est renvoyée au *sink*.

2.4.8. LEACH :

L'idée est de former des clusters de nœuds capteurs en se basant sur la force du signal reçue et d'employer le *cluster-head* local comme routeur du *sink*. Tout traitement de données est local au cluster. Le rôle de *cluster-head* est échangé aléatoirement entre les nœuds afin d'équilibrer les charges.

2.4.9. PEGASIS (Power Efficient GATHERing in Sensor Information Systems):

PEGASIS est une amélioration du protocole LEACH. Au lieu de former plusieurs clusters, PEGASIS forme des chaînes de nœuds de sorte que chaque nœud transmet et reçoit du nœud voisin appartenant à la chaîne. Un seul nœud est choisi, parmi cette chaîne, pour transmettre au *sink*. Ce nœud est nommé « *leader node* ». Les données recueillies se déplacent d'un nœud à un autre, et seront agrégées puis envoyées au *sink* par le nœud *leader*.

Dans le PEGASIS hiérarchique les nœuds construisent une chaîne qui forme un arbre hiérarchique. Chaque nœud leader, choisi dans un niveau particulier, transmet des données aux nœuds du niveau supérieur de la hiérarchie jusqu'à atteindre la station de base *Sink*.

2.4.10. TEEN et APTEEN:

TEEN (*Threshold sensitive Energy Efficient sensor Network protocol*) est un protocole hiérarchique conçu pour être sensible aux changements imprévus des attributs détectés tels que la température. L'architecture du réseau est basée sur un groupement hiérarchique où les nœuds les plus proches forment des clusters. Après la construction des clusters, le *cluster-head* diffuse deux seuils aux nœuds. Qui sont la valeur minimale d'un attribut pour pouvoir être transmis et le degré minimale du changement de cet attribut.

Le TEEN adaptatif (APTEEN) est une extension du TEEN basée sur la capture périodique des données et la réaction aux événements temps-réel. Quand la station de base forme les clusters, les cluster-heads diffusent les attributs, les seuils et le plan de transmission à tous les nœuds et effectuent également l'agrégation des données afin d'économiser l'énergie.

2.4.11. MECN et SMECN :

MECN (*Minimum Energy Communication Network*) utilise le GPS a basse puissance. L'idée principale est de trouver un sous-réseau, qui aura moins de nœuds et exige moins de puissance pour la transmission entre deux nœuds particuliers quelconques. SMECN (Small MECN) est une extension de MECN. Le sous-réseau construit par SMECN est probablement plus petit (en termes de nombre d'arcs) que celui construit par MECN si les diffusions peuvent atteindre tous les nœuds dans une région circulaire autour du diffuseur.

2.4.12. GAF (Geographic adaptive fidelity):

Le protocole GAF conserve l'énergie par la mise en veille des nœuds inutiles dans le réseau sans affecter le niveau de fidélité du routage. Il forme une grille virtuelle pour le domaine couvert. Chaque nœud emploie sa position indiquée par le GPS pour s'associer à un point dans la grille virtuelle. Des nœuds liés au même point sur la grille sont considérés équivalents en termes de coût de routage. Une telle équivalence est exploitée en maintenant quelques nœuds situés dans un secteur particulier de la grille dans l'état de sommeil afin d'économiser de l'énergie.

2.4.13. GEAR (Geographic and Energy-Aware Routing):

L'idée est de restreindre le nombre d'intérêts dans la diffusion dirigée en considérant seulement certaine région plutôt que d'envoyer les intérêts au réseau entier.

2.4.14. SAR (Sequential Assignment Routing):

SAR est une approche multi-chemins qui s'efforce à réaliser l'efficacité énergétique et la tolérance aux fautes. SAR crée des arbres en prenant en compte les métriques QoS, la ressource énergétique sur chaque chemin et le niveau de priorité de chaque paquet. En utilisant ces arbres, des routes multiples du *sink* aux capteurs sont formés. Une ou plusieurs routes peuvent, alors, être emprunter.

2.4.15. SPEED :

Ce protocole exige que chaque nœud maintienne des informations sur ses voisins. Il procède par un routage géographique pour sélectionner les nœuds de prochains sauts jusqu'à arriver à la destination finale *Sink*. En outre, SPEED assure une vitesse de livraison des paquets constante, qu'on note *Setspeed*. Ceci permet de garantir des délais de livraison de bout en bout acceptables. Ces délais peuvent être estimés en divisant la distance séparant le nœud source du *sink* par la vitesse *setspeed*.

2.5. Conclusion :

La technologie des réseaux de capteurs reste très prometteuse, et leur défis majeurs est de trouver des protocoles de routage qui permettent, à la fois, de :

- consommer le moins d'énergie possible,
- assurer la connectivité du réseau et la couverture du champ surveillé,
- assurer une livraison fiable et rapide,
- tolérer aux pannes,
- s'adapter aux changements de topologie ...

Nous avons présenté dans ce chapitre les différents protocoles de routage déployés dans les réseaux de capteurs. Dans la suite, nous nous intéressons plus aux protocoles de routage basé sur l'assurance de la QoS.

Protocoles de routage avec QoS :

SPEED, MMSPEED et E2MSPEED

3.1. Introduction :

Les réseaux des capteurs forment un nouveau environnement différent des réseaux traditionnels et bien adapté au service des applications critiques comme les alertes des catastrophes naturelles ou bien les applications militaires comme la détection des intrusions, la surveillance des champs de bataille... Pour répondre aux exigences des applications pareilles, il a fallu développer de nouveaux mécanismes de support de la QoS. Ces mécanismes doivent, à la fois, optimiser la fiabilité temps réelle de ses réseaux et respecter leurs spécifications (la densité des nœuds, leurs durées de vie limitées, la topologie instable due à la mobilité, l'ajout ou la défaillance des nœuds ...).

En fait, les protocoles de qualité de service existants, comme ceux utilisés dans les réseaux Ad Hoc, se basent sur la découverte des chemins de bout en bout, la réservation des ressources et la récupération des chemins en cas de changement de la topologie de réseau. Ces caractéristiques rendent ces protocoles non adaptés à la nature des réseaux de capteurs et leurs applications : Tout d'abord, le temps mis pour la découverte de chemin augmente le délai de bout en bout, chose non tolérable surtout pour les trafics très urgents. Ensuite, la réservation des ressources pour des trafics imprévisibles, comme les trafics urgents, apparaît illogique. De ce fait, le développement de nouveaux protocoles de qualité de service pour les réseaux de capteurs s'avère indispensable.

Dans ce chapitre, nous présentons une étude comparative des deux protocoles de routage dédiés pour la garantie de la qualité de service temps réel dans les réseaux des capteurs à savoir SPEED et MMSPEED, puis nous procédons à une optimisation des performances temps réel des réseaux de capteurs.

3.2. Le protocole SPEED :

Le protocole SPEED [8] est classé parmi les protocoles de routage géographique, basé sur la qualité de service. Sa caractéristique fondamentale est la garantie d'un délai de livraison de

bout en bout optimale. Grâce à cette spécification, SPEED est le protocole le plus approprié pour des applications temps réel des réseaux des capteurs [8]. En effet, ce protocole essaye de garantir une vitesse de transmission des paquets constante dans tout le réseau.

Afin d'assurer le routage avec qualité de service temps réel, SPEED fait appel à l'interfonctionnement de plusieurs modules, comme l'illustre la figure 3.1.

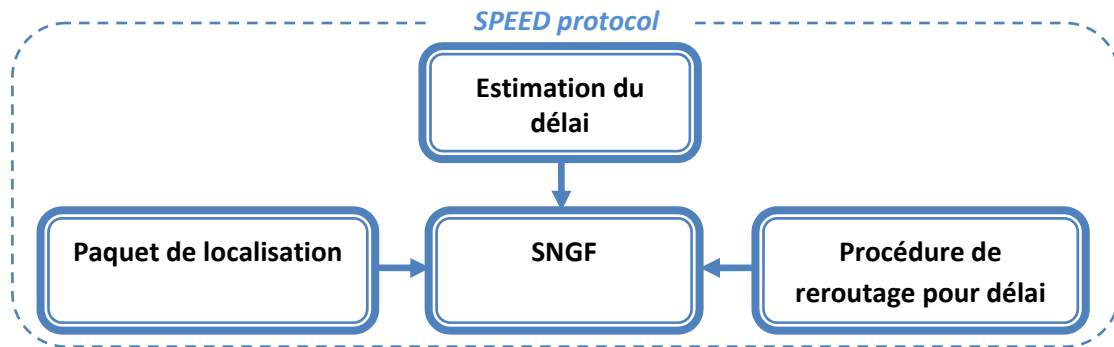


Figure 3.1 Architecture fonctionnelle du protocole SPEED

Les modules mis en évidence par l'architecture fonctionnelle de SPEED opèrent de la façon suivante :

Le *SNGF (Stateless Non-deterministic Geographic Forwarding)* est le module de routage, responsable de la sélection du prochain saut offrant la vitesse de livraison des paquets souhaitée. Le *Backpressure Rerouting* est appelé pour re-router les paquets si un cas de congestion du nœud de prochain saut est rencontré. *Delay Estimation* est le mécanisme permettant la prédiction du délai nécessaire pour la transmission des paquets vers le prochain saut. *Beacon Exchange* permet la localisation géographique des nœuds.

3.2.1. Paquet de localisation :

Les nœuds, exécutant le protocole SPEED, diffusent périodiquement des paquets de localisation pour tenir à jour les changements de la topologie du réseau. Les nœuds récepteurs interprètent les informations reçues et les sauvegardent pour construire leurs tables de voisinage. Ces paquets ont le format suivant:



Figure 3.2 Paquet de localisation

- Le champ *NeighborID* identifie le voisin émetteur de ce paquet de localisation.
- Le champ *Position* contient la position géographique du nœud émetteur du paquet.
- Le champ *SendToDelay* est le délai estimé pour atteindre le nœud destination identifié par *Global_ID*.

- Le champ *ExpireTime* est un *timer* désignant la validité de l'information transportée, c'est-à-dire, à son expiration, le nœud supprimera ces informations de sa table de voisinage. Le nombre de paquets de localisation échangé devient de moins en moins important si les nœuds se déplacent lentement ou sont fixes.

3.2.2. Estimation de délai :

Le module « *Delay estimation* », utilisé par le protocole SPEED, joue un rôle très déterminant dans la maintenance et la garantie de la livraison des paquets dans les délais exigés.

Les réseaux de capteurs disposent généralement d'une bande passante limitée [8], ce qui rend difficile l'utilisation des paquets de signalisation propre à l'estimation des délais de livraison. Pour cela, l'exploitation des paquets *DATA*, passant par un nœud capteur, s'avère la meilleure solution pour ce genre de mesure.

- **Estimation du délai intermédiaire :**

L'estimation du délai se fait au niveau du nœud émetteur. Celui-ci enregistre instantanément le temps de départ du dernier bit du paquet *DATA*, noté $t_{ij}^{départ}$, et calcule le temps mis pour effectuer un seul saut quand il reçoit l'acquittement de son paquet envoyé. À son arrivée au nœud j , ce dernier acquitte la réception du paquet *DATA* en indiquant le temps écoulé du traitement du paquet *ACK*, noté t_{ij}^{ACK} . Ainsi, le nœud émetteur enregistre le temps d'arrivée du paquet *ACK*, noté $t_{ij}^{arrivé}$, et mesure le temps mis pour effectuer un seul saut suivant la formule suivante :

$$t_{ij}^{saut} = t_{ij}^{arrivé} - t_{ij}^{ACK} - t_{ij}^{départ}$$

L'estimation du délai courant est donnée en combinant cette nouvelle valeur calculée avec les délais antérieurs suivant la méthode *Exponential Weighted Moving Average* (EWMA) [9].

3.2.3. Stateless Non-deterministic Geographic Forwarding (SNGF)

- **Calcul de la vitesse de livraison :**

L'exemple suivant explique la méthode de calcul de la vitesse de livraison entre deux nœuds voisins :

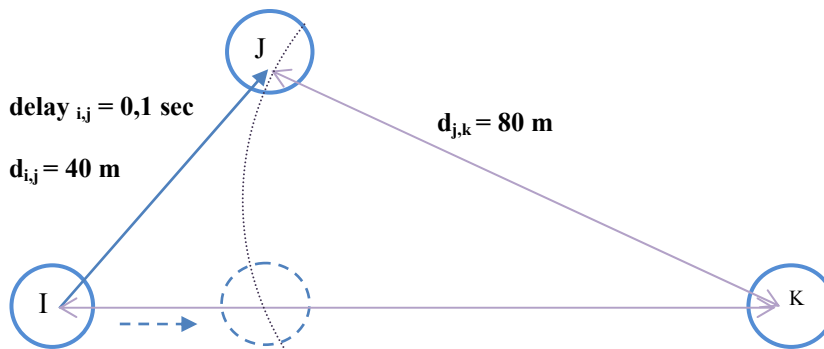


Figure 3.3 Calcul de la vitesse de livraison

Nous considérons deux nœuds voisins i et j (c.f figure). Les distances leurs séparant de la destination k sont respectivement $d_{ik} = 100\text{m}$ et $d_{jk} = 80\text{m}$. Si le nœud i transmet ses paquets vers le nœud j dans un délai $delay_{ij} = 0.1$ seconde, ce qui permet de s'approcher de la destination d'une distance d égale à $(d_{ik} - d_{jk}) = 20\text{m}$. Dans ce cas, la vitesse de transmission intermédiaire $speed_{ij}^k$ se mesure comme le rapport $(d_{ik} - d_{jk}) / delay_{ij} = 200\text{m/s}$.

Il est clair que la vitesse de transmission intermédiaire change d'un saut à un autre puisqu'elle varie en fonction de la distance entre les nœuds voisins et le temps de transmission. Si chaque nœud du réseau envoie les paquets vers le voisin en maintenant une vitesse supérieure à une vitesse seuil $setspeed$, alors cette dernière est forcément garantie par tous les nœuds. Ainsi de n'importe quel nœud source vers le nœud destination « Sink », le délai de bout en bout est estimé comme suit [8]:

$$\text{Délai bout en bout} = D_{\text{source, Sink}} / \text{Setspeed}$$

avec $D_{\text{source, Sink}}$: distance entre le nœud source et la destination finale « Sink ».

• **Choix de nœud de prochains sauts :**

Le *SNGF* (*Stateless Non-deterministic Geographic Forwarding*) est le module responsable de la sélection du prochain saut offrant la vitesse de livraison des paquets souhaitée. Nous expliquons son principe en se basant sur l'exemple de la figure 3.4 :

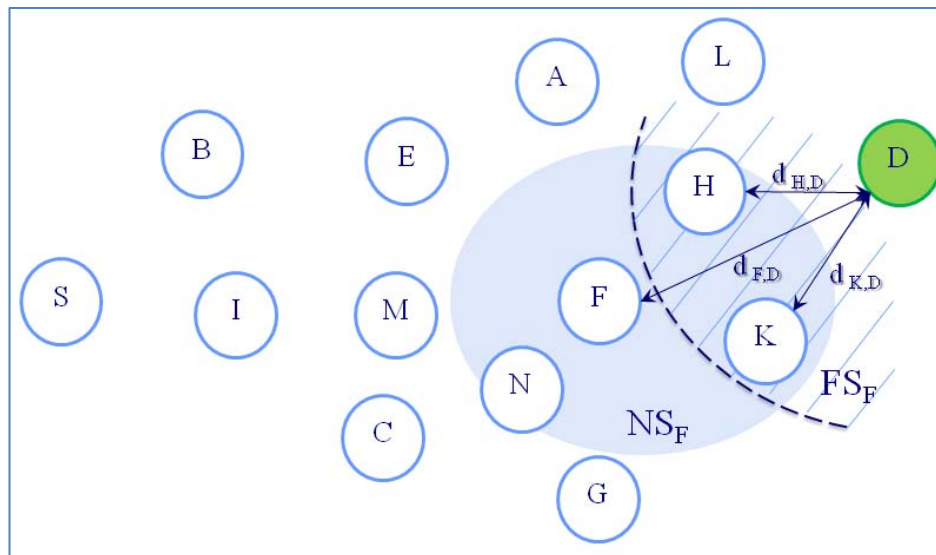


Figure 3.4 Choix de nœud de prochain saut

Le nœud source F désire transmettre un paquet DATA vers la destination D. il doit alors choisir le nœud de saut suivant tout en respectant les règles suivantes :

1. Les paquets ne sont transmis que vers les nœuds se trouvant dans le champ FS_F (*Forwarding candidate Set of Node F*). Ce champ contient les nœuds qui sont à la portée du nœud F et qui sont susceptibles d'être sélectionnés pour continuer la transmission des données vers le nœud destination.

Le champ FS_F groupe les nœuds de prochain saut qui sont plus proche à la destination que le nœud F et qui offre une vitesse de livraison supérieure ou égale à la vitesse de seuil *Setspeed*. Ceci est exprimé par [8]:

$$FS_F = \{ \text{nœud } j \in NS_F / d_{F,D} - d_{j,D} > 0 \text{ et vitesse de livraison} \geq \text{Setspeed} \}$$

avec NS_F : (*Neighbors Set*) nœuds voisins du nœud i,

$d_{F,D}$: la distance séparant le nœud i du nœud destination

$d_{j,D}$: distance séparant le nœud j de prochain saut du nœud destination.

2. Le nœud du prochain saut devra être choisi parmi le premier groupe et disposant de la vitesse de transmission la plus importante. (décrit dans le paragraphe 3.2)
3. Dans le cas où la zone FS_F est dépourvue des nœuds, autrement dit, si le nœud F échoue de trouver un prochain saut, il rejette alors le paquet en question et envoie des paquets *Backpressure* (décrit dans le paragraphe 3.4) aux nœuds précédents pour leurs interdire momentanément la transmission des paquets vers sa région.

3.2.4. Procédure de re-routage pour délai :

a) Format du paquet « Delay Back pressure » :

Le paquet « *Delay Back pressure* » comporte les champs suivants :



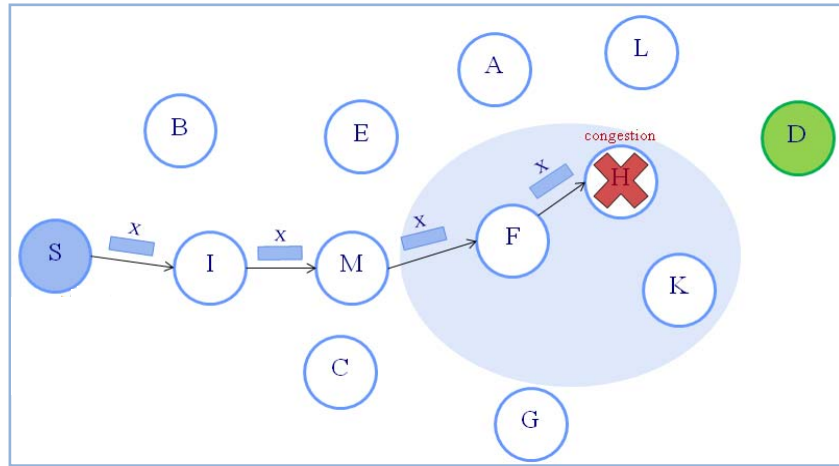
Figure 3.5 Paquet « *Delay Back pressure* »

- le champ *ID* contient l'identité du nœud émettant le paquet « *Delay Back pressure* ».
- le champ *Destination* contient l'identité du nœud destination.
- le champ *SendToDelay* contient la durée de livraison estimée entre les deux nœuds *ID* et *Destination*. Cette durée est le délai intermédiaire nécessaire, $\text{delay}_{ID, Destination}$, pour effectuer le saut entre ces deux derniers. Ce champ prend la valeur infinie ∞ , si le nœud *ID* ne trouve aucun nœud voisin pour lui envoyer le paquet DATA.

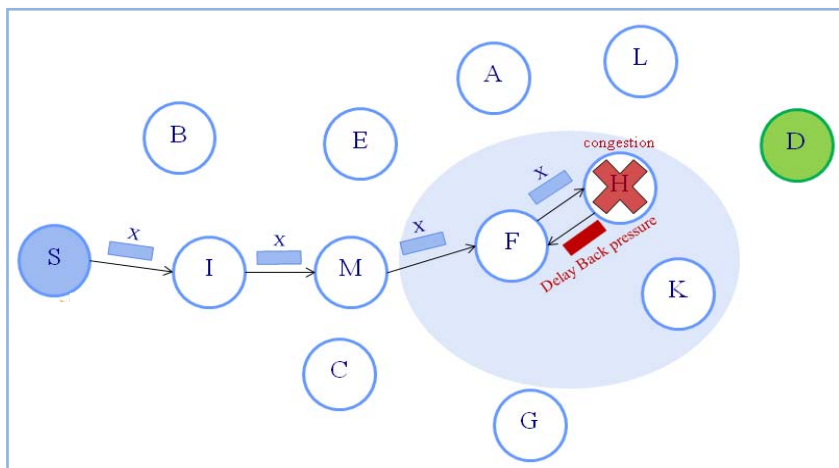
b) Procédure de re-routage en cas de congestion :

Le protocole SPEED utilise une procédure de re-routage appelée « *Backpressure rerouting* », si au cours de route, on rencontre des nœuds congestionnés.

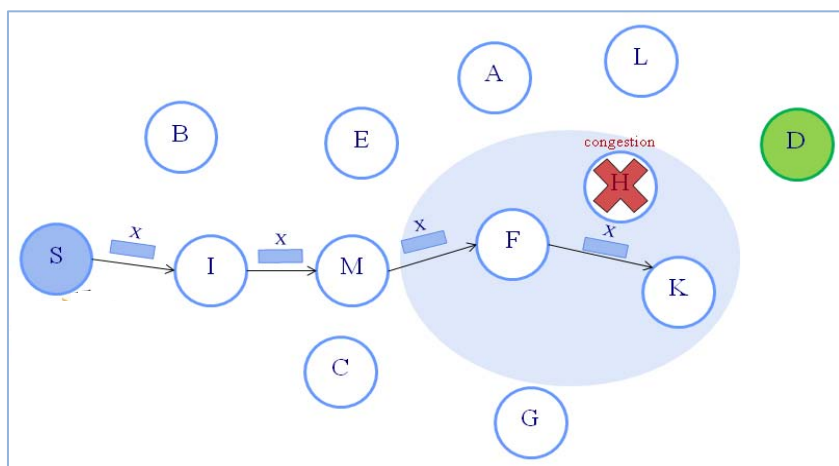
L'exemple illustré par les figures 3.6 (a, b, c, d), décrit clairement la procédure de re-routage :



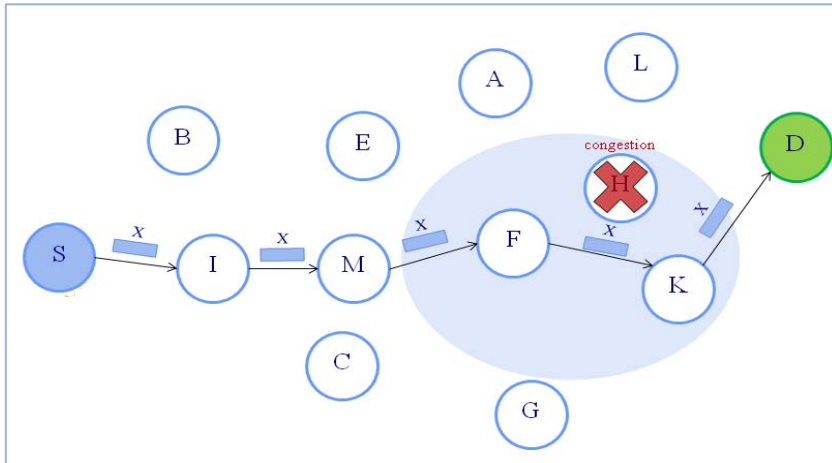
(a)



(b)



(c)

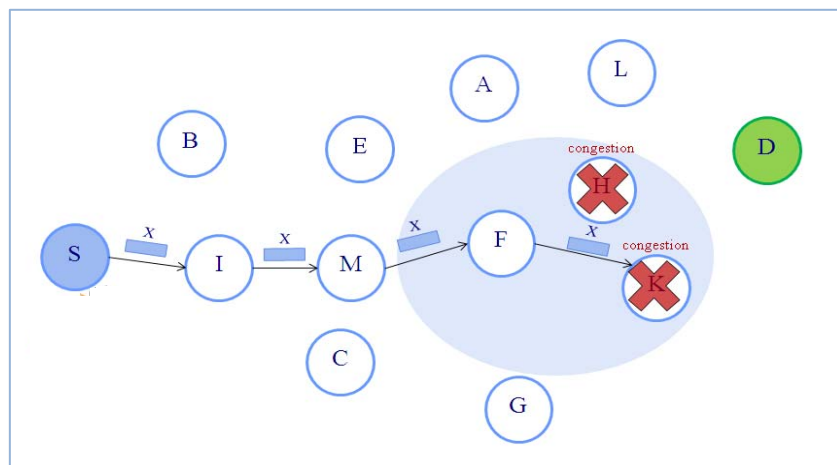


(d)

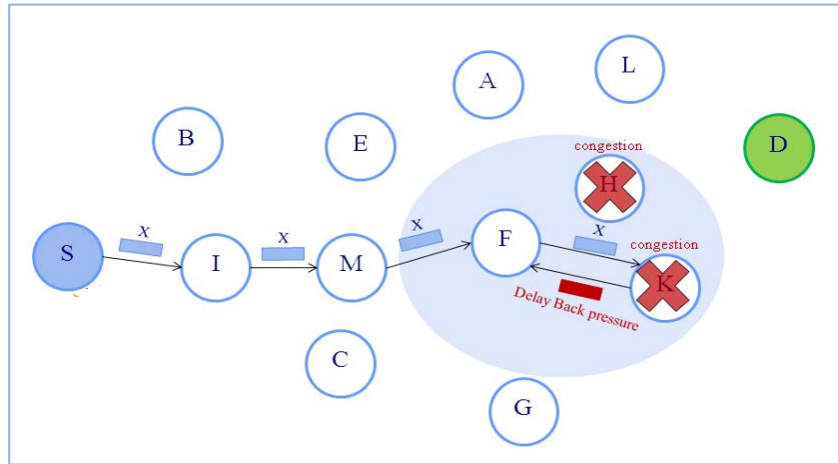
Figure 3.6 Procédure de re-routage pour délai « Delay back pressure rerouting » (cas 1)

Le nœud S désire transmettre un paquet DATA, noté X, jusqu'à la destination D. En arrivant au nœud F, celui-ci choisit comme prochain saut le nœud H (le choix du nœud est effectué par le module SNGF). Mais il se trouve que ce nœud est congestionné, alors il répond le nœud F par un paquet « *Delay back pressure* » pour lui informer qu'il ne peut pas délivrer le paquet X dans un délai favorable vu son état de congestion. Le nœud F active, alors, son module SNGF pour re-sélectionner un autre nœud appartenant à l'ensemble FS_F (*Forwarding Set* pour le nœud F), et il finit par re-router le paquet X au nœud K.

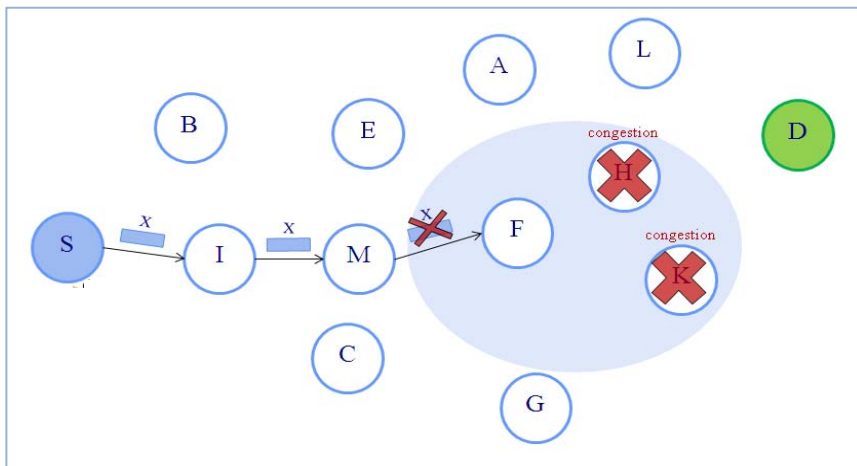
Un deuxième cas peut se présenter si tous les nœuds de l'ensemble FS_F est en état de congestion. Ce cas de figure est illustré par les figures II.8. (e,f,g,h,i):



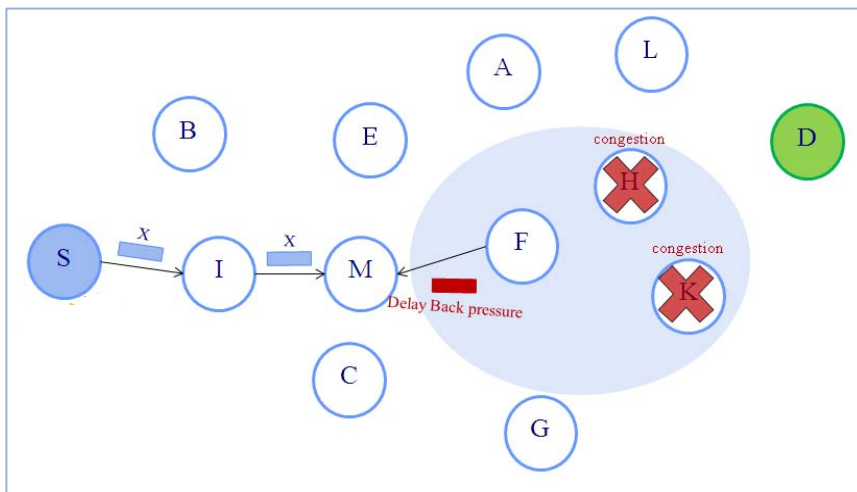
(e)



(f)



(g)



(h)

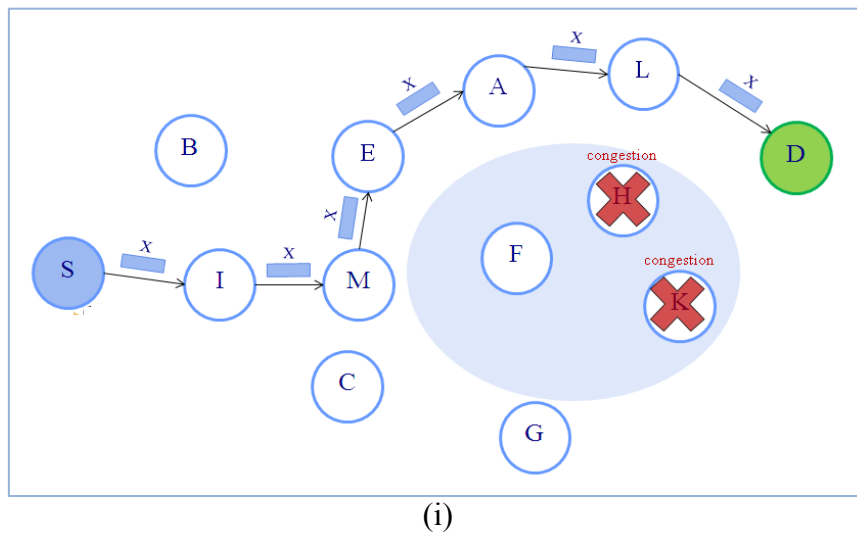
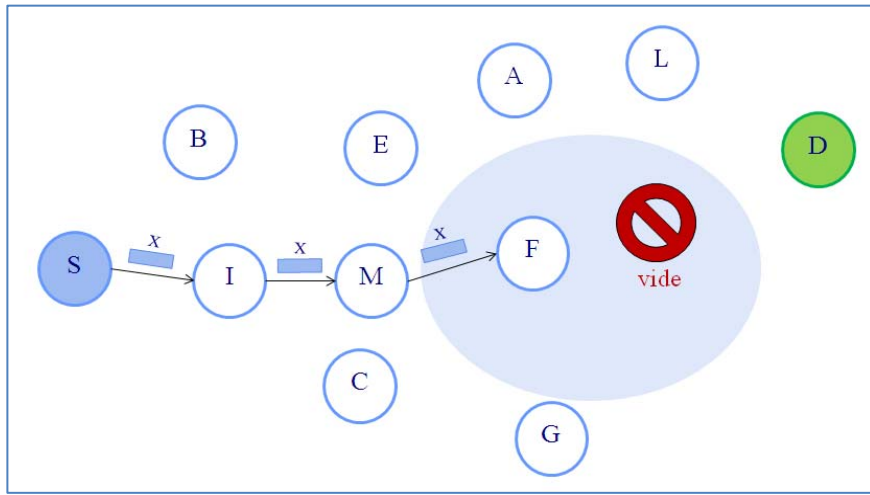


Figure 3.7 Procédure de re-routage pour délai « Delay back pressure rerouting » (cas 2)

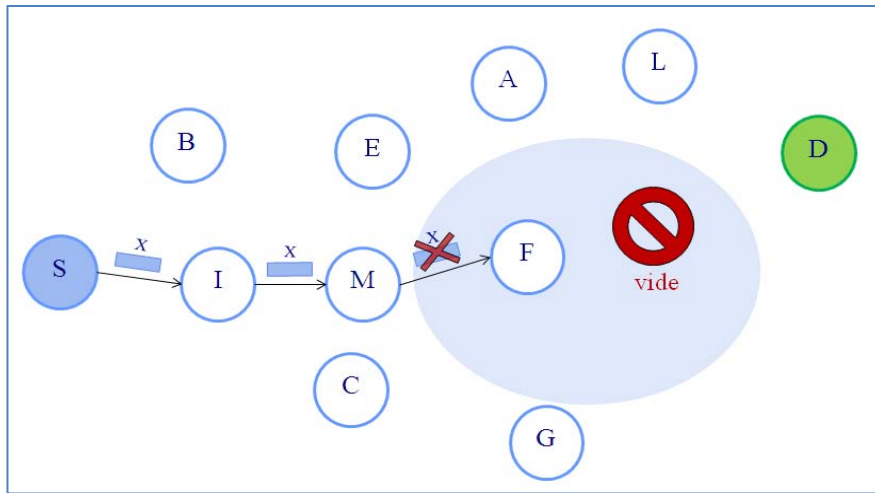
Reprenant les étapes précédentes (a), (b), (c) et (d) de la figure 3.6. Supposant qu'à l'étape (d), le nœud F trouve, aussi, le nœud K en état de congestion. Ce dernier va lui envoyer, alors, le paquet « *Delay back pressure* ». Supposant maintenant que F ne trouve aucun autre nœud susceptible d'être celui du prochain saut. Ce nœud F supprime, alors, le paquet DATA et envoie au nœud M un paquet « *Delay back pressure* » lui indiquant qu'il a échoué de transmettre ce paquet X. Dans ce cas, le nœud M active son SNGF pour sélectionner un autre prochain saut (dans notre exemple, le nœud E est choisi. De même, si le nœud M trouve tous ses voisins, appartenant à son champ FS_M , congestionnés, il réagit de la même façon que le nœud F et le nœud I devra alors acheminer les paquets vers un autre nœud.

c) Void avoidance :

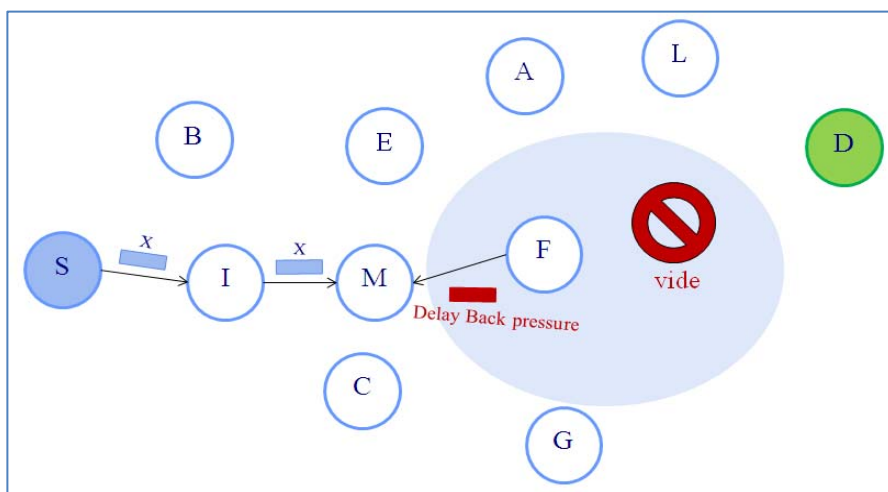
Un troisième cas peut se présenter si l'ensemble des nœuds de prochains sauts FS_F est vide. Le protocole SPEED sait résoudre ce genre de problème en se basant sur la notion de *Void Avoidance* illustré par les figures II.9. (a,b,c) :



(a)



(b)



(c)

Figure 3.8 Mécanisme de *Void avoidance*

D'après le scénario de ces figures, le nœud F ne trouve aucun nœud voisin qui mène vers la destination D, par conséquent, il supprime le paquet DATA et envoie le paquet « *Delay Back pressure* », contenant les champs suivants (*ID*, *DESTINATION*, ∞), vers le nœud du saut précédent. à son tour, le nœud M affecte la valeur ∞ au champ *SendToDelay* pour dire que le choix du nœud F, comme un intermédiaire, nécessite un délai de transmission très important, donc il ne va plus être sélectionné comme un prochain saut vers la destination finale et le nœud M se trouve obligé de choisir un prochain saut différent.

3.2.5 Le protocole de la couche MAC: DCF

Le protocole SPEED n'exige pas l'implémentation des mécanismes de QoS au niveau de la couche MAC. Pour cela, le protocole DCF (*Distributed Coordination Function*) a été choisi au niveau de cette couche [10].

En effet, DCF peut supporter seulement le service *Best-Effort*. Il n'y a aucun mécanisme de différenciation de service selon la priorité des trafics. Les paquets des flux urgents et moins urgents sont stockés ensemble dans une même file d'attente FIFO, et donc, le traitement de ces paquets urgents peut probablement être retardé par celui des paquets moins urgents, chose qui dégrade la qualité de service temps réel en terme de délai de bout en bout.

En conclusion, le protocole SPEED se caractérise essentiellement par la garantie des délais réduits de bout en bout, ce qui lui permet de répondre aux exigences des applications temps réel. Cependant, avec l'existence des trafics de priorités différentes, SPEED se montre incapable de les servir selon leurs priorités, puisqu'il offre une seule vitesse de livraison indépendamment du degré de priorité de flux. Par conséquent, il ne peut pas assurer des délais de livraison de bout en bout convenables aux exigences temps réel de chaque flux de trafic. Dans la suite, nous présentons une extension de ce protocole appelée MMSPEED qui résout ce genre de problème en supportant la qualité de service selon la contrainte de délai de bout en bout [11].

3.3. MMSPEED : Multi Path Multi SPEED

En s'inspirant du protocole précédent, le protocole MMSPEED présente une évolution dans les protocoles avec QoS.

De plus des avantages hérités du protocole SPEED, MMSPEED se caractérise par l'offre de plusieurs vitesses de transmission et la possibilité d'établissement de plus qu'un chemin vers la destination. En effet, chaque vitesse offerte permet de définir un niveau de QoS temporel et chaque route supplémentaire aide à améliorer la qualité de trafic. Ces deux mécanismes

permettent respectivement de respecter le degré de criticité de chaque application, de transmettre les paquets dans les délais exigés , d'éviter les problèmes fréquemment rencontrés comme la congestion et de réduire le taux de perte des paquets...

Par le biais de ces deux mécanismes, le MMSPEED se distingue par l'offre de plusieurs niveaux de QoS suivant les exigences des trafics.

3.3.1. Garantir une QoS temporelle :

Le protocole MMSPEED permet la livraison des paquets suivant différentes vitesses de transmission selon leurs urgences. Plus le degré d'urgence est important, plus les paquets sont transmis suivant une vitesse de transmission importante et plus le délai de bout en bout est court. Cette technique permet de différencier le service selon la priorité temporelle du trafic.

Ainsi, la couche réseau de ce protocole est une couche multi vitesses, qui peut être modélisée comme plusieurs couches virtuelles, offrant chacune, une seule vitesse de transmission seuil, noté $setspeed_L$ telle que $1 < L < \text{nombre totale des vitesses offertes}$.

a) Classification des trafics :

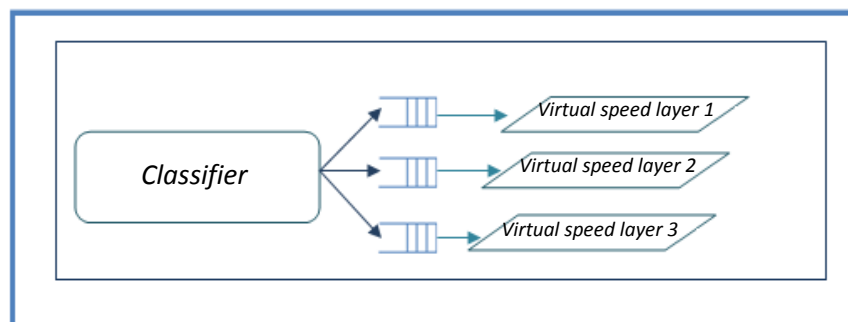


Figure 3.9 Modélisation de la couche réseau du protocole MMSPEED

Comme illustré par la figure 3.9, une classification virtuelle des trafics au niveau de chaque nœud est appliquée pour classer les paquets selon leurs urgences. Pour commencer, le « classifier » du nœud source S calcule la vitesse de transmission d'un paquet DATA, noté X , en fonction de la distance $d_{S,D}(X)$, lui séparant du nœud destination D, reçue suite à l'échange des paquets de localisation, et le délai de bout en bout souhaité $deadline(X)$ (ce délai sera indiqué dans le paquet lui-même). La vitesse minimale demandée $Reqspeed(X)$ pour respecter le délai de bout en bout est calculée comme le rapport :

$$Reqspeed(X) = d_{S,D}(X) / deadline(X)$$

Suivant cette nouvelle donnée, le « classifier » sélectionne la couche virtuelle L dont sa vitesse de transmission de seuil $setspeed_L$ vérifie la condition suivante [11] :

$$setspeed_L = \min_{j=1}^L \{setspeed_j / setspeed_j \geq Reqspeed(X)\}$$

b) Mise à jour du délai de bout en bout restant :

Pour bien déterminer la vitesse *Reqspeed*, après chaque saut, les nœuds intermédiaires doivent mettre à jour le délai de bout en bout restant $deadline(X)$. Pour se faire, chaque nœud intermédiaire mesure le temps écoulé dans le traitement local du paquet, noté $t^{elapsed\ time}$, et le transmette vers le nœud de saut suivant. De cette façon, tout nœud intermédiaire peut estimer correctement le temps restant pour atteindre la destination finale sans dépasser le délai *deadline* global.

Pour calculer le temps écoulé avec précision, la couche MAC du nœud intermédiaire, recevant le dernier bit d'un paquet DATA, enregistre instantanément le temps d'arrivée de ce paquet, $t^{arrival}$. Après que le nœud en question traite localement le paquet, sa couche MAC date l'instant où le premier bit du paquet est envoyé à la couche physique, noté $t^{departure}$. On peut aussi calculer le temps nécessaire pour la transmission de tout le paquet, noté $t^{transDelay}$, en fonction de sa longueur et le taux de transmission. Ainsi, le temps écoulé au niveau du nœud capteur est mesuré comme suit :

$$t^{elapsed\ time} = t^{departure} + t^{transDelay} - t^{arrival}$$

La valeur $t^{elapsed\ time}$ calculée, sera transportée avec le paquet DATA. Une fois reçu par le nœud de saut suivant, ce dernier peut dégager l'information $t^{elapsed\ time}$ et met à jour le délai de bout en bout restant comme suit :

$$deadline(X) = deadline(X) - t^{elapsed} - t^{propDelay}$$

$t^{propDelay}$ représente le temps de propagation entre les deux nœuds voisins. Ce temps peut être négligé.

Suivant la valeur de la variable $deadline(X)$, tout nœud intermédiaire i peut vérifier si la vitesse courante $setspeed_L$ est suffisante pour transmettre les paquets dans le délai exigé, c'est-à-dire, vérifier si $(d_{i,D} / setspeed_L) \leq deadline(X)$, avec $d_{i,D}$ distance entre le nœud intermédiaire i et la destination D. si cette comparaison n'est pas vérifiée, le nœud en question recalcule la vitesse demandée $Reqspeed = d_{i,D} / deadline(X)$ et re-selecte la vitesse de seuil adéquate $setspeed_L = \min_{j=1}^L \{setspeed_j / setspeed_j \geq Reqspeed(X)\}$, et re-choisit la couche réseau virtuelle L qui offre cette vitesse de seuil.

Grâce à ces deux mécanismes de classification des trafics et mise à jour du délai de bout en bout restant, le protocole MMSPEED a pu garantir la livraison des paquets dans les délais désirés. Ce protocole intègre aussi un troisième mécanisme pour assurer la qualité de trafic désirée résultant ainsi des taux de livraison des paquets importants.

3.3.2. Garantir une QoS en fonction du PDR:

Deux contraintes majeures peuvent dégrader la performance d'une route choisie à savoir la congestion de ses nœuds ou bien leurs défaillances, car ils peuvent augmenter le risque de rejet des paquets. Le protocole MMSPEED a pu diminuer ce risque en adoptant un routage multi-chemin, et la nature dense des réseaux des capteurs a été un facteur favorisant vu que plusieurs chemins redondants existent, menant vers la même destination. En fait, le routage multi-chemin est adopté, seulement, pour les trafics exigeant des taux de livraison important. Dans le cas contraire, les paquets seront routés dans un seul chemin. Cette idée a été bien illustrée par la figure suivante :

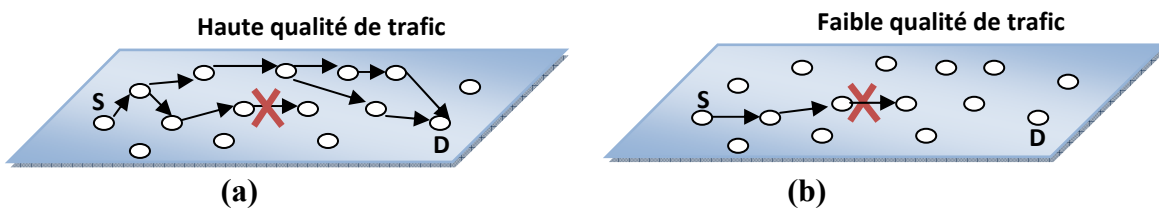


Figure 3.10 Différentiation de la qualité de trafic

Pour la faible qualité de trafic, la garantie d'arrivée de plus grand nombre des paquets est condamné par la probabilité de rencontrer l'une des contraintes majeurs citées précédemment dans la route adoptée, alors que, pour la haute qualité de trafic, on ne rencontre pas cette problématique vu la variété des états des routes qui peuvent transporter les paquets.

a) Choix des nœuds de prochain saut :

Pour assurer la qualité de trafic souhaitée (le taux de livraison souhaité : *desired Packet Delivery Ratio*), chaque nœud capteur doit être capable de décider vers quel(s) nœud(s) voisin(s) il va diriger les paquets. En effet, chaque nœud peut maintenir le dernier taux des paquets rejetés, c'est-à-dire que un nœud intermédiaire i peut se souvenir du dernier taux de rejet, noté $e_{i,j}$, par rapport à son voisin j . Moyennant cette probabilité $e_{i,j}$, le nœud i peut estimer quel niveau de qualité de trafic sera maintenu s'il choisit le nœud j comme prochain saut.

On a supposé que le taux de rejet de paquet est le même pour tous les nœuds, alors, la probabilité de livraison offerte peut être estimée comme suit [11]:

$$\text{Reaching Probability: } RP_{i,j}^d = (1 - e_{i,j}) (1 - e_{i,j})^{[d_{jD} / d_{i,j}]}$$

avec :

$(1 - e_{i,j})$ désigne le taux de paquets délivrés avec succès par le nœud i vers le nœud j .

$(1 - e_{i,j})^{[d_{jD} / d_{i,j}]}$ désigne le taux de paquets délivrés avec succès par le nœud i vers le nœud D en passant par le nœud j .

Suivant la valeur de RP_{ij}^d , on peut déterminer le nombre des nœuds de prochain saut pour satisfaire la probabilité de livraison exigée.

Le calcul du nombre des nœuds de prochain saut suit une procédure très simple :

Tout d'abord, on initialise à zéro une variable TRP (*Total Reaching Probability*) et chaque fois que le nœud i désigne un voisin j comme un saut suivant tout en se basant sur la valeur de $e_{i,j}$, la valeur de TRP sera mise à jour comme suit [11]:

$$TRP = 1 - (1-TRP) (1- RP_{i,j}^d)$$

Avec $(1-TRP)$ désigne la probabilité qu'aucune route ne peut transporter les paquets vers la destination finale D.

$(1- RP_{i,j}^d)$ désigne la probabilité que la route passant par le nœud j échoue au cours de la transmission des paquets vers la destination finale D.

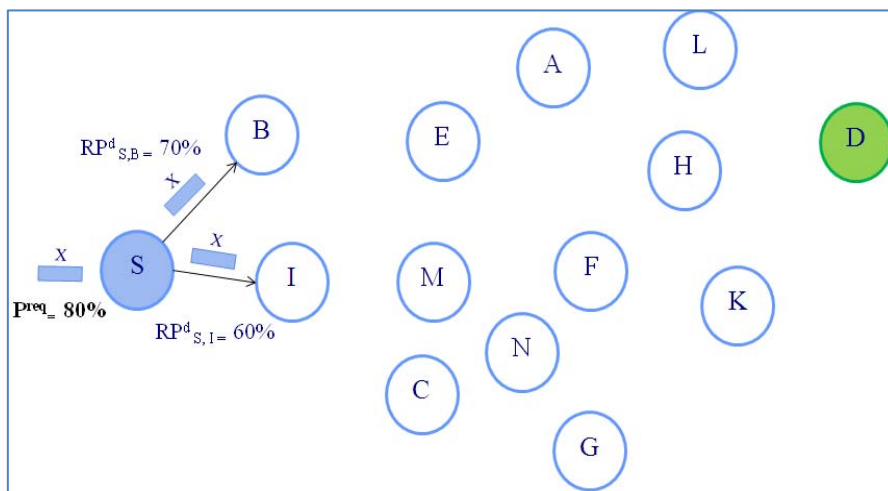
Ainsi, $(1-TRP) (1- RP_{i,j}^d)$ désigne la probabilité que toutes les routes y compris celle qui passe par le nœud j échouent pour la transmission des paquets vers la destination finale D.

En conséquence, $1 - (1-TRP) (1- RP_{i,j}^d)$ désigne la probabilité qu'au moins une route réussit la transmission des paquets.

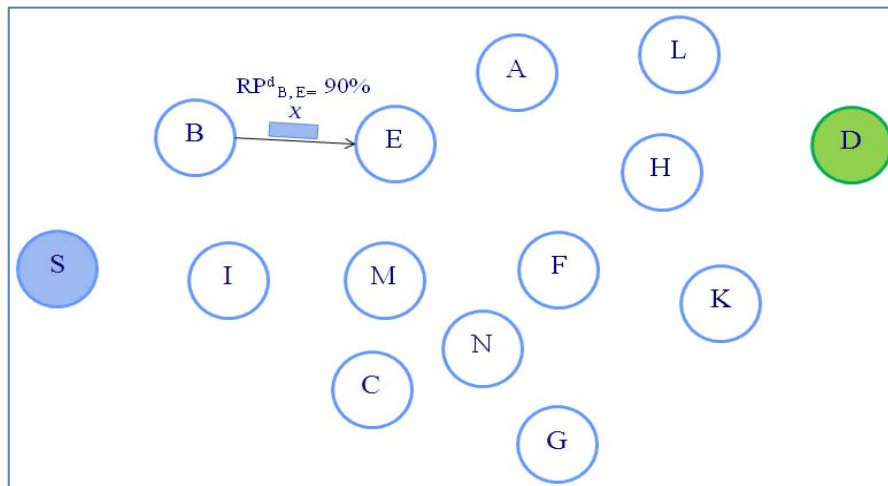
Le nœud i arrête la procédure de sélection des nœuds si la valeur de TRP atteint ou dépasse le dPDR.

Il faut noter que l'estimation des nombres de nœuds, comme prochains sauts, est spécifique pour chaque nœud intermédiaire et doit être recalculée au niveau de chaque nœud sélectionné comme le prochain saut.

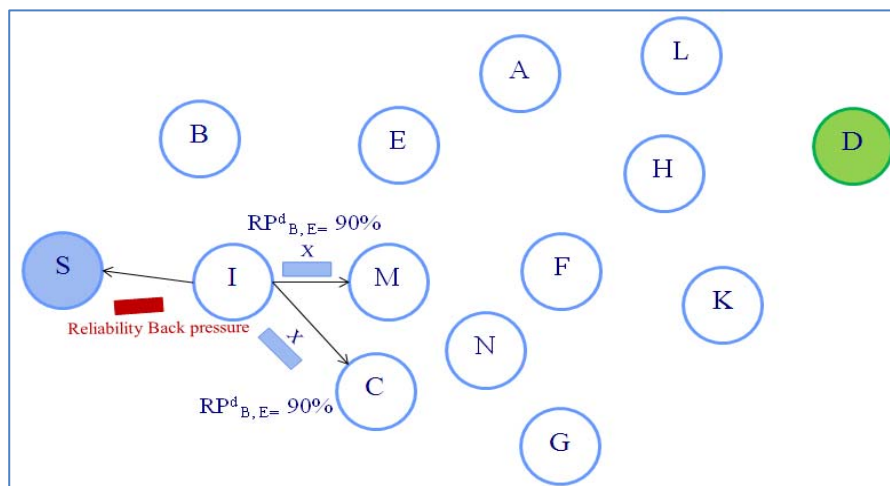
L'exemple suivant illustre bien la procédure expliquée précédemment pour le choix des prochains sauts selon la contrainte qualité de trafic:



(a)



(b)



(c)

Figure 3.11 Sélection des prochains sauts selon la contrainte qualité de trafic

Le nœud source S a détecté un événement et désire le transférer vers la destination avec une qualité de trafic égale à 80%.

Le nœud S choisit le nœud B de saut suivant, déterminé par son SNGF. Il calcule sa probabilité de livraison offerte $RP^d_{S,B}=70\%$. À ce niveau, le nœud S vérifie si le nombre des nœuds de prochain saut est suffisant pour maintenir le taux de livraison souhaité : Pour ce faire, il calcule la variable $TRP = 1 - (1 - RP^d_{S,B})$ et vérifie si cette dernière est supérieure ou égale au taux désiré. Dans notre cas, le TRP prend la valeur 70%, qui est nettement inférieur au dPDR = 80%. Le nœud S continue, alors, la sélection d'autres prochains sauts, pour partager la charge avec le nœud B et choisit le nœud offrant la deuxième plus importante probabilité de livraison à savoir le nœud I avec $RP^d_{S,I}=60\%$. le nœud S met à jour la variable

TRP qui aura comme nouvelle valeur : $TRP = 1 - (1-TRP) (1- RP_{S,I}^d) = 88\%$, qui est supérieur au dPDR = 80%. Dans le cas où le TRP égalise ou dépasse la probabilité de livraison souhaité, le nœud S arrête la sélection des prochains sauts et transmet en multicast le paquet.

Le nœud B refait les mêmes opérations que le nœud S pour déterminer les nœuds de sauts suivants, et trouve que son voisin, le nœud E, peut assurer une probabilité de livraison PR = 90%, un pourcentage largement supérieur au celui désiré. Dans ce cas, le nœud B ne va plus sélectionner d'autres prochains sauts et émet le paquet seulement vers le nœud E.

Un cas extrême peut se présenter si, par exemple, le nœud I se trouve incapable de garantir la probabilité souhaité P^{req} , même s'il transmet le paquet DATA vers tous ses voisins. Ceci est exprimé par un $TRP = 1 - (1- RP_{I,M}^d) (1- RP_{I,C}^d) < 80\%$. Le nœud I envoie, quand même, le paquet DATA en multicast vers tous ses voisins et un paquet de « *Reliability back pressure* » vers le nœud précédent S contenant comme information la valeur de TRP maximale qu'il peut supporter. À son tour, le nœud S corrige la valeur de la probabilité de livraison supporté par le nœud I en lui affectant la valeur de TRP reçue.

3.3.3. Le protocole de la couche MAC : EDCF

Pour mieux garantir la qualité de service temps réel, le protocole MMSPEED exige que la couche MAC supporte, elle aussi, la différenciation de service.

Parmi les performances requises pour cette couche, on cite les suivants :

- La priorité d'accès au médium dépend de la classe de la couche réseau (la couche réseau virtuelle).
- La livraison en multicast des paquets vers plusieurs nœuds.
- Les mesures des délais moyens pour chaque nœud voisin individuellement.
- Les mesures des taux de perte pour chaque nœud voisin individuellement.

Le protocole choisi pour répondre à ces exigences est l'EDCF [10,12] (*Enhanced Distributed Coordination Function*) proposé par le standard IEEE 802.11e. Ce dernier consiste en la gestion de huit classes de priorités (huit TC (*Traffic Category*)).

Dans l'EDCF, deux méthodes sont introduites pour supporter la différenciation de service. La première, est l'utilisation des espaces inter trames IFS (*Inter Frame Space*) différents pour les différents AC. Un nouvel type de ISF est né : AIFS (*Arbitration IFS*), et utilisé dans EDCF (au lieu du DIFS dans DCF (*Distributed Coordination Function*)). L'idée est d'affecter à ce paramètre des valeurs minimales quand il s'agit d'un flux prioritaires et contrairement quand il s'agit des flux moins prioritaires, par conséquent, les flux prioritaire auront plus de chance d'accéder au médium que les flux moins prioritaire.

La deuxième méthode introduite, comporte l'allocation de différentes tailles de fenêtre CW pour les différents AC. Assigner la plus petite taille de fenêtre CW à la file AC la plus prioritaire pour s'assurer, dans la plupart des cas, que les files de priorités hautes occupent le canal devant ceux de priorités basses.

Avec ces nouveaux paramètres, l'EDCF est supposé améliorer les performances du DCF dans des conditions de congestions.

3.4. Le protocole E3MSPEED : Enhanced Mac MMSPEED

3.4.1. Contribution :

Pour améliorer la QoS temps réel dans les réseaux de capteurs, nous proposons, dans ce projet, un nouveau protocole que nous appelons l'E3MSPEED (*Enhanced MAC MMSPEED*). Ce nouveau protocole dérive du protocole MMSPEED. La seule différence entre les deux réside au niveau de la couche MAC. En fait, nous avons remplacé le protocole EDCF (*Enhanced Distributed Coordination Function*), utilisé avec le protocole MMSPEED, par le protocole EDCA (*Enhanced Distributed Access Channel*).

En effet, le mode EDCA apporte quelques optimisations dans le classement et l'ordonnancement des paquets prêts à être transmis. Nous avons, alors, exploité ces optimisations afin de garantir une meilleure QoS temps réel dans les réseaux de capteurs. Pour se faire, nous avons gardé le protocole MMSPEED au niveau de la couche réseau, mais nous avons changé le protocole au niveau de la couche MAC par le protocole EDCA. Ce nouveau couplage est, alors, nommé « *Enhanced MAC MMSPEED* » (E3MSPEED).

Nous présentons, dans ce qui suit, le protocole EDCA et ses optimisations par rapport au protocole EDCF.

3.4.2. Le protocole de la couche MAC : EDCA

a) De l'EDCF à l'EDCA :

Le mode EDCA [10], basé sur le mode EDCF antérieur, apporte quelques optimisations dans le classement et l'ordonnancement des paquets prêtes à être transmises. Le passage entre ces deux modèles a nécessité quelques changements de la nomenclature:

- 8 catégories de trafic TCs (*Traffic Category*) deviennent 4 catégories d'accès ACs (*Access Category*). En fait, huit priorités du trafic sont employées, par le protocole EDCF, avec leurs propres files d'attente, comme indiqué dans IEEE 802.1d. Cependant, quatre files d'attente sont utilisées, par le protocole EDCA, à cause de la transformation des huit priorités de trafic à seulement quatre ACs [10].

- EDCF « Enhanced Distributed Coordination Function » devient EDCA « Enhanced Distributed Access Channel ».

b) Le protocole EDCA :

Le protocole EDCA (*Enhanced Distributed Channel Access*) contrôle l'accès au canal de transmission avec différenciation entre les flux de données et favorise mieux, que le protocole EDCF, la transmission des trames d'informations plus prioritaires que celles moins importantes en les classifiant en plusieurs catégories d'accès liées à leurs priorités respectives. En fait, EDCA fournit un accès différencié et distribué au canal d'accès. Cette technique attribut chaque trafic à une catégorie d'accès (AC) contenant des valeurs bien définies pour les paramètres d'accès DCF. L'accès au support pour une station dépend de catégorie d'accès associée au flux à transmettre.

- **Les catégories d'accès**

La norme 802.11e a défini ces quatre catégories d'accès noté AC (*Access Category*).

- AC_VO : pour les applications temps réels tel que la voix.
- AC_VI : pour les applications vidéo.
- AC_BE : pour le trafic « Best Effort ».
- AC_BK : pour le trafic Background.

Chaque AC possède son propre AIFS et maintient son propre compteur de Backoff. Ces ACs, gèrent huit niveaux de priorités (de 0 à 7) relatives à la norme 802.11D. Les correspondances entre ces priorités et les catégories d'accès sont récapitulées aussi dans le tableau 3.1.

	Priorité d'utilisateur	Désignation	Catégorie d'accès	Désignation 802.11e
↓ Priorité décroissante	1	BK	AC_BK	Background
	2	BK	AC_BK	Background
	0	BE	AC_BE	Best Effort
	3	EE	AC_BE	Vidéo
	4	CL	AC_VI	Vidéo
	5	VI	AC_VI	Vidéo
	6	VO	AC_VO	Voix
	7	NC	AC_VO	Voix

Tableau 3.1 Le passage de priorité d'utilisateur (802.11d) aux catégories d'accès (802.11e)

Il s'agit dans ce cas d'avoir quatre niveaux de priorité pour huit catégories de trafic reflétées chacune par des fenêtres de contention (minimale et maximale) et des délais inter trames différents.

Catégorie d'accès	CWmin	CWmax	AIFSN
AC_BK	a CWmin	CWmax	7
AC_BE	a CWmin	CWmax	3
AC_VI	$(a \text{ CWmin} + 1) / 2 - 1$	a CWmin	2
AC_VO	$(a \text{ CWmin} + 1) / 2 - 1$	$(a \text{ CWmin} + 1) / 2 - 1$	2

Tableau 3.2 Paramètres d'accès pour chaque AC en mode EDCA

- **Accès au support en mode EDCA**

EDCA est une solution basée sur une approche par priorité liée aux catégories de trafic. Cette solution appelée 802.11e nécessite une extension de la méthode d'accès DCF.

Les fenêtres de contention sont gérées avec des mécanismes de calcul spécifiques permettant d'avoir une croissance exponentielle de celle-ci pour chaque catégorie de trafic.

Pour assurer une certaine différenciation de service offert à différents types de trafic, on introduit la différenciation dans divers paramètres de la fonction de coordination distribuée (DCF) du standard :

- Différents facteurs d'incrémentation de backoff pour différentes priorités, PF (*Persistence Factor*).
- Différents taille minimale et maximale des fenêtres de contention, CWmin et CWmax.
- Différents espaces inter-paquets DIFS, AIFS.
- Différent temps d'occupation de support TXOPLimit.

Le délai de reprise d'une station devient ainsi une fonction de ces paramètres :

$$\text{Backoff_Timer [AC]} = f(\text{AIF [AC]}, \text{CWmin [AC]}, \text{CWmax [AC]}, \text{PF [AC]})$$

avec :

- AIF [AC] est le délai inter-trame de la catégorie d'accès (AC). CWmin [AC] est la fenêtre minimale de AC.
- CWmax [AC] est la fenêtre maximale de l'AC.
- PF [AC] est un facteur de persistance utilisé pour réduire encore la probabilité de collision de TC.
- TXOPLimit [AC] (Transmission Opportunity) période pendant laquelle la station a droit d'envoyer ces données.

La figure 3.12 illustre l'emploi de 4 fonctions d'accès au support. Chaque fonction est associée à une file d'attente qui comporte une catégorie d'accès. Les valeurs d'accès, pour chaque catégorie d'accès, adoptés dans notre simulation sont indiqués dans le tableau 3.3.

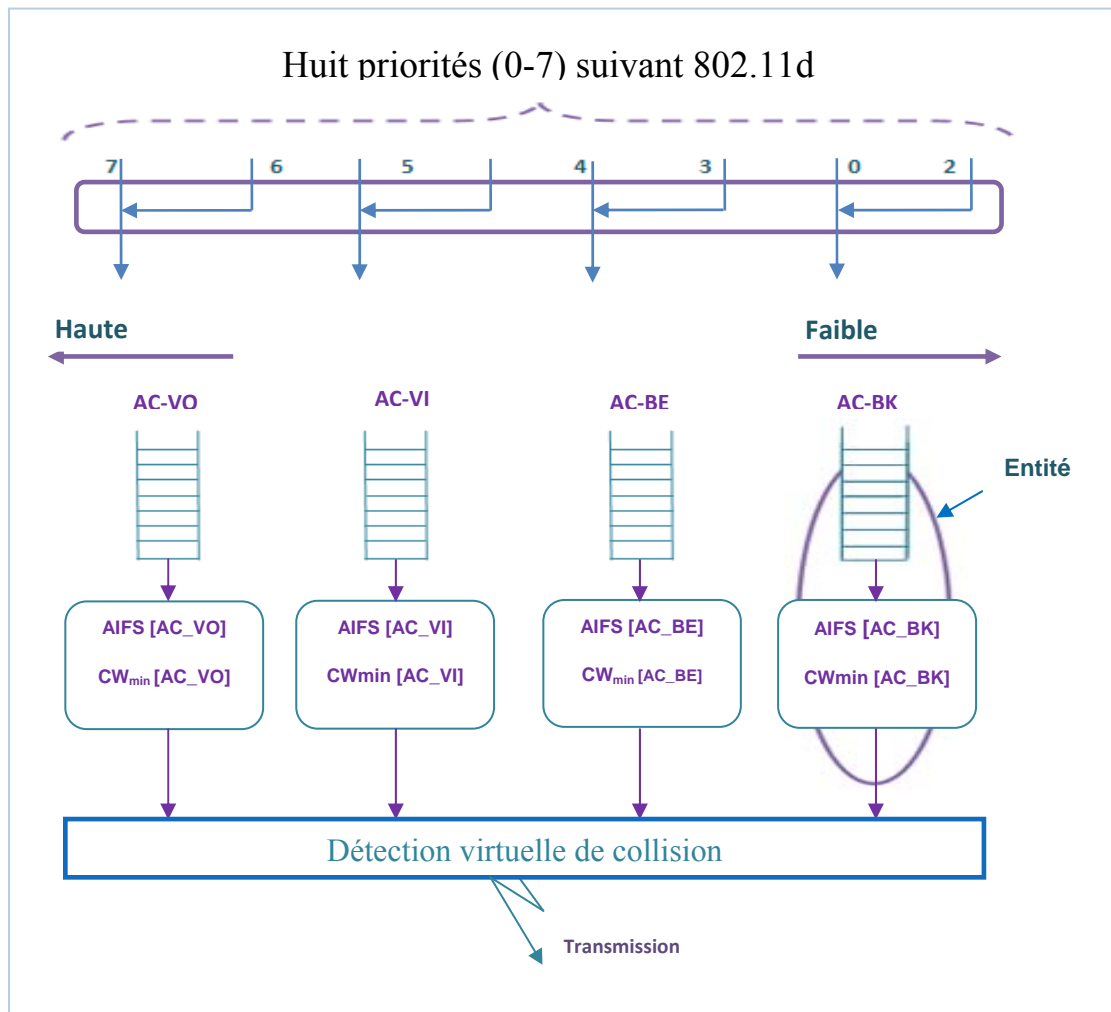


Figure 3.12 Différentiation de services au niveau de la couche MAC EDCA

	AC_VO [0]	AC_VI [1]	AC_BE [2]	AC_BK [3]
AIFSN	2	2	3	7
CW_{min}	7	15	31	31
CW_{max}	15	31	1023	1023
TXOPLimit	5.00 E-5	3.00 E-5	3.00 E-5	3.00 E-5
PF	1	2	2	2

Tableau 3.3 Valeur d'accès d'après le *Draft IEEE 9.0*

Les plus petites valeurs de CWs et AIFSN sont attribuées à la catégorie d'accès la plus prioritaire. Ainsi plus le trafic à transmettre est prioritaire plus son accès est rapide.

- **Mécanisme d'accès EDCA**

Rappelons que le mode DCF, est basé sur le protocole d'accès CSMA/CA imposant aux stations souhaitant communiquer de vérifier d'abord que le canal radio est inoccupé, puis de respecter un petit laps de temps (Backoff + IFS (ce sont deux temporisateurs) avant de débiter la transmission pour éviter les collisions et les erreurs de transmission.

Avec le mode EDCA, l'IFS devient l'AIFS (Arbitration IFS), et est d'autant plus court que le trafic correspondant est classé dans une catégorie de plus grande priorité, ainsi lorsque plusieurs demandes d'accès se sont en même temps, celle appartenant à la catégorie de plus haute priorité passe devant, c'est ce qu'on appelle le TxOPLimit? (Transmission Opportunities), notons que même pour la plus haute priorité le temps de backoff reste invariant pour empêcher les collisions.

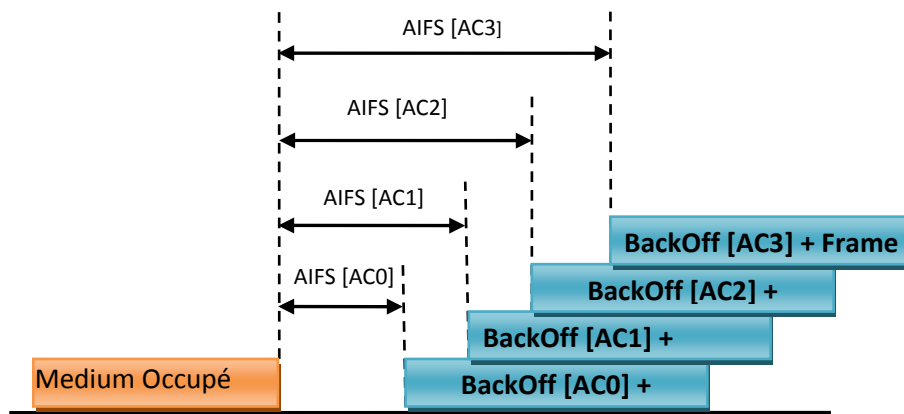


Figure 3.13 Accès EDCA

Chaque AC correspond à une file de sortie et chaque file de sortie tente d'émettre pendant une opportunité de transmission (TXOPLimit) en respectant les nouvelles conditions inspirées de DCF :

- La priorité entre les classes est créée artificiellement en accordant un IFS propre à chacune des files appelé arbitration IFS (AIFS) qui retardera plus au moins l'accès au canal: $AIFS = DIFS + n * Time_slot$
- Les valeurs de la fenêtre de contention CW_{min} et CW_{max} à partir desquelles sont calculées les valeurs de compteur de backoff propre à chaque file de sortie.
- Au sein d'une même station les files de priorité inférieure laisse la place à celle de priorité supérieure.

- Lorsque deux files d'attente entre en collision, c'est celle de priorité supérieure qui se voit attribuée une opportunité de transmission (TXOPLimit). On simule alors une collision virtuelle pour la queue de priorité inférieure qui doit ré exécuter un Backoff.
- Le facteur de persistance (PF) permet d'augmenter la taille de la fenêtre de contention (CW) après chaque collision selon l'équation suivante :

$$CW_{min}[i] = ((CW_{old}[i] + 1) * PF - 1)$$

Quand PF est grand, la fenêtre de contention est grande en moyenne, par suite il a moins de chance d'accéder au canal, et son débit utile est réduit.

- **Mécanisme CFB**

Un autre paramètre a été introduit par la norme dans sa proposition actuelle : ce sont Les intervalles TXOPLimit (Transmission Opportunity) qui représentent l'intervalle de temps pendant lequel une station a le droit d'émettre. Durant un EDCA TXOPLimit, une station a droit d'émettre plusieurs paquets à partir d'une seule AC tout en utilisant l'inter-trame SIFS entre l'ACK et son paquet correspondant transmis. Cette transmission est notée CFB (contention-free burst).

3.5. Conclusion :

Ce chapitre a fait le sujet d'une étude théorique des protocoles de routage avec QoS à savoir SPEED et MMSPEED. En effet, le protocole SPEED se caractérise essentiellement par la garantie des délais de bout de bout réduits. Cet aspect lui permet de répondre aux exigences des applications critiques et d'offrir une QoS temps réel satisfaisante. Cependant, avec l'existence de trafic avec différents niveaux de priorités, SPEED se trouve incapable de les servir différemment, ce qui dégrade, de plus en plus, la qualité de service de flux prioritaire. Le protocole MMSPEED a permis de résoudre cette problématique en adoptant une différenciation de service en termes de délai de bout en bout. MMSPEED a permis, aussi, de garantir les taux de PDR exigés par le trafic, et ce, grâce au mécanisme de partage de charge qu'il emploie si le cas le nécessite.

Pour maintenir sa fiabilité, le protocole MMSPEED exige l'interopérabilité avec un protocole de QoS au niveau de la couche MAC, à savoir l'EDCF. Mais, un renouvellement de ce dernier a été développé, donnant naissance au protocole EDCA. Nous avons, alors, remplacé l'ancien protocole EDCF par le nouveau EDCA au niveau de la couche MAC tout en gardant le protocole MMSPEED au niveau de la couche réseau. Ceci constitue le principe de base de notre nouveau protocole E3MSPEED.

Chapitre 4

Évaluation des performances des protocoles : SPEED, MMSPEED et E3MSPEED

L'objectif de cette partie est l'évaluation et l'étude comparative des performances de chacun des protocoles discutés auparavant, à savoir : SPEED, MMSPEED et E3MSPEED. Les simulations de ces derniers ont été réalisées par le simulateur JSIM [13].

4.1. Paramétrage de la simulation:

Avant de lancer les simulations, nous devons ajuster certains paramètres qui constitueront le contexte de notre simulation. Ces paramètres sont présentés par le tableau 4.1 :

Paramètres du contexte de la simulation	
Nombre des nœuds	50
Portée des nœuds	40m
Dimension du réseau	200m * 200m
Vitesse d'émission	10 paquet/s
Taille de paquet de données	32 octets
Bande passante	20 kbps
Paramètres de la couche MAC : EDCF	
Classe des trafics (haute et faible priorité)	2
SIFS	10 μ s
Time slot	20 μ s
Persistent Factor (PF)	2
Trafic prioritaire	Trafic moins prioritaire
AIFS 2	AIFS 5
$CW_{min} = 15, CW_{max} = 255$	$CW_{min} = 256, CW_{max} = 1024$
Environnement de simulation	
Machine	Pentium IV, 3.06GHz, 1.5 Go
Système d'exploitation	Windows XP
JSIM	Version 1.3

Tableau 4.1 Paramètres du contexte de la simulation

4.2. Méthodologie de la simulation :

Nous comparons, en premier temps, les deux protocoles SPEED et MMSPEED en fonction de la contrainte délai de bout en bout. Ensuite une comparaison basée sur la contrainte de qualité de trafic est mise en avant. Enfin, une combinaison de ces deux contraintes est exploitée afin d'interpréter le comportement de chaque protocole. En deuxième étape, nous évaluons et nous comparons les performances du nouveau protocole proposé, *Enhanced MAC MMSPEED* : E3MSPEED, par rapport au protocole MMSPEED en nous intéressant à la garantie de la QoS temps réel dans les réseaux de capteurs.

4.3. SPEED et MMSPEED :

a) Étude comparative basée sur le délai de bout en bout :

Pour mettre en évidence les performances des protocoles SPEED et MMSPEED en différenciation de service selon la contrainte délai de bout en bout, nous avons divisé le trafic en deux groupes : le premier groupe est désigné pour le trafic prioritaire, ayant des exigences strictes en termes de délai de bout en bout, égale à 0.3 seconde. Le deuxième groupe représente le trafic moins prioritaire dont la contrainte délai de bout en bout égale à 1s.

Dans notre contexte de simulation, la couche réseau, exécutant le protocole MMSPEED, est configurée en deux couches réseaux virtuelles. La première couche offre une vitesse de livraison importante égale à 1000 m/s, pour répondre aux exigences strictes des trafics. La deuxième couche virtuelle transmet les paquets selon une vitesse moins importante égale à 250 m/s. Concernant le protocole uni-vitesse SPEED, les paquets des flux prioritaires et moins prioritaires seront délivrés avec la même vitesse de transmission maximale égale à 1000m/s afin de réaliser des délais de bout en bout les plus réduits possibles.

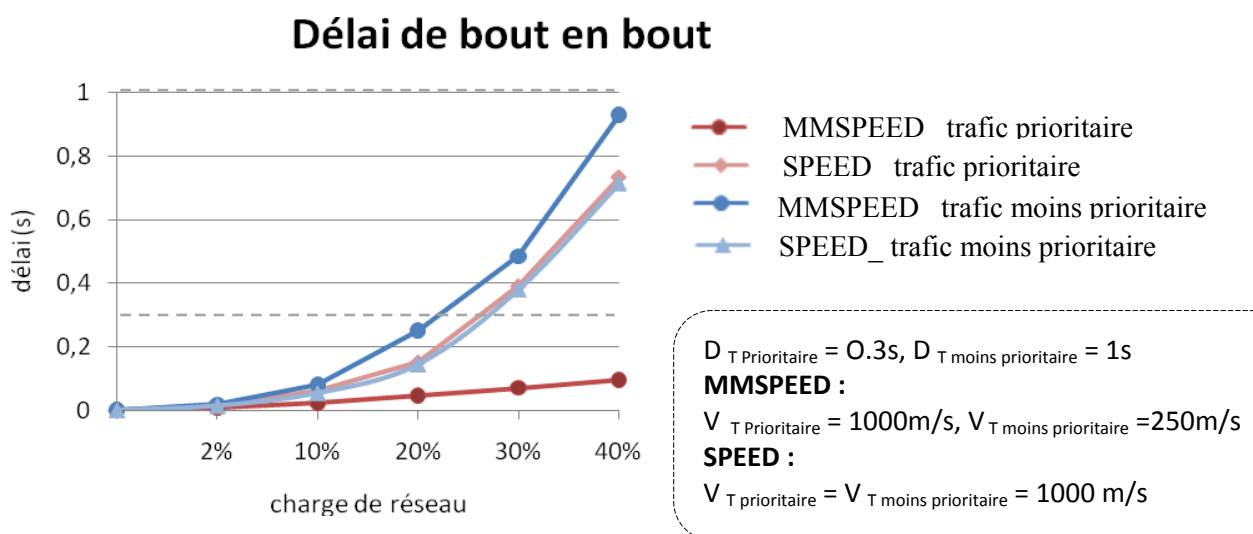


Figure 4.1 Délai de bout en bout : SPEED et MMSPEED

Dans la figure 4.1, les trafics prioritaires et moins prioritaires, servis par le protocole MMSPEED, sont présentés respectivement par les couleurs rouge et bleu. Les couleurs rouge clair et bleu clair désignent respectivement le trafic prioritaires et moins prioritaires servis par le protocole SPEED.

La figure 4.1 illustre l'influence de la charge de réseau sur la durée du délai de bout en bout. On entend dire par la charge de réseau le taux des nœuds sources des trafics par rapport au nombre totale des nœuds capteurs. Ces nœuds émettent de trafic avec des niveaux de priorités adéquats aux événements à avertir.

Cette figure décrit bien la différenciation de service en termes de délai de bout en bout offerte par le protocole MMSPEED et montre l'incapacité du protocole SPEED à l'assurer :

En effet, le protocole MMSPEED est montré capable de respecter les exigences temporelles des deux groupes de trafics et de transmettre les paquets dans des délais plus optimaux que ceux exigés jusqu'à une charge de réseau égale à 40%. Pour une telle charge, les paquets prioritaires ont été transmis au cours d'une durée égale au tiers du délai exigé et la livraison des paquets moins prioritaires n'a pas dépassé le délai exigé, égal à 1 seconde. Cette limitation aux délais exigés met en valeur le mécanisme de mise à jour du délai restant exploité par le protocole MMSPEED (comme décrit dans le paragraphe 3.3.1.b). La différenciation de service offerte par le protocole MMSPEED est envisagé par l'écart réalisé entre les délais de bout en bout des deux groupes de trafics et qui s'élève jusqu'à 0.85 seconde pour cette même charge du réseau. C'est grâce aux mécanismes de classification des trafics au niveau de la couche réseau (comme décrit dans le paragraphe 3.3.1.a,) et la livraison des paquets suivant des vitesses de transmission conformes à leurs priorités que le protocole MMSPEED a réussi la différenciation de service selon la contrainte délai de bout en bout.

Concernant le protocole SPEED, celui ci se trouve désavantagé par rapport au protocole MMSPEED. En fait, la figure 4.1 montre deux inconvénients de ce protocole. La première est son incapacité de se limiter aux exigences temporelles des trafics prioritaire. C'est que SPEED ne peut plus livrer les paquets prioritaires dans le délai désiré dans un réseau chargé plus que 25%. Son deuxième inconvénient est le traitement similaire des deux types de trafics, prioritaires et moins prioritaires. Malgré les diverses exigences, les délais de bout en bout réalisés pour les deux groupes de trafics sont très proches et parfois confondus.

En conclusion, le protocole MMSPEED se comporte mieux que le protocole SPEED dans la livraison des paquets, surtout les plus prioritaires. Raisonant suivant le pourcentage de la charge de réseau la plus importante, le protocole MMSPEED réduit le délai de livraison des trafics urgents d'un écart égale à 0,6 seconde par rapport au protocole SPEED. Pour les trafics

moins urgents, SPEED s'avère plus optimale. C'est tout à fait attendu puisque la vitesse de transmission des paquets moins prioritaire utilisée par le protocole SPEED est plus importante que celle utilisée par MMSPEED.

b) Qualité de trafic :

Dans cette partie, nous montrons les performances des deux protocoles SPEED et MMSPEED, en différenciation de service selon la contrainte qualité de trafic. Pour cela, nous avons divisé les trafics en deux groupes : le premier groupe avec une contrainte qualité de trafic stricte, exprimé par le paramètre *desired PDR* (*dPDR*) qui est égale à 75%. Cette valeur veut dire qu'on désire avoir un taux de livraison des paquets minimale égale à 75%. Le deuxième groupe est désigné pour les trafics non urgents, avec la contrainte qualité de trafic, *dPDR*, égale à 25%. Nous avons conservé les mêmes vitesses de transmission, pour les deux protocoles, comme sont détaillées dans le paragraphe 4.3.a.

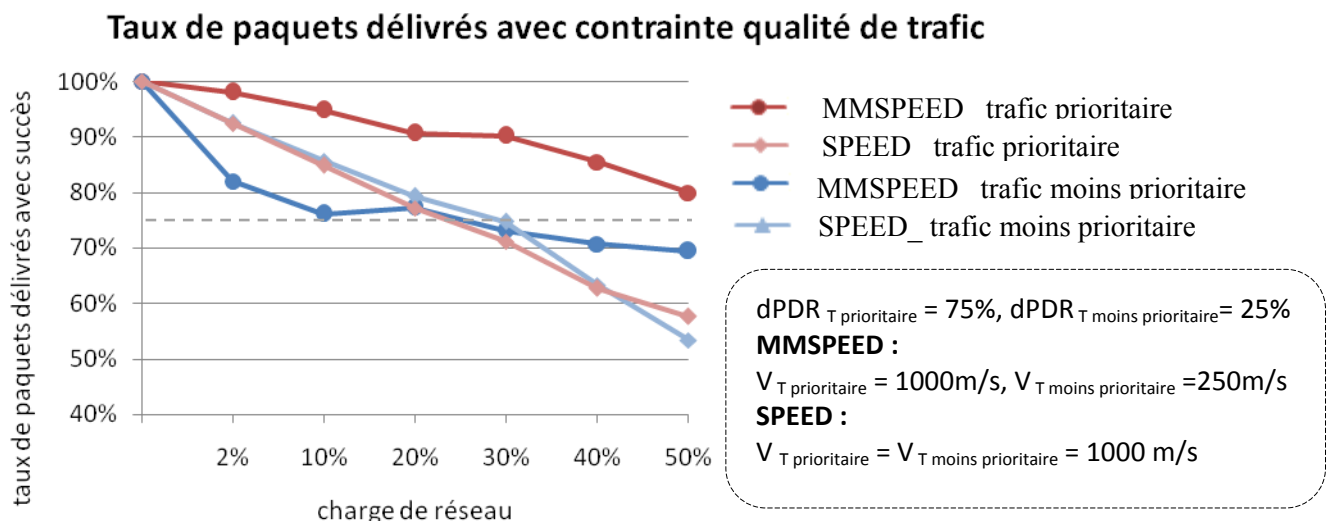


Figure 4.2 Taux de livraison des paquets avec contrainte qualité de trafic : SPEED (1000 m/s) et MMSPEED

Nous adoptons les mêmes que dans la figure 4.1 pour illustrer les trafics prioritaires et moins prioritaires servis par les protocoles SPEED et MMSPEED.

La figure 4.2, montre que le taux de livraison des paquets avec succès diminue avec l'augmentation de la charge du réseau. Malgré cette décroissance, MMSPEED a pu maintenir le taux de livraison désiré pour chaque groupe de trafic dans un réseau chargé jusqu'à 50%. Ceci met en évidence le mécanisme de partage de charge, activé par le protocole MMSPEED, dans le cas où le nœud cible pour le prochain saut se trouve incapable d'assurer le taux PDR désiré (comme décrit dans le paragraphe 3.3.2). L'écart réalisé entre les PDRs des deux groupes de trafics varie entre 10% et 20%. Ceci fait preuve de la réussite de ce protocole en la

différentiation de service selon la contrainte qualité de trafic. Cependant, le protocole SPEED se montre, encore une fois, incapable de servir les deux groupes de trafic selon leurs contraintes de qualité de trafic. En effet, SPEED réalise presque les mêmes taux de livraison pour les deux groupes de trafic malgré leurs différentes exigences, ce qui prouve son faiblesse dans la différenciation de service. Aussi, SPEED ne peut plus maintenir le taux de livraison exigé par les trafics prioritaires si la charge de réseau dépasse 20%.

En conclusion, le protocole MMSPEED est plus performant que le protocole SPEED en assurant la contrainte qualité de trafic. En raisonnant suivant la charge de réseau la plus importante, MMSPEED augmente le taux de livraison d'un pourcentage égale à 20% pour les trafics prioritaires et d'un pourcentage de 15% pour les trafics moins prioritaires, et ce, par rapport au protocole SPEED.

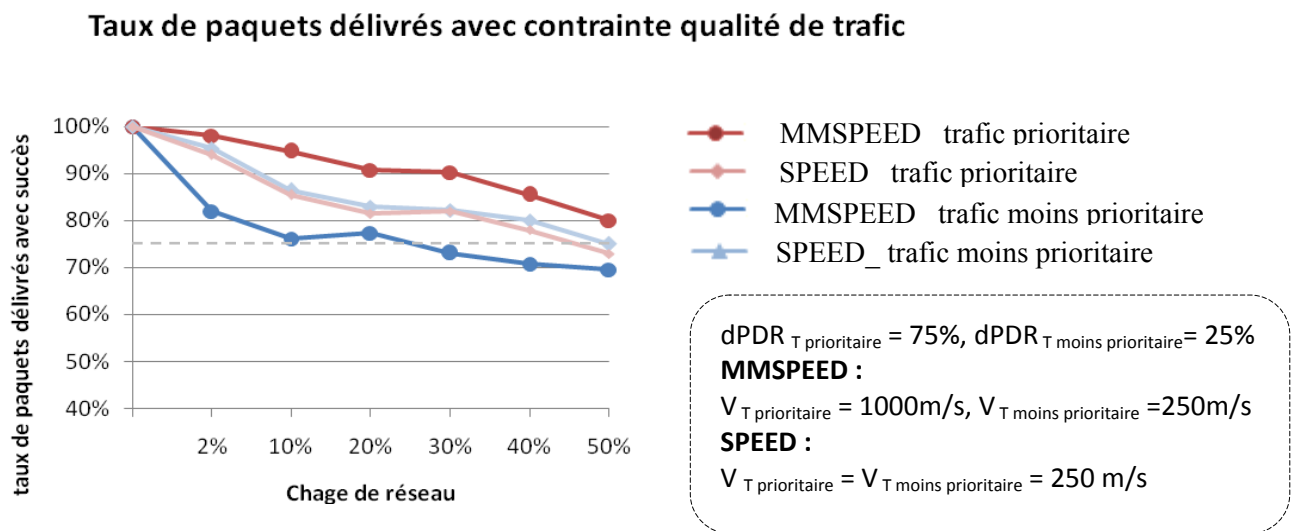


Figure 4.3 Taux de livraison des paquets avec contrainte qualité de trafic : SPEED (250m/s) et MMSPEED

Dans l'expérience illustrée par la figure 4.3, nous avons choisi de délivrer le trafic, servi par le protocole SPEED, suivant une vitesse de transmission plus faible, égale à 250 m/s. Nous constatons que, en délivrant les paquets suivant une faible vitesse de transmission, le taux PDR, réalisé par le protocole SPEED, a bien amélioré. C'est parce que la probabilité de trouver des nœuds de prochains sauts, offrant une vitesse de transmission égale à 250 m/s est plus importante que de trouver des nœuds de prochains sauts avec une vitesse égale à 1000 m/s, ce qui minimise le taux de perte des paquets.

En délivrant les paquets suivant une vitesse pareille, le protocole SPEED essaye de se rapprocher, en termes de taux de livraison des paquets (PDR), au protocole MMSPEED. En effet, pour une charge de réseau égale à 50%, l'écart dans la livraison de trafic prioritaire s'élève presque à 7% pour le trafic prioritaire et moins prioritaire, alors qu'il a atteint 20% en

délivrant les paquets suivant la vitesse 1000 m/s, comme illustré par la figure 4.2. Mais malgré ça, le protocole SPEED n'arrive pas à concurrencer le protocole MMSPEED en la garantie des taux PDR répondant à différentes exigences de trafics.

c) Contrainte combinée : délai de bout en bout et qualité de trafic

Pour étudier les performances des protocoles SPEED et MMSPEED en fonction des deux contraintes de délai de bout en bout et qualité de trafic, nous avons réalisé une troisième expérience qui différencie les flux de données en quatre groupes :

- G1 : avec un délai de bout en bout court égal à 0.3s et une haute qualité de trafic égale à 0.75.
- G2 : avec un large délai de bout en bout égal à 0.3s et haute qualité de trafic égale à 0.25.
- G3 : avec un délai de bout en bout court égal à 1s et une faible qualité de trafic égale à 0.75.
- G4 : avec un large délai de bout en bout égal à 1s et une faible qualité de trafic égale à 0.25.

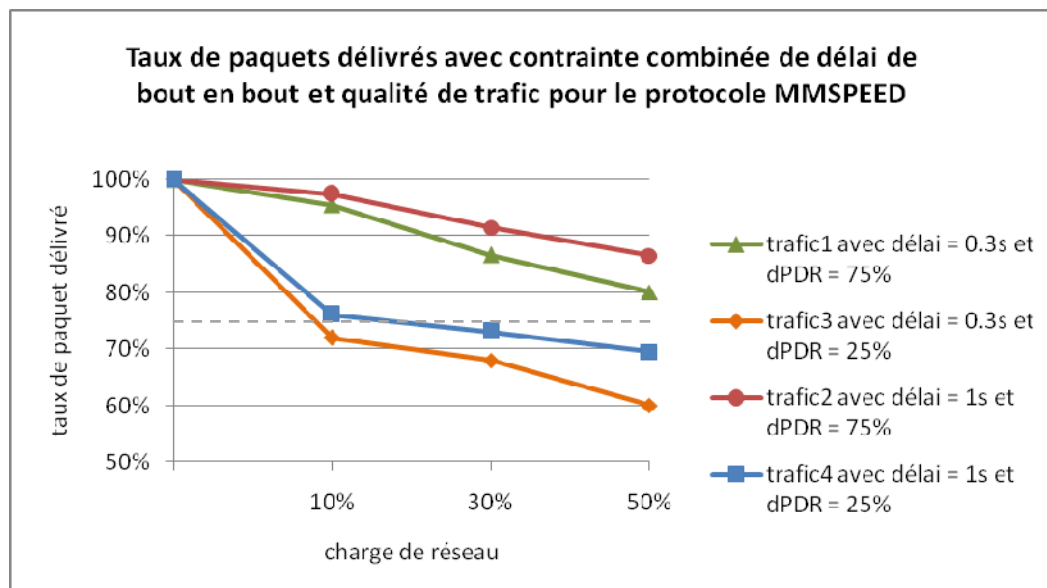


Figure 4.4 Taux de livraison des paquets avec contraintes délai de bout en bout et qualité de trafic pour le protocole MMSPEED

La figure 4.4 présente les taux des paquets délivrés avec succès et dont leurs durées de transmission n'ont pas dépassé les délais exigés pour les quatre groupes de trafics. Ces taux chutent progressivement avec la surcharge du réseau. Malgré leurs décroissances, chaque groupe de trafic garde un taux de livraison supérieur au seuil et vérifie la qualité de trafic exigé. Ceci prouve la fiabilité du protocole MMSPEED en la différenciation de service et la garantie de la QoS en termes de délai de bout en bout et qualité de trafic et met en évidence tous ses mécanismes.

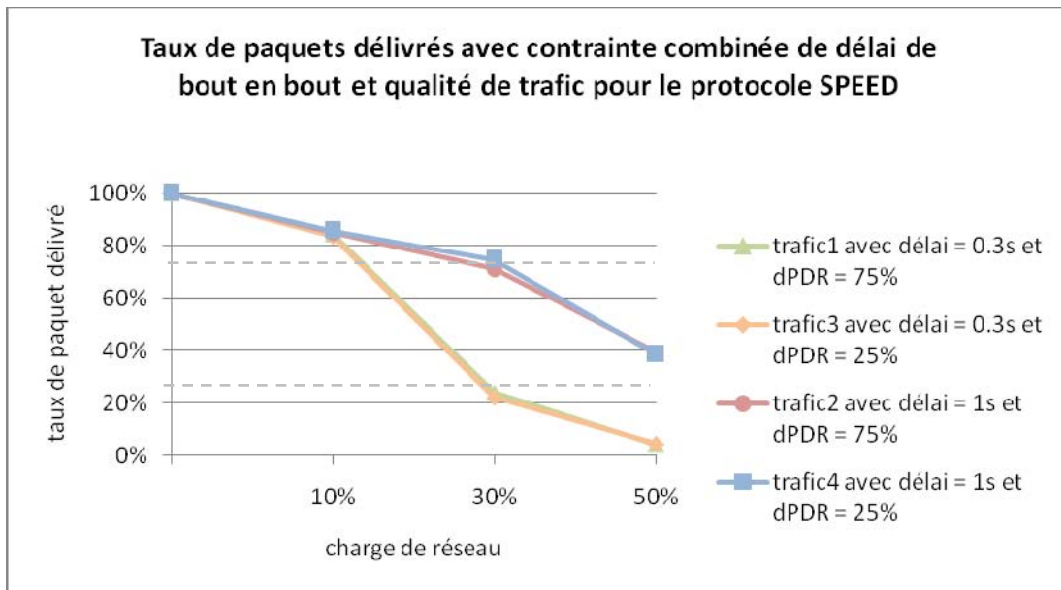


Figure 4.5 Taux de livraison des paquets avec contraintes délai de bout en bout et qualité de trafic pour le protocole SPEED

Nous avons réalisé la même expérience pour juger les performances du protocole SPEED par rapport à celles du protocole MMSPEED. En fait, SPEED est limité en la garantie de la QoS en termes la contrainte combinée de délai de bout en bout et qualité de trafic pour les trafics urgents, ayant un délai de livraison stricte égale à 0.3 seconde. Les taux de livraison pour ces trafics urgents, présentés par les couleurs vert et orangé, commencent à se dégrader si la charge de réseau dépasse le 11% et réalisent un taux égal à 5% dans un réseau chargé en moitié, alors que MMSPEED assure un taux supérieur à 80% pour la même charge. Pour le trafic non urgent, exigeant un délai de bout en bout égale à 1 seconde et un PDR élevé égale à 75%, le protocole SPEED ne peut respecter ses contraintes que dans un réseau chargé jusqu'à 30%, alors que le protocole MMSPEED réalise un taux égal à 85% pour une charge de réseau de 50%. le seule groupe de trafic que le protocole SPEED a pu vérifier ses contraintes, pour une charge de réseau de 50%, est le trafic le moins prioritaire ayant un délai de bout en bout large égal à 1 seconde et un PDR réduit égal à 25%. Mais le taux des paquets délivrés avec succès, et au cours du délai exigé, reste aussi inférieur à celui réalisé par le protocole MMSPEED. En fait, SPEED marque un taux égal à 40% pour une charge de réseau égale à 50%, alors que MMSPEED réalise un taux égal à 70% pour cette même charge.

La grande différence entre les deux protocoles en la réalisation des taux de livraison des paquets dans les délais exigés met en valeur deux mécanismes employés par MMSPEED : le premier est le mécanisme de partage de charge qui permet de vérifier la contrainte de qualité

de trafic et l'envoi des paquets de « *Reliability Back pressure* » qui permet de corriger les estimations des taux de livraison des nœuds voisins (comme décrit dans le paragraphe 3.3.2). Le deuxième mécanisme est la mise à jour du délai restant qui permet de calculer le temps restant pour compléter la livraison des paquets (paragraphe 3.3.1.b).

d) Paquet de contrôle :

Les paquets de contrôle, mis en jeu, incluent les paquets de localisation, émis périodiquement pour mettre à jour la topologie du réseau de capteurs, les paquets de « *Delay back pressure* », émis si le nœud capteur ne peut pas contribuer à l'assurance du délai de bout en bout exigé, enfin, les paquets de « *Reliability back pressure* », émis si tous les nœuds de prochain saut ne peuvent pas assurer la qualité de trafic exigée. On doit noter que les deux premiers types de paquet de contrôle sont employés par les deux protocoles SPEED et MMSPEED, alors que le dernier type de paquet de contrôle n'est déployé que par le protocole MMSPEED.

Dans cette partie, nous nous intéressons à la génération des paquets de contrôle pendant la transmission des paquets du trafic prioritaire et moins prioritaire. Pour le cas du trafic prioritaire, celui-ci exige des contraintes de délai de bout en bout et qualité de trafic strictes et égales respectivement aux 0.3 seconde et 75%. Cependant, le trafic moins prioritaire exige des contraintes moins strictes égales à une durée de 1 seconde pour le délai de bout en bout et un pourcentage de 25% des paquets délivrés avec succès.

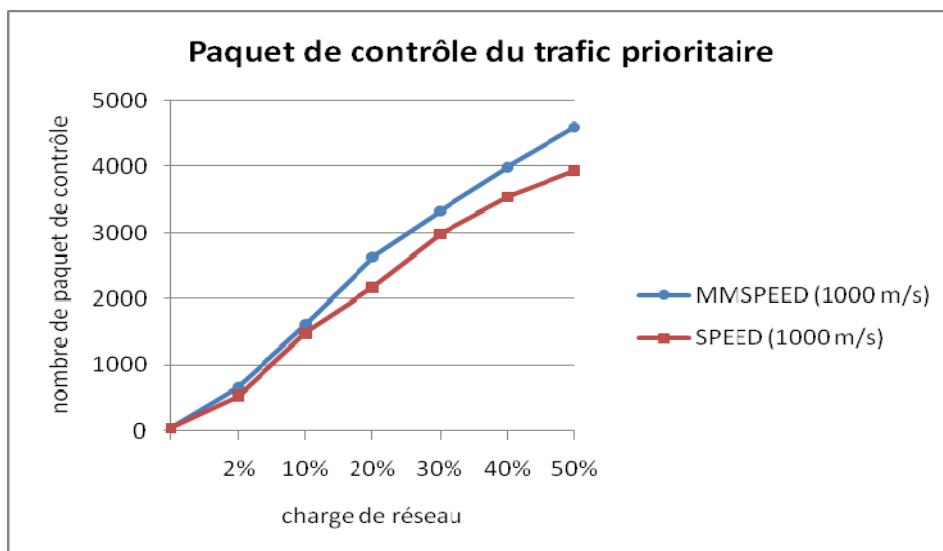


Figure 4.6 Paquet de contrôle du trafic prioritaire : SPEED et MMSPEED

La figure 4.6 montre l'évolution des paquets de contrôle, générés pendant la livraison du trafic prioritaire par chacun des protocoles SPEED et MMSPEED, en fonction de la charge de réseau. Sous les mêmes contraintes du trafic prioritaire, le protocole MMSPEED génère plus de paquet de contrôle que le protocole SPEED, et ce, dans un réseau chargé à partir de 10%.

l'écart entre les paquets de contrôle émis par les deux protocoles est approximé par un taux presque égal à 11%. Ceci peut être expliqué par les facteurs suivants :

- Les deux protocoles, SPEED et MMSPEED, génèrent le même nombre des paquets de localisation puisqu'ils emploient le même mécanisme « *Neighbor beacon exchange* » (comme décrit au paragraphe 3.2.1).
- Concernant les paquets de contrôle « *Delay back pressure* », les deux protocoles en génèrent aussi le même taux puisqu'il s'agit de la livraison du trafic prioritaire avec les mêmes contraintes. En fait, les protocoles ont la même probabilité de trouver des nœuds de prochain sauts offrant la vitesse de transmission exigée. Ceci fait que les deux protocoles procèdent par l'émission des mêmes taux de paquets « *Delay back pressure* » s'ils ne réussissent pas à trouver les nœuds de prochain sauts adéquats (comme décrit au paragraphe 3.2.4).
- En plus de ces deux paquets de contrôle, seulement le protocole MMSPEED génère des paquets « *Reliability back pressure* » si tous les nœuds de prochains sauts se trouvent incapables de garantir le taux PDR exigé. Ce cas commence à se produire dans un réseau chargé de plus que 10%.

En conclusion, l'écart envisagé dans la génération des paquets de contrôle, entre les deux protocoles SPEED et MMSPEED, met en évidence le troisième type de paquets de contrôle à savoir « *Reliability back pressure* ».

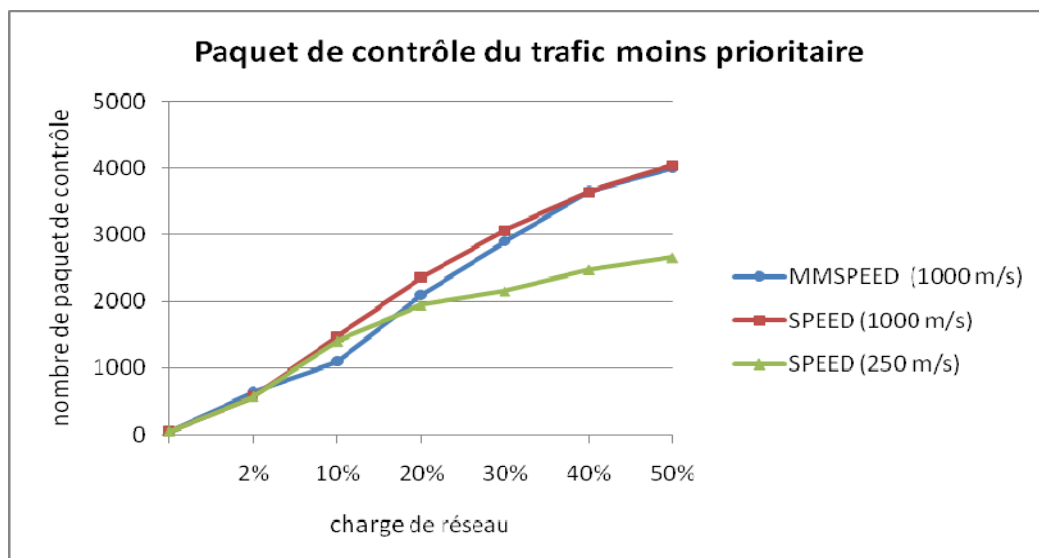


Figure 4.7 Paquet de contrôle du trafic moins prioritaire

la figure 4.7 illustre l'augmentation des paquets de contrôle, générés pendant la livraison des paquets moins prioritaire, en fonction de la charge de réseau.

En premier lieu, nous avons simulé le protocole SPEED avec une vitesse égale à 1000 m/s pour le favoriser par rapport au protocole MMSPEED qui gère mieux les trafics. Nous avons constaté que, pour les faibles charges de réseau, le nombre des paquets de contrôle généré par le protocole SPEED est inférieur à celui généré par le protocole MMSPEED. L'écart réalisé entre les deux protocoles atteint presque 30% pour une charge de réseau égale à 10% et commence à s'annuler à partir d'une charge de réseau allant du 40% au 50%.

En effet, cet écart met plus en cause la génération des paquets « *Delay back pressure* » : Le taux de ces paquets, généré par le protocole MMSPEED, est moins important que celui généré par le protocole SPEED. C'est parce que ce dernier essaye de délivrer les paquets de flux moins prioritaires suivant une vitesse de transmission importante, égale à 1000 m/s, alors que, le protocole MMSPEED offre une vitesse plus faible pour ce type de trafic qui est égale à 250 m/s. Dans ce cas, le protocole SPEED est moins favorisé que le protocole MMSPEED à trouver des nœuds de prochain sauts offrant la vitesse de transmission exigée. Par conséquent, il génère plus de paquet de « *Delay back pressure* » (comme décrit dans le paragraphe 3.2.4). Ceci met plus en évidence le caractère multi vitesse du protocole MMSPEED qui lui permet la différenciation de service en livrant les paquets prioritaires et moins prioritaires suivant des vitesses de transmission différentes.

Pour des pourcentages de charge de réseau importants, allant du 40% au 50%, les deux protocoles émettent presque le même taux des paquets de contrôle. Pour des tels charges, la probabilité de trouver des nœuds de prochains sauts offrant la vitesse de transmission exigée est de plus en plus faible.

Nous avons refait la même expérience, mais en changeant la vitesse de transmission utilisé par le protocole SPEED. Nous avons, alors, choisi de transmettre les paquets suivant une vitesse plus faible, égale à 250 m/s. Ce changement a provoqué une importante diminution dans les paquets de contrôle générés par le protocole SPEED. Comme l'indique la figure 4.7, le protocole SPEED réalise une diminution de trafic *overhead* généré, allant du 16% pour une charge de réseau égale à 20%, au 30% pour une charge de réseau égale à 50%. Cette diminution est due au fait que la probabilité de trouver des nœuds de prochain sauts, offrant une vitesse de transmission égale à 250 m/s est plus importante que de trouver des nœuds de prochains sauts avec une vitesse égale à 1000 m/s. Ceci diminue la génération des paquets de « *Delay back pressure* ».

4.4. MMSPEED et E3MSPEED:

Dans ce paragraphe, nous focalisons notre étude à montrer l'apport de notre contribution, à savoir le protocole E3MSPEED, par rapport au protocole MMSPEED. Puisque nous avons montré que ce dernier est plus performant que le protocole SPEED.

Nous présentons, tout d'abord, les paramètres de la couche MAC opérant suivant le protocole EDCA dans le tableau suivant :

Paramètres de la couche MAC : EDCA	
Classe des trafics (haute et faible priorité)	2
SIFS	10 μ s
Time slot	20 μ s
Trafic prioritaire	
AIFS	2
$CW_{min} = 7,$ $CW_{max} = 15$	
PF	1
Trafic moins prioritaire	
AIFS	7
$CW_{min} = 31,$ $CW_{max} = 1023$	
PF	2

Tableau 4.2 Paramètres du protocole EDCA

a) Délai de bout en bout :

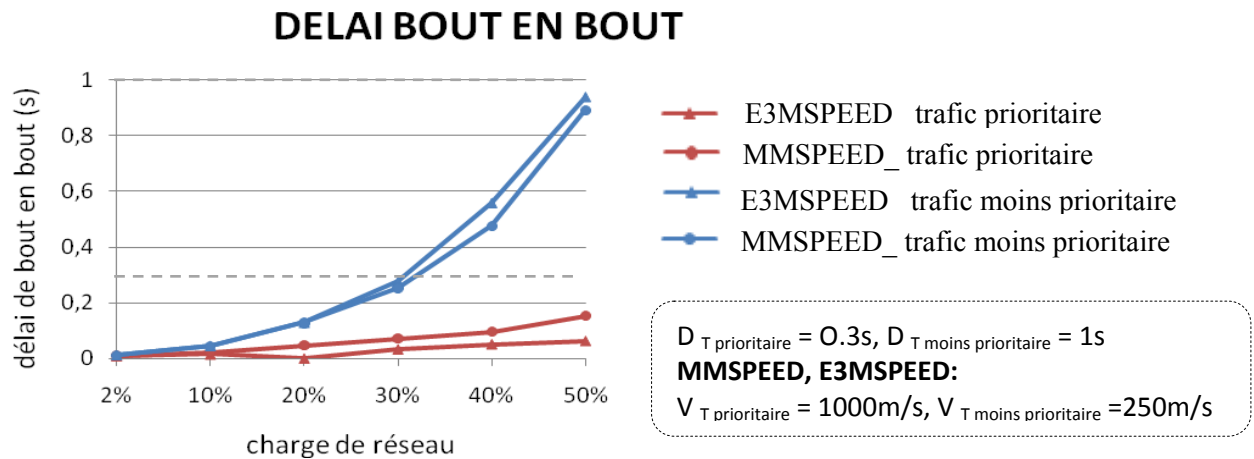


Figure 4.8 Délai de bout en bout : MMSPEED et E3MSPEED

Le couplage des deux protocoles basés sur la garantie de la qualité de service : MMSPEED au niveau de la couche réseau et EDCA au niveau de la couche MAC, a bien amélioré les performances temps réel du réseau des capteurs. Comme l'indique la figure 4.7, le nouveau protocole E3MSPEED a réalisé des délais de bout en bout plus optimaux pour les deux groupes de trafics tout en respectant les contraintes exigées. Concernant le trafic prioritaire, la contribution du nouveau protocole est clairement envisageable à partir d'une charge de réseau

de 15%. Dans un réseau chargé en moitié, E3MSPEED a optimisé le délai de livraison de bout en bout d'une marge égale à 0,1 seconde. Pour le trafic moins prioritaire, le nouveau protocole a aussi optimisé ces délais d'une marge variante entre 0.05 et 0.1 seconde à partir d'une charge de réseau égale à 30%. Le protocole E3MSPEED a réussi à différencier les services selon les exigences de chaque trafic. Cette différence de service est envisagé par l'écart réalisé entre les délais de livraison des deux types de trafic et qui s'élève jusqu'à 0.9 seconde. Cette expérience met en preuve l'apport du protocole EDCA par rapport au protocole EDCF en la garantie de la QoS selon la contrainte délai de bout en bout.

b) Contrainte combinée : délai de bout en bout et qualité de trafic

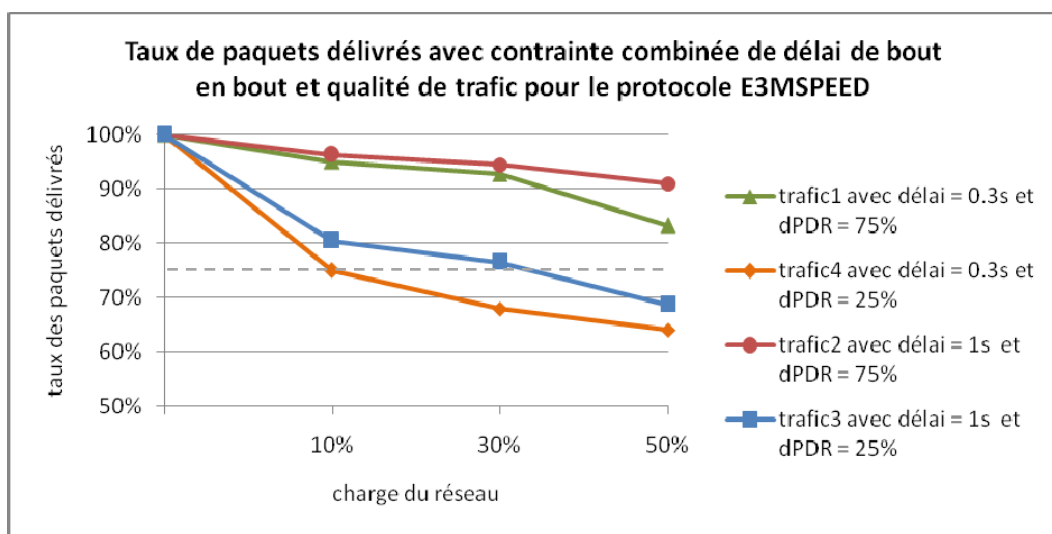


Figure 4.9 Taux de livraison des paquets avec contraintes délai de bout en bout et qualité de trafic pour le protocole E3MSPEED

Nous reprenons le même scénario de simulation, expliqué dans le paragraphe 4.3.c pour justifier les performances du protocole E3MSPEED en termes de livraison des paquets selon les contraintes délai de bout en bout et qualité de trafic.

La figure 4.8 prouve les améliorations ajoutées par le protocole E3MSPEED : En comparant cette figure avec la figure 4.3, les taux de livraison des paquets pour les quatre groupes de trafics ont augmenté par rapport à ceux réalisés par le protocole MMSPEED et sont toujours au dessus du seuil. Pour les trafics prioritaires, présentés par les couleurs verte et orange, E3MSPEED a réalisé une légère augmentation entre 2% et 5% du taux des paquets délivrés sous les deux contraintes. Les taux des trafics moins prioritaires, présentés par les couleurs rouge et bleue, se sont améliorés d'un pourcentage variant entre 4% et 7%.

Cette amélioration est due à l'optimisation des délais de livraison des paquets suite à la contribution du protocole EDCA au niveau de la couche MAC (comme montré par la figure

4.7). C'est que l'optimisation des délais de transmission des paquets permet l'augmentation des taux de paquets délivrés pendant une durée égale au délai exigé. Puisque ces délais de livraison ont été légèrement améliorés, les taux de livraison ont aussi légèrement augmenté.

c) **Paquet de contrôle :**

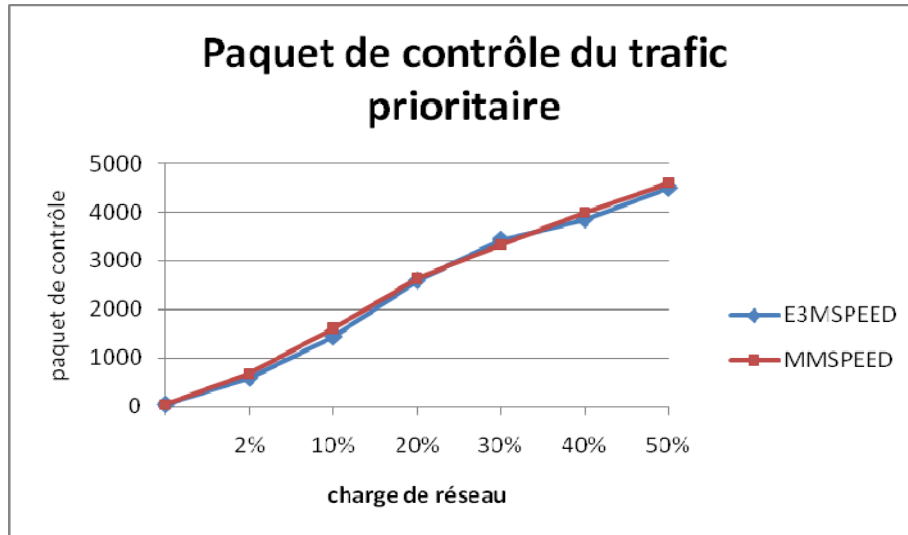


Figure 4.10 Paquet de contrôle du trafic prioritaire : MMSPEED et E3MSPEED

Nous reprenons le même scénario de simulation, expliqué dans le paragraphe 4.3.d, pour justifier les performances du protocole E3MSPEED en termes de génération de trafic *overhead*. Les paquets *overhead* mis en jeu contiennent les paquets de localisation, les paquets de « *Delay back pressure* » et les paquets de « *Reliability back pressure* ».

Le figure 4.10 montre l'évolution des paquets de contrôle générés pendant la livraison du trafic prioritaire, par chacun des protocoles MMSPEED et E3MSPEED, en fonction de la charge de réseau. Sous les mêmes contraintes du trafic, le nouveau protocole E3MSPEED génère presque le même taux des paquets de contrôle que le protocole MMSPEED. Pour une charge de réseau égale à 50%, les paquets de contrôle générés par les deux protocoles et pendant la livraison des deux types de trafic, atteignent des taux importants et presque égaux. Ceci est dû au fait que, pour une telle charge, la probabilité de trouver des nœuds de prochain sauts répondant aux exigences du trafic prioritaire est de moins en moins faible. Ce qui produit plus de paquets de « *Delay back pressure* ».

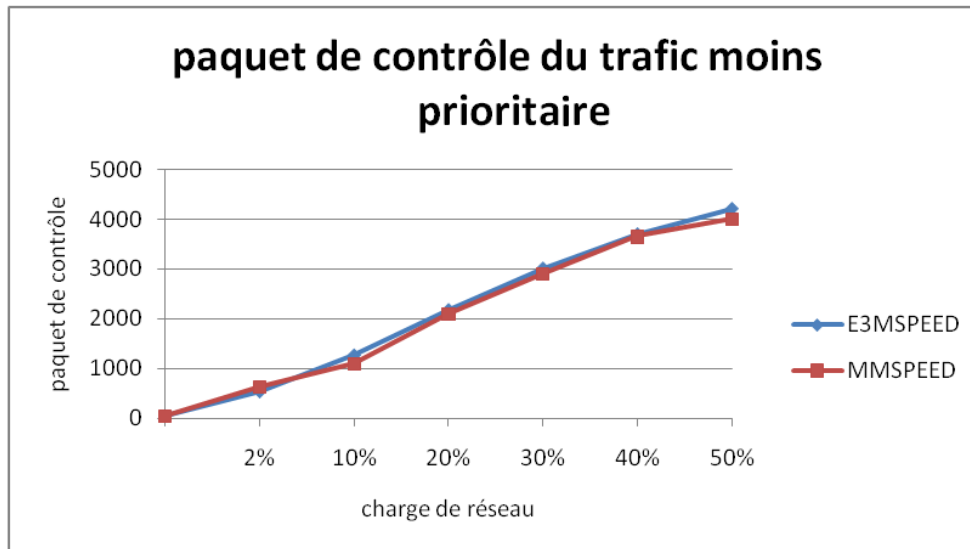


Figure 4.11 Paquet de contrôle du trafic prioritaire : MMSPEED et E3MSPEED

La figure 4.11 décrit la génération du trafic de contrôle pendant la transmission du trafic moins prioritaire par les deux protocoles MMSPEED et E3MSPEED. En effet, ces deux protocoles en génèrent presque les mêmes taux. Ces derniers atteignent de taux important dans des réseaux de plus en plus chargés, mais restent légèrement inférieur aux taux réalisés pendant la transmission du trafic prioritaire (figure 4.10). C'est parce que les contraintes de ce type de trafic sont plus sévères, ce qui rend plus difficile leurs garanties et, donc, engendre plus de paquets de contrôle de « Delay back pressure ».

En conclusion, le nouveau protocole E3MSPEED est d'autant coûteux que le protocole MMSPEED en termes de génération de trafic de contrôle au cours de la livraison du trafic prioritaire.

4.5. Conclusion :

Ce chapitre a fait le sujet d'une étude comparative de deux protocoles de routage basée sur la qualité de service temps réel à savoir SPEED et MMSPEED, et d'une évaluation du nouveau protocole proposé, l'E3MSPEED.

Les simulations présentées montrent bien que MMSPEED est plus fiable que SPEED de point de vue garantie de la QoS temps réel. En effet, le protocole MMSPEED s'avère capable de respecter la contrainte de délai de bout en bout exigée par les différents types de trafic présentant différents niveaux de priorités. Alors que le protocole SPEED n'est pas capable d'offrir une telle différenciation de service, basée sur le délai de bout bout. Aussi, MMSPEED a réussi à garantir les taux de PDR exigés, et ce, grâce au mécanisme de partage de charge qu'il exploite si le besoin l'exige. Le protocole MMSPEED est un peu plus gourmand en trafic *overhead* par rapport au protocole SPEED, mais, ça ne peut pas dégrader ses

performances vu que l'écart en trafic *overhead* entre les deux est faible. Concernant notre protocole E3MSPEED, il a assuré plus de performances que MMSPEED en terme de garantie de QoS temps réel au sein des réseaux de capteurs. E3MSPEED a mieux optimisé le délai de bout en bout et a assuré une amélioration des taux de paquets délivrés avec succès et dont leur livraison n'a pas dépassé le délai de bout en bout exigé.

Conclusion générale

Les réseaux de capteurs présentent une solution adéquate pour servir les applications critiques telles que les alertes de catastrophes naturelles, la détection des intrusions, etc. Pour répondre aux exigences temps réel de ce type d'applications, il a fallu développer de nouveaux protocoles de routage qui garantissent la QoS temps réel et qui tiennent compte de certaines contraintes spécifiques de ces réseaux, comme la densité importante des nœuds capteurs, leurs mobilités, leurs autonomies énergétiques limitées, leurs durées de vie limitées...

C'est dans cette optique que nous avons mené nos travaux dans ce projet. En effet, nous nous sommes intéressés aux protocoles de routage basés sur la QoS pour les réseaux de capteurs à savoir les protocoles SPEED et MMSPEED. En nous basant sur une étude comparative de ces deux protocoles, nous avons proposé un nouveau protocole de routage, appelé E3MSPEED (*Enhanced Mac MMSPEED*). Ce qui constitue notre principale contribution. Le protocole E3MSPEED permet d'optimiser d'une manière plus efficace que MMSPEED et SPEED, les performances temps réel des réseaux de capteurs.

C'est que le protocole SPEED se caractérise essentiellement par la garantie des délais de bout de bout réduits, mais, non conformes aux délais exigés, surtout quand il s'agit de livrer des flux de données de priorités différentes. Cette incompétence a mené au développement d'un deuxième protocole MMSPEED. Ce dernier dérive du protocole SPEED et favorise la différenciation de service en terme de délai de bout en bout en délivrant les paquets de trafic suivant une vitesse de transmission adéquate à leurs priorités. MMSPEED contribue aussi à la garantie des taux de PDR importants pour le trafic exigeant, et ce, en adoptant un routage multi-chemins si le besoin persiste. Pour maintenir sa fiabilité, le protocole MMSPEED exige l'interopérabilité avec un protocole de QoS au niveau de la couche MAC. Le protocole choisi avec le MMSPEED est l'EDCF. Mais, un renouvellement de ce dernier a été développé, donnant naissance au protocole EDCA. Nous avons, alors, exploité ce nouveau protocole pour améliorer la QoS temps réel dans les réseaux de capteurs. En effet, nous avons remplacé l'ancien protocole EDCF par le nouveau EDCA au niveau de la couche MAC tout en gardant le protocole MMSPEED au niveau de la couche réseau. Ceci fait l'idée de notre nouveau protocole E3MSPEED. Les simulations effectuées, dans le cadre de notre projet, ont confirmé, d'une part, la fiabilité du protocole MMSPEED par rapport au protocole SPEED en fonction des deux métriques de délai de bout en bout et des taux PDR délivrés avec succès.

D'autre part, ils ont mis en évidence l'apport du protocole E3MSPEED en fonction de ces mêmes contraintes. En conclusion, le nouveau protocole, E3MSPEED, s'avère plus performant en la garantie de la QoS selon les contraintes délai de bout en bout et qualité de trafic.

Notre travail peut être amélioré par d'autres travaux de prolongation. Une première perspective consistera à l'étude de performances et de jugement de la QoS temps réel offerte par les protocoles SPEED, MMSPEED et E3MSPEED au sein des réseaux de capteurs mobiles. Une deuxième perspective visera l'optimisation de la consommation d'énergie par ces protocoles. En fait, les protocoles de routage basés sur la QoS sont un peu gourmand en énergie ce qui contredit l'autonomie énergétique limitée des réseaux de capteurs.

Bibliographie

- [1] CHEE-YEE CHONG AND SRIKANTA P. KUMAR, “**Sensor Networks: Evolution, Opportunities, and Challenges**”, 2003.
- [2] I.F. Akyildiz, W. Su*, Y. Sankarasubramaniam, E. Cayirci, “**Wireless sensor networks: a survey**”, Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA, Received 12 December 2001; accepted 20 December 2001.
- [3] Archana Bharathidasan, Vijay Anand Sai Ponduru, “**Sensor Networks: An Overview**”, Department of Computer Science University of California.
- [4] Laura Raileanu et Fatemi Nastaran, « **Les Réseaux de Senseurs** », Haute école d’ingénierie et de gestion du Canton de Vaud, 10/01/2006.
- [5] Dazhi Chen and Pramod K. Varshney, “**QoS Support in Wireless Sensor Networks: A Survey**”, Syracuse University, U.S.A.
- [6] Jamal N. Al-Karaki Ahmed E. Kamal, “**Routing Techniques in Wireless Sensor Networks: A Survey**”, Dept. of Electrical and Computer Engineering Iowa State University.
- [7] M. Erwan ERMEL, « **Localisation et Routage géographique dans les réseaux sans fil hétérogènes** », Université Pierre et Marie CURIE, 21 Juin 2004.
- [8] Tian Hea John A Stankovica Chenyang Lub Tarek Abdelzahera, “**Speed: A Stateless Protocol for Real-Time Communication in Sensor Networks**”, University of Virginia Washington University in St Louis.
- [9] J. F. Kurose, K. W. Ross. “**Computer Networking A Top-Down Approach Featuring the internet**”, ISBN 0-201-47711-4 Addison Wesley Longman Inc.
- [10] REBAI Ahmed Riadh, « **Optimisation Globale de la Qualité de Service dans les réseaux locaux sans fils de type IEEE 802.11 : Mise en Œuvre de nouveaux mécanismes de gestions de QoS pour la couche d’accès** », Décembre 2005, Mars 2007.

- [11] Emad Felemban, Chang-Gun Lee and Eylem Ekici, “**MMSPEED: Multipath Multi-SPEED Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks**”, juin 2006.
- [12] Lassaâd Gannoune, Stephan Robert, Neha Tomar and Tarun Agarwal, “**Results on Dynamic Adaptation of the Contention Window Max (CWmax) for Enhanced Service Differentiation in IEEE 802.11 Wireless Ad-Hoc Networks**”, Ecole d’Ingénieurs du Canton de Vaud (EIVD), Indian Institute of Technology (IIT).
- [13] <http://www.j-sim.org/>

Annexe

1. Présentation du simulateur JSIM (*Java SIMulator*) :

Afin de juger le simulateur JSIM en son évaluation des performances des réseaux de capteurs, nous synthétisons les avis des membres du projet JSIM [13]:

Le simulateur JSIM fonctionne avec la plateforme Java et est une licence gratuite. Il est utilisé à l'INT et permet de simuler des réseaux de l'ordre de 1000 nœuds. Le passage à l'échelle peut toutefois être amélioré. JSIM souffre peut être de sa jeunesse et quelques corrections s'avèrent nécessaires.

Ce simulateur utilise quasi-indifféremment deux langages : Java et OTCL. L'architecture et le code sont suffisamment bien structurés pour permettre une prise en main relativement aisée.

Concernant l'analyse des résultats, il est plus aisé que pour le cas du simulateur NS2. JSIM permet d'utiliser, comme générateur de trafic, n'importe quelle application Java et semble gérer correctement l'aspect consommation d'énergie.

2. Installation du simulateur JSIM sous la plateforme WINDOWS

2.1. Préparation d'environnement :

Etape1 :

Pour installer le simulateur JSIM, il faut déjà avoir installé sur votre PC un logiciel spécial depuis le site web de la société Sun Microsystems, son nom complet est *Java 2 Software Development Kit (J2SDK)*. Pour des raisons de stabilité, il est hautement recommandé d'installer la version *J2SDK 1.4* ou bien des versions ultérieures. La version dont j'ai utilisé pendant mon projet de PFE est *J2SDK 5.0* et peut être téléchargée depuis le site suivant : <http://java.sun.com/j2se>

Etape2 :

La compilation des codes sources pour le simulateur JSIM peut se faire soit via le compilateur *Apache Ant*, soit par la commande « *make* ».

Le premier compilateur est valable depuis le lien : <http://ant.apache.org/>

Pour compiler via la commande « *make* », on doit installer l'utilitaire *GNU-Make makefiles* in Java. Cet utilitaire permet d'installer des commandes propres aux plateformes LINUX sous WINDOWS.

Etape3 :

La version du simulateur JSIM la plus utilisée est J-SIM version1.3. Elle peut être téléchargée depuis le lien : http://www.j-sim.org/cgi-bin/j-sim_downloader.cgi

Il faut noter que la version J-Sim version1.3 n'est pas compatible avec la version **J2SDK 5.0**. Pour des questions d'interopérabilités, on doit installer le patch4 ou bien des versions de patch ultérieures. Le patch dont j'ai installé est *patch4-version1.3* valable sur: <http://www.j-sim.org/patch.html>

2.2. Installation de JSIM :**Etape4 :**

L'installation du JSIM se fait sur plusieurs sous étapes:

On décompresse sous la racine C:\ le package téléchargé J-Sim_v1.3.tar. On obtient par la suite un dossier nommé jsim-1.3.

Rem : la décompression des fichiers .tar sous WINDOWS se fait avec le logiciel WINRAR.

On modifie le fichier setcpath.bat placé sous C:\jsim-1.3 comme suit :

On affecte à la variable **%J_SIM%** le chemin d'installation du simulateur JSIM

On affecte à la variable **% JAVA_HOME %** le chemin d'installation de **J2SDK5.0**

On initialise une variable **% ANT_HOME %** au chemin d'installation du compilateur *Apache Ant* et on ajoute à la variable **%PATH%** le chemin vers les commandes de compilation si on le choisit comme compilateur pour JSIM.

Si on a choisit de compiler les codes sources du simulateur JSIM avec la commande « *make* », on doit donc ajouter le chemin d'installation de à la variable **%PATH%**.

On ajoute à la variable **%CLASSPATH%** le chemin vers les classes du simulateur JSIM.

Si le compilateur choisi pour le simulateur JSIM, est *Apache Ant*, le fichier *setcpath.bat* doit comprendre les données suivantes :

```
@echo off
set JAVA_HOME=C:\jdk
set J_SIM=C:\jsim-1.3
set ANT_HOME=C:\ant
set PATH=%PATH%;c:\jdk\bin;c:\ant\bin
set CLASSPATH=.;c:\jsim-1.3\classes;c:\jsim-1.3\jars\tcl.zip;c:\jsim-
1.3\jars\jython.jar
```

Si la compilation pour le simulateur JSIM est suivant la commande « *make* », le fichier `setcpth.bat` doit être de la forme suivante :

```
@echo off
set JAVA_HOME=C:\jdk
set J_SIM=C:\jsim-1.3
set PATH=%PATH%;c:\jdk\bin;c:\UnxUtils_1\usr\local\wbin
set CLASSPATH=.;c:\jsim-1.3\classes;c:\jsim-1.3\jars\tcl.zip;c:\jsim-
1.3\jars\jython.jar
```

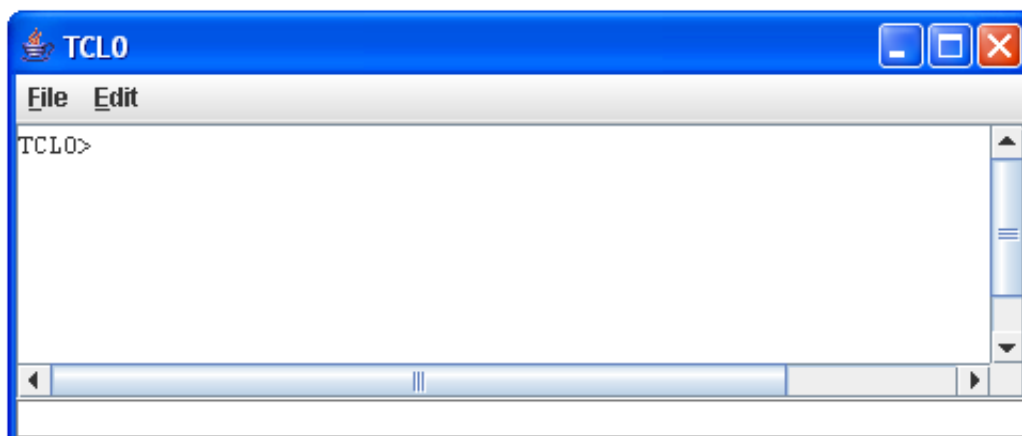
1. On ouvre maintenant la fenêtre des invites des commandes DOS, et on se place sous le répertoire `jsim-1.3`. pour cela, on tape les commandes suivantes :

```
cd C:\
cd jsim-1.3
```

2. On exécute maintenant le fichier `setcpth.bat` en l'appelant sous la fenêtre d'invite des commandes.
3. On compile tous les codes sources du simulateur JSIM en tapant la commande « *make* » si vous avez installé l'utilitaire *GNU-Make makefiles in Java* ou la commande « *ant compile* » si vous avez installé le compilateur *Apache Ant*.
4. On décompresse maintenant le dossier ***patch1.3-4.tar.gz***, déjà téléchargé, sous le répertoire `jsim-1.3` et on accepte les mises à jour des fichiers existants.
5. On recompile de nouveau les codes sources et on ignore tous les avertissements.

Etape5 :

Pour vérifier si l'installation de JSIM a réussi, vous devez avoir une fenêtre pop up intitulé `TCL0` qui apparait en tapant la commande suivante : `java drcl.ruv.System`



Etape 6 :

Chaque fois que vous désirez simuler un script TCL, vous devez changer le chemin courant vers où est placé votre script, puis vous tapez la commande suivante :

```
java drcl.ruv.System « nom de votre script.tcl »
```

3. Implémentation du protocole MMSPEED

Une implémentation de protocole MMSPEED, fonctionnant avec le simulateur JSIM, est disponible depuis le lien : <http://www.ece.osu.edu/~felembe/guide.htm>. Il suffit de télécharger le dossier nommé *mmspeed.zip*, fournie dans le lien indiqué, le décompresser puis le placer sous le chemin `$JSIM/src/drcl/inet/protocol/` avec `$JSIM` : le chemin d'installation du simulateur JSIM. Après, on exécute la commande « *make* » depuis le même chemin pour compiler les scripts JAVA et TCL contenu dans ce dossier *mmspeed*.

4. Implémentation du protocole SPEED

Jusqu'à la date de l'écriture de ce rapport, nous n'avons pas trouvé des scripts propres au protocole SPEED. Pour pouvoir le simuler, nous avons exploité sa relation de primitive avec le protocole MMSPEED. En fait, nous avons modifié les scripts JAVA du protocole MMSPEED pour qu'il se comporte comme le protocole SPEED. Pour cela, il a fallu bien étudier les différences entre les deux.

5. Technique de simulation

La simulation des protocoles de routage des réseaux de capteurs connaît quatre phases interdépendantes à savoir : pré-simulation, simulation, post-simulation et exploitation.

5.1. Pré-simulation

Avant toute simulation, nous devons préciser les paramètres du scénario à simuler. Pour ce faire, nous devons initier les variables nécessaires au niveau du script TCL. Ce dernier est placé sous le chemin `$JSIM/src/drcl/inet/protocol/mmspeed/testMMSPEED.TCL`

```
## Terrain Size 200x200                                # Sink Node
                                                         set target 0

# Terrain Width X axis
set maxX 20.0                                           # the random seed
set minX 0                                              set seed 3
                                                         # Mobility of the Nodes
# Terrain Hight Y axis
set maxY 20.0                                           set Node_Speed 0
set minY 0
                                                         ##### Start MMSPEED Layer Settings
set NumberofNodes 50
                                                         # Number of SPEED Classes
set sourcenode 39                                       set levels 2
```

```

# SetSpeed for each class          ##### Simulation Settings
set speed_0 1000                  ! n*/app setHighFlowRP .75
set speed_1 250                   ! n*/app setLowFlowRP .25
##### End MMSPEED Layer Settings ! n*/app setHighFlowDeadline .3
                                   ! n*/app setLowFlowDeadline 1
                                   ! n*/app setDest [getXAddress 0 ]
set queue_size 1000              [ getYAddress 0 ]
set bandwidth 200000              #####

```

Une fois, tous les paramètres sont correctement initiés, nous lançons la simulation en exécutant la commande suivante sous la fenêtre DOS :

```
java drcl.ruv.System testMMSPEED.tcl
```

5.2. Simulation

Durant cette phase, JSIM simule les différents scénarios pendant une durée bien fixée. Une redirection des résultats de ces simulations est possible et aussi recommandé pour les filtrer d'avantage. En fait, JSIM permet la création d'autant de fichier trace que vous désirez et vous permet aussi de diriger le résultat, que vous sélectionnez, vers le fichier trace choisie.

Tout d'abord, nous devons créer un port de redirection au niveau du fichier .java qui fournit le résultat que nous cherchons. À travers ce port, nous redirigeons les résultats vers le fichier trace déjà créé. Nous expliquons plus cette procédure suivant un exemple décrivant la procédure de calcul de délai de bout en bout au niveau du nœud « Sink ».

Au niveau du script .java :

- Création du port de redirection des résultats:

```
public static final String port1_TRACE_PORT_ID = ".port1";
protected Port port1Port = addPort(port1_TRACE_PORT_ID);
```

- Calcul de délai de bout en bout:

```
double now = getTime();
double delay = (now-timestamp); // timestamp est le temps d'émission du
                                paquet par le nœud source
double deadline = packet.getDeadline();
```

- Redirection des résultats vers le port « port1 »:

```
port1Port.doSending ("got Packet from: "+ packet.getSrcID()+" sent at:"
+ timestamp + " with Deadline:" + deadline + " and priority:" +
packet.getTOS()+" Packet Delay:" + delay);
```

Au niveau du script TCL:

- Création du fichier trace et redirection des résultats :

```
set filename_ SINK.result
set file_ [mkdir drcl.comp.io.FileComponent .file]
$file_ open $filename_
connect -c n0/app/.trace@ -to $fileAlll_/in@ }
```

Après la simulation, nous aurons le fichier trace nommé SINK.result créé sous le répertoire courante **\$JSIM/src/drcl/inet/protocol/mmspeed**. Ce fichier contient les informations suivantes:

```

Packet from: 45 sent at: 11.671067555003985 Packet Delay: 0.00588631773399
Packet from: 44 sent at: 11.66401604612051 Packet Delay: 0.024788201416804
Packet from: 40 sent at: 11.68300510902085 Packet Delay: 0.017476166101827
Packet from: 41 sent at: 11.69472127512268 Packet Delay: 0.011625649461622
Packet from: 49 sent at: 11.71844192359964 Packet Delay: 0.005863013873804
Packet from: 44 sent at: 11.66401604612051 Packet Delay: 0.066045479242413
Packet from: 46 sent at: 11.748025718633897 Packet Delay: 0.00585063816520
Packet from: 43 sent at: 11.77730060711087 Packet Delay: 0.005813173050981
Packet from: 48 sent at: 11.82386918943804 Packet Delay: 0.005942585717310
Packet from: 42 sent at: 11.861393526272428 Packet Delay: 0.00805246939662
Packet from: 42 sent at: 11.926292743629757 Packet Delay: 0.01186267147113
...
...

```

5.3. Post-simulation:

Dans cette phase, nous allons récupérer les fichiers trace, que nous avons créé, et nous allons extraire les résultats que nous voulions visualiser ou interpréter. Cette extraction ainsi que toute autre opération de calcul est assurée par plusieurs scripts en langage AWK¹.

Nous présentons la commande utilisée pour calculer le délai de bout en bout moyen, exécutée sous la fenêtre DOS :

```

gawk "{commande à executer}" fichier trace >> fichier_filtré
gawk "{ delai += $9 } END { print delai }" Sink.result >> delai.awk

```

5.4. Exploitation:

Une fois la phase de post-simulation est achevée, nous pouvons exploiter les fichier .awk obtenus pour tracer les courbes ou bien effectuer d'autres calculs moyennant d'autres logiciels (MatLab, Excel, GnuPlot, etc).

¹ Aho, Weinberger, & Kernighan.