

République algérienne démocratique et populaire
Ministère de l'enseignement Supérieure et de la recherche scientifique
Université des sciences et de la technologie Houari Boumediene
Faculté d'électronique et d'informatique, Institut d'informatique

Thèse magister en Informatique

Sujet :

Optimisation de la consommation
d'énergie pour le routage dans les
réseaux mobiles ad hoc

Thème proposé par :
Dr. BADACHE Nadjib

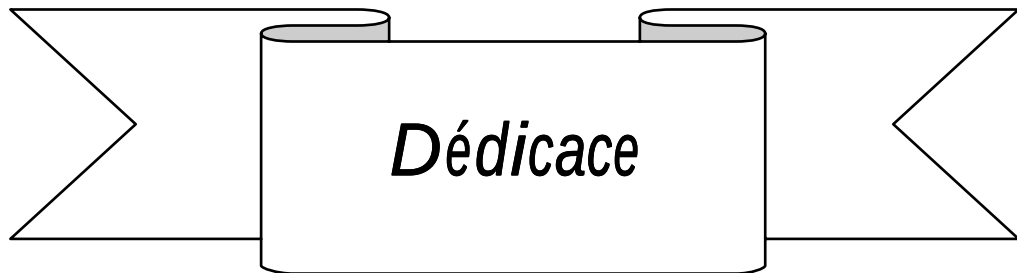
Etudié par :
DJENOURI Djamel

Devant le jury composé de :

Monsieur AISSANI
Madame AISSANI
Monsieur BELKHIR

Président
Membre
Membre

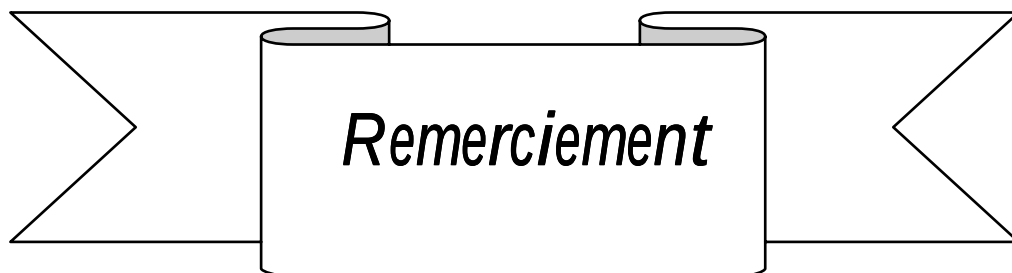
12 juillet 2003



*Je suis heureux d'avoir ici l'occasion d'exprimer mes bien
vives gratitude, et mes chaleureuses salutations à:*

- ❖ *Mes chers parents Nasserredine et Fatima-Elzohara*
- ❖ *Ma chère fiancée Fella AMRAR*
- ❖ *Mes chers frères Rachid et Youssef*

Je leur dédié ce modeste travail



Remerciement

Je tiens à remercier mon directeur de recherche Dr Nidjib badache, d'abord de m'avoir fait confiance en me proposant ce sujet, ensuite pour sa disponibilité, ses lectures, et ses critiques et conseils constructifs.

J'adresse également mes sincère remerciement à mademoiselle Hassina ALIANE, le chef de notre laboratoire au CERIST, pour sa compréhension, et pour tous les moyens qu'elle a mis à ma disposition

Grand merci à mon cher père, Nasserdine DJENOURI pour ses lectures attentives, ses encouragements, et son soutien.

Indexe

Introduction	1
 Chapitre I : Terminaux sans fil	
I.1 Introduction	4
I.2 Classification des terminaux sans fil	4
I.2.1 LAPTOPS (ordinateurs portables)	4
I.2.2 Tablettes à crayon (« PEN TABLETS »)	6
I.2.3 Ordinateurs personnels de poche (HANDHELD PERSONAL COMPUTERS HPCs)	6
I.2.4 Assistants numériques personnels (Personal Digital Assistants PDAs)	7
I.2.5 Téléphones cellulaires (Cellular Phones)	8
I.2.6 Cartes réseaux locaux sans fil (WIRELESS LOCAL AREA NETWORK WLAN, et BlueTooth)	9
I.2.7 Les unités combinées (COMBINATION DEVICES)	9
I.3 Caractéristiques et challenges des terminaux existants	10
I.3.1 Efficacité en énergie (ENERGY EFFICIENCY)	10
I.3.2 Auto configuration	13
I.3.3 Adaptabilité	13
I.3.4 Connaissance de la localisation et du contexte	13
I.3.5 Partitionnement fonctionnel entre le terminale et le réseau	13
I.3.6 Sécurité	13
I.3.7 Interface utilisateur	14
 Chapitre II : Les réseaux mobiles ad hoc et le problème de routage	
II.1 Introduction	16
II.2 les réseaux cellulaires	16
II.3 Les réseaux ad hoc	17
II.3.1 Définition	18
II.3.2 Quelque type d'application	19
II.3.3 Les caractéristiques des réseaux ad hoc	20
II.3.4 Le problème de routage dans les réseaux ad hoc	21
II.3.5 Clasification des protocole de routage	22
II.3.5.1 <i>Notions générales utilisées par les protocoles de routage</i>	23
II.3.5.2 <i>Les protocole de routage pro-actifs</i>	22
II.3.5.3 <i>Les protocole de routage réactifs</i>	26

Chapitre III : Techniques et métriques proposées pour minimiser la consommation d'énergie

III.1 Introduction	29
III.2 Mécanisme pour réduire la consommation d'énergie	29
III.2.1 Contrôle de la puissance de transmission	29
III.2.2 Eteindre l'interface réseau	32
III.2.3 Utilisation des antennes dirigées (DIRECTIONAL ANTENNAS)	33
III.2.4 Utilisation d'un protocole de routage efficace	34
III.3 Caractéristiques désirables d'un protocole de routage efficace en consommation d'énergie	35
III.3.1 Calcul du coût d'un lien et contrôle de la puissance de transmission de paquets de données	35
III.3.2 Permettre de prendre en compte toutes les routes possibles, pour découvrir la route à énergie minimale	36
III.3.3 Suivre la trace de coût des liens	36
III.3.4 S'adapter à l'augmentation du nombre de nœuds	37
III.4 Métrique d'un protocole de routage de consommation minimale d'énergie	38
III.4.1 Minimiser l'énergie consommée par paquet	38
III.4.2 Maximiser le temps avant le partitionnement du réseau	39
III.4.3 minimiser la variance d'énergie entre les nœuds	39
III.4.4 Minimiser le coût par paquet	40
III.4.5 Minimiser le coût maximal	41

Chapitre IV : Proposition de nouvelles métriques et d'un protocole de routage

IV.1 Introduction	44
IV.2 Présentation générale du protocole DSR (Dynamic Source Routing)	45
IV.2.1 Découverte de route	45
IV.2.2 Maintenance de route	46
IV.2.3 Optimisation	47
IV.3 Implémentation du mécanisme d'utilisation de puissances de transmission adaptatives	48
IV.4 Métriques proposées	50
IV.5 politique round robin sur les chemins	54
IV.6 Proposition du protocole	57
IV.6.1 les principales modifications	57
IV.6.2 Protocole	58
IV.6.3 Étude de complexité	64

Chapitre V : Environnement de simulation	
V.1 Introduction	69
V.2 Généralités sur la simulation	69
V.2.1 Système réel et objectif de simulation :	69
V.2.2 Limite de l'expérimentation directe	70
V.2.3 Systèmes discret et continue	70
V.2.4 Modèles de simulation	70
V.2.5 Gestion du temps	71
V.2.6 Simulation par événements discrets	71
V.2.7 Simulateur	71
V.3 PARSEC	72
V.3.1 Introduction	72
V.3.2 Entité	72
V.3.3 Message	72
V.3.4 événement	73
V.3.5 L'exécution parallèle	73
V.3.6 Exécution de parsec	74
V.4 GloMoSim	75
V.4.1 Introduction	75
V.4.2 Agrégation des nœuds	77
V.4.3 Structure des répertoires de GloMoSim	77
V.4.4 Exécution de GloMoSim	78
V.4.5 Les APIs de GloMoSim	79
V.4.6 Description du fichier d'entrée (de configuration)	79
V.4.7 Les statistiques obtenues dans GloMoSim	83
 Chapitre VI : Démarche et Résultats de Simulations	
VI.1 Introduction	88
VI.2 Environnement de simulation	88
VI.2.1 Extension de GloMoSim	88
VI.2.2 métriques de comparaison	89
VI.2.3 Paramètre et models utilisés	91
VI.2.4 démarche de la simulation	94
VI.3 Résultats	95
VI.4 conclusions	105
ANNEX	107
BIBLIOGRAPHIE	110

Introduction générale

Les réseaux sans fil ad hoc sont des réseaux à plusieurs sauts (MULTI HOP), sans infrastructure (INFRASTRUCTELESS NETWORKS), où les nœuds forment l'infrastructure du réseau, et maintiennent d'une manière coopérative la connectivité du réseau. Leur aptitude à être installés rapidement, et devenir opérationnels sans avoir besoin d'infrastructure filaire fixe, rend ces réseaux des candidats prometteurs pour plusieurs types d'application.

Chaque nœud du réseau joue en même temps le rôle d'un routeur et d'un hôte. En plus des services propres à son utilisateur, chaque nœud doit transmettre les paquets pour les autres, les paquets sont alors envoyés via plusieurs hôtes mobiles intermédiaires, et suivent ainsi un chemin multi sauts (MULTI-HOP). Pour cela, un protocole de routage distribué est nécessaire.

Les nœuds d'un réseau ad hoc sont des calculateurs portables sans fil (LAPTOP, POCKET PC...etc). Ils sont caractérisés par: des ressources modestes (CPU, espace de stockage, la bande passante), et surtout, sont alimentés par des ressources d'énergie limitées, ce qui imposent nécessairement des contraintes pour concevoir des protocoles de routage performants. De là, les protocoles doivent prendre en considération la contrainte de limitation en ressource d'énergie.

En plus, les réseaux Ad Hoc possèdent un certain nombre de caractéristiques qui imposent de nouvelles exigences aux protocoles de routage. Citons la limitation de la bande passante, et le changement fréquent de la topologie qui est la conséquence de la mobilité des nœuds. Ce changement de topologie va influencer sur les chemins utilisés pour le transfert de paquets. Donc le protocole de routage doit s'adapter rapidement à ce changement, en tenant compte de la contrainte d'énergie.

Ce travail est consacré à l'étude de la consommation d'énergie pour le routage dans les réseaux mobile ad hoc. Nous y proposons une solution en définissant de nouvelles métriques et techniques. En se basant sur ces techniques et métriques, nous proposerons un nouveau protocole de routage efficace en consommation d'énergie. Les performances de ce protocole sont évaluées par simulation.

Ce document est composé de six chapitres:

Le premier chapitre est une présentation des calculateurs portables, et de leurs caractéristiques. Le second est une introduction aux réseaux mobiles, et particulièrement aux réseaux ad hoc, et au problème de routage. On y trouve aussi une présentation générale des protocoles de routage standards destinés aux réseaux ad hoc.

Le troisième chapitre est une description des mécanismes proposés pour réduire l'énergie consommée par les terminaux mobiles dans les réseaux ad hoc. Il contient aussi, une description des caractéristiques et métriques souhaitables d'un protocole de routage efficace en consommation d'énergie.

Le quatrième chapitre consiste à décrire les nouvelles métriques et techniques que nous proposons, ainsi que notre nouveau protocole de routage. Dans le chapitre cinq on trouve une introduction à la simulation, et une présentation de l'environnement de simulation que nous avons utilisé. Cet environnement est composé du langage PARSEC et du simulateur GloMoSim développés à l'université californienne UCLA. Et enfin le sixième et dernier chapitre est une étude de performance du protocole proposé par simulation. Dans ce chapitre, on trouve une description de la démarche de simulation, et une présentation des résultats obtenus.



Chapitre 1

*Les terminaux utilisés dans les
réseaux mobiles*

I. Introduction

Les réseaux sans fil ont été créés, au début, pour assurer les services téléphoniques à des usagers munis de téléphones *portables*, ce qui leur permet d'être *mobiles* pendant la communication. Les progrès technologiques ont dépassé la téléphonie, en produisant des *ordinateurs* de plus en plus petits, légers et portables, des terminaux sans fil [PLA98], tel que des détecteurs sans fil [GAA97, LWI99], des cameras sans fil [GJA96], etc..... Ainsi les réseaux *informatiques* mobiles ont vu le jour, et la communication entre les utilisateurs munis de calculateurs portables via des liaisons sans fil devient de plus en plus populaire.

Cependant, les calculateurs portables sont moins performants que les ordinateurs classiques et ils sont alimentés par des sources d'énergies autonomes (batteries), donc limitées. Les liaisons sans fil sont moins fiables et moins sécurisées que les liaisons filaires. Il faut donc tenir compte de toutes ces contraintes en concevant des applications ou des protocoles destinés aux réseaux sans fil.

Dans ce chapitre, on va présenter les calculateurs portables. En commençant par les classer, en suite, on parlera des caractéristiques des calculateurs existants et des challenges de conception.

II. Classification des terminaux sans fil

Dans [PLM99], Paul Lettieri et al. ont classé les terminaux sans fil en six catégories, ils ont exclu les « LAPTOS» jugés grands et volumineux, nous reprenons cette classification en ajoutant des exemples de terminaux commercialisés, en y incluant les « LAPTOS» qui sont quand même des terminaux sans fil et portables.

1 LAPTOS (ordinateurs portables)

Ce sont des PCs plus ou moins portables. Actuellement, ils ont des caractéristiques très proches de celles des PCs classiques, munis d'interfaces de communications sans fil, et peuvent être alimentés par des batteries. Cependant, ils sont plus lourds et consomment beaucoup d'énergie par rapport aux autres terminaux décrits ci-après.

Le tableau suivant (figure 1.2) présente quelques LAPTOS que l'on trouve actuellement sur le marché.



Figure 1.1 : LAPTOP

Marque	Prix	Processeur	écran	RAM	Poids(lb)	#Port USB
LAPTOPS Windows						
Dell inspiron 4100	\$1605	1GHz Pentium 3	14 ``	256	6.2	1
Toshiba Satellite 3005-S304	1600	1GHz Pentium 3	14 ``	256	6.5	3
Compaq 1700	1590	1GHz Pentium 3	14 ``	256	6.0	2
HP Pavilio n5426	1400	900 MHz athlon-M	14 ``	256	7.5	2
Gateway solo 5300cl	2020	1GHz Pentium 3	14 ``	256	5.7	1
IBM thinkPad R30 2656-60u	1600	1GHz Pentium 314 »	5.7	128	5.7	2
LAPTOP Macintosh						
Apple iBook	1500	60MHz G3	12.1	128	5.1	2

Figure 1.2 Quelques LAPTOPS de l'année 2002 [CON02]

2 Tablettes à crayon (« PEN TABLETS »):

Ce sont des LAPTOPs sans clavier, ce dernier est remplacé, comme le nom l'indique, par un crayon. Ils sont munis d'écrans couleur VGA(640*480), de processeurs Pentium et des disques durs. Ils sont exploités par Windows9x ou WindowsNT.

Le poids d'une PEN TABLET représente, généralement, 50 % de celui d'un LAPTOP. Le crayon remplace la souris pour les pointages et les cliques sur l'écran, quant à la reconnaissance du texte manuscrit (« HANDWRITTEN RECOGNITION »), elle est rarement utilisée par ce terminal. Et pour la saisie, un clavier virtuel affiché sur l'écran, peut être utilisé .

On trouve sur le marché plusieurs produits appartenant à cette classe de terminaux. Nous citons comme exemples : Orasis du Dauphin, Stylistic 1000 de Fujitsu, AMITY VP de Mitsubishi et Panasonic CF-01. Notons que quelques terminaux, tels que Stylistic de Fujitsu, possèdent une carte modem intégrée, alors que d'autres permettent l'ajout d'une carte modem PC.

A part l'absence du clavier, une PEN TABLET possède, pratiquement, les performances d'un « LAPTOP » moyen.



Figure 1.3 PEN TABLET

3 Ordinateurs personnels de poche (HANDHELD PERSONAL COMPUTERS HPCs)

Les terminaux de cette catégorie se ressemblent. Caractérisées par un clavier d'une forme réduite, et d'un petit écran couleur VGA (HALF-VGA 640*240), ils sont considérés comme des LAPTOPS miniaturisés. On les nomme aussi PALMTOP ou POCKET PC.

Ils ne contiennent pas de disques durs, ils tournent donc sous des plates formes destinées aux systèmes sans unités de stockage permanentes, telle que Windows CE qui permet d'exécuter des versions réduites d'applications Windows : traitement de texte, tabulation, présentation, jeu vidéo etc.....

Les microprocesseurs utilisés dans ces terminaux sont rarement ceux de la famille Intel, on utilise souvent les microprocesseurs de MIPS et Hitachi SupreH. Et pour la communication, des interfaces modem et des ports série et infrarouge sont disponibles.

Un HPC pèse environ 1 à 2 lb (POUND), ce qui est équivalent à 1/3 du poids d'un LAPTOP.

Cossiopeia A-20 de Casio, palmtop PC 620LX de Hewlett packard, Velo-500 de Philips, Phenon Yltra de LG Electronics, Pocket PC H3850 de Compaq et PocketPC Falcon 4210 sont des exemples des HPCs.



PocketPC Falcon 4210

Ecran ¼ VGA, clavier 16 touches,
Batterie Lithium-Ion rechargeable,
microsoft pocket pc OS

Figure 1.4 Exemple de HPC

4 Assistants numériques personnels (Personal Digital Assistants PDAs)

Contrairement aux HPCs, les PDAs sont des unités monolithiques, ils consomment moins de ressources (ex écran 1/4 VGA, RAM plus limitée). Ils ont les dimensions d'une main, à peu près 3 in * 5 in * 3/4 in ($\approx 8\text{ cm} * 12.5\text{ cm} * 15/8\text{ cm}$). Un PDA ne comporte pas de clavier, en utilise généralement des boutons et/ou un crayon avec des techniques de reconnaissance de textes manuscrit, il contient des ports séries pour permettre la connexion à un PC, un autre ordinateur portable ou un autre périphérique, et parfois il contient même une carte modem. Il tourne sous la plate forme Windows CE.

On trouve sur le marché plusieurs marques de PDAs, nous citons entre autre : Everx Freestyle, Philips Nino, et avigo de Texas Instruments, et iPAQ H3135 de American Best Computing.



- 5 boutons de navigation,
- 16 Mo de RAM
- Batterie Lithium-Ion rechargeable
- poids = 6 oz (≈ 170 g)

iPAQ H3135

Figure 1.5 Exemple de PDA

5 Téléphones cellulaires (Cellular Phones)

Contrairement aux terminaux précédents, les téléphones cellulaires sont centrés entièrement sur l'aptitude de communication, bien que les PDAs et les HPCs peuvent communiquer et synchroniser les données avec un ordinateur de bureau, et même échanger les données avec d'autres ordinateurs portables, imprimantes ou autres périphériques. Actuellement la fonction principale de ces unités est le maintien de rendez-vous, des contacts, et autres notes et données personnelles, alors que les téléphones cellulaires ne sont rien d'autre que des unités de communication, et de transmission de parole, utilisés dans les réseaux cellulaires à longue distance. Leur fonctionnalité est complémentaire de celle des calculateurs précédents, il sont plus petits mais consomment plus d'énergie à cause des puissances des fréquences utilisées.

On trouve sur le marché différents modèles et marques de plusieurs sociétés qui fabriquent ces téléphones cellulaires entre autres Nokia, Ericsson, Motorola, et Sumsung.



Figure 1.6 Téléphone cellulaire

6 Cartes réseaux sans fil (WIRELESS LOCAL AREA NETWORK WLAN, et BlueTooth)

Ce sont des cartes qu'on peut ajouter aussi bien aux ordinateurs portables qu'aux périphériques sans fil. Elles sont utilisées dans les réseaux d'intérieur (INDOOR NETS), par exemple dans des immeubles de bureaux, ou des hôpitaux, qui occupent une zone de quelques dizaines voire de centaines de mètres. Contrairement aux réseaux cellulaires qui occupent plusieurs kilomètres (OUTDOOR NETS).

Elles permettent une utilisation en mode infrastructuré (à station de base), ainsi qu'en mode ad hoc (voir dans les chapitres suivants). Les cartes WLAN utilisent les normes IEEE 802.11, tandis que les cartes BlueTooth utilisent, comme leur nom l'indique, les normes de BlueTooth [JHM98] qui permettent une interconnexion facile des unités hétérogènes.

Ces cartes sont caractérisées entre autres par le débit maximal, et la portée de communication, ou le domaine de puissance, qui représente la distance maximale sur laquelle, une carte peut émettre un signal. Voici une comparaison de quelques caractéristiques, entre deux types de cartes.

	WLAN	BlueTooth
Access Method	DSSS/FHSS	FSK
Transmit Baudrate	1Mbps, 2Mbps, 5.5Mbps, 11Mbps	1Mbps or 2Mbps
Transmit Max Range	above 100 meters	10 to 100 meters
Power	High	Low
Security	Optional WEP	Authentication Key, mgmt, Encryption

Figure 1.7 Comparaison entre WLAN et BlueTooth

7 Les unités combinées (COMBINATION DEVICES)

Ce sont des unités qui n'appartiennent à aucune des catégories précédentes, On cite :

- Nokia 9000[NOK00, NOK01] : qui peut être considéré comme un terminal qui combine un HPC et un téléphone cellulaire.
- CruisePAD de Cruise Technology: il ressemble au «PEN TABLET» décrit précédemment, mais en fait ce n'est qu'un terminal d'E/S muni d'une interface WLAN, et d'une CPU i486, il est généralement connecté à un ordinateur de bureau qui joue le rôle d'un serveur, et c'est à son niveau que les traitements réels se réalisent.
- REX de Franklin Electronics [FRA] : c'est un terminal d'E/S mais qui est capable d'une légère manipulation des données.



Figure 1.8 Nokia 9000

III. Caractéristiques et challenges des terminaux existants

Les terminaux sont partagés principalement en deux catégories :

- Les terminaux de communication (téléphones cellulaires et interfaces réseaux) : il sont principalement des supports de communication
- les terminaux de traitement d'information (LAPTOPS, PEN TABLETs, et HPCs, PDAs) : leur but est de rétrécir un PC classique en une unité plus petite, légère, et donc portable. Pour la communication, on utilise des interfaces réseaux qui sont généralement des cartes reliées à des bus internes, et ne sont pas intégrées au sein de l'architecture de base. On trouve ce principe même dans l'architecture de NOKIA9000 [NOK01], ceci augmente la consommation d'énergie de ces interfaces [PLM99].

Dans ce paragraphe, nous parlons des caractéristiques souhaitées des terminaux sans fil, et des challenges de conception :

1 Efficacité en énergie (ENERGY EFFICIENCY)

À cause des contraintes de mobilité et portabilité, les terminaux sans fil sont alimentés par des sources d'énergie autonomes (batteries), qui sont limitées. Une des caractéristiques désirables pour ces terminaux et leurs applications est d'assurer la plus longue vie à la batterie. Ceci en produisant des batteries plus petites et durables, des composants électroniques plus intégrés et qui consomment moins d'énergie, et surtout des logiciels qui prennent en compte la contrainte d'énergie.

1.1 Batteries

Les terminaux sans fil utilisent des batteries rechargeables. Les progrès technologiques permettent de fabriquer des batteries de plus en plus durables. Ceci en augmentant la densité (capacité par rapport au poids ou au volume), ce qui permet de fabriquer des

batteries de plus en plus chargées et petites. Les principaux types de batteries utilisées sont : Nickel Cadmium (NiCad), Nickel-Metal Hydride (NiMH) et Lithium-ion (Li-ion ou LiOn). Les deux premières classes (à base du Nickel) sont plus économiques mais elles perdent plus rapidement leurs charges quand elles ne sont pas utilisées[AOL03]. Le tableau de la figure 1.9 montre les caractéristiques de quelques batteries commercialisées.

Toutefois, le progrès dans la fabrication des batteries est loin de satisfaire les besoins de calcul et de communication des terminaux, surtout si on le compare au progrès de fabrication des circuits électroniques (leur complexité se multiplie chaque 18 mois [PLM99]), la figure 1.10 montre l'évolution des capacités des batteries NiCad et NiMH depuis 30 ans.

La durée de vie (pour une mise en marche continue) des batteries actuelles varie entre quelques heures pour les LAPTOPS à quelque dizaines d'heures pour les PDAs[PCW03].

Il faut noter aussi les progrès dans la fabrication des chargeurs portables de batteries, entres autres citons le « methanol fuel cell » de la société MTI Micro, fabriqué en août 2002, qui possède la capacité d'une dizaine de batteries, et PowerPad 120 et 160 de la firme Canadienne Electrovaya [PCW03]. Mais toutes ces sources portables sont limitées, et n'empêchent pas les déconnexions fréquentes d'un terminal pour le recharger.

Toutes ces contraintes imposent une bonne gestion de la source d'énergie critique pour augmenter sa durée de vie.

Battery	Rechargeable ?	Gravimetric density (Wh/lb)	Volumetric density (Wh/lb)
Alkaline-MnO ₂ (typical AA)	No	65.8	347
Silver oxide	No	60	500
Li/MnO ₂	No	105	550
Zinc air	No	140	1150
NiCd	Yes	23	125
Li-Polymer	Yes	65-90	300-415

Figure 1.9 Caractéristiques de quelques batteries commerciales [PLM99]

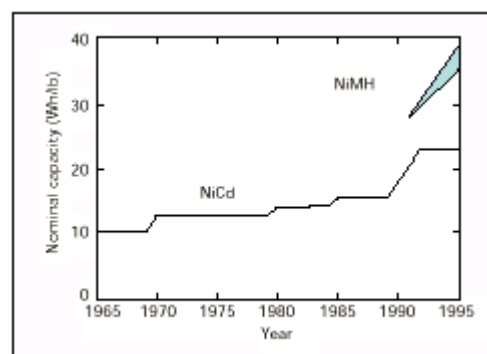


Figure 1.10 l'évolution de capacités des batteries NiCad et NiMH au cours de 30 ans.

1.2 Comment diminuer la consommation d'énergie ?

Il y a deux niveaux complémentaires de solutions pour atteindre cet objectif :

- Solutions électroniques (LOW LEVEL SOLUTION) : ces solutions ont été abordées durant ces dernières années. Elles concernent la conception des circuits électroniques à grande échelle d'intégration et qui fonctionnent sous des puissances minimales. Plusieurs techniques ont vu le jour; nous citons entre autres : le parallélisme ou le pipeline en conjonction avec le changement de tension (voltage scaling), qui permettent de minimiser l'énergie consommée par les puces [PLM99]. Ces solutions ont donné de bons résultats, mais restent insuffisantes.
- Solutions logiciels (HIGHER LEVEL SOLUTION): les protocoles et applications utilisés doivent prendre en considération l'énergie nécessaire pour le fonctionnement, en minimisant la communication et les traitements, on dit dans [PLM99] que les solutions logicielles représentent la clé pour l'efficacité d'énergie dans le futur, (*The key to energy efficiency in future wireless terminals will be designing in higher layers of the wireless terminal*).

1.3 Energie consommée en différents modes de fonctionnement

On distingue principalement trois modes de fonctionnement des CPUs: allumé (on), veille (standby), et arrêté (STOPPED), et trois autres pour les interfaces de communication: envoi de message, réception de message, et «sleep» (on peut aussi trouver le mode «IDLE»). Cependant, il y a une grande différence de l'énergie consommée entre ces modes (figure 1.10). La plupart du temps, les composants des terminaux ne sont pas utilisés, donc il faut les éteindre durant ces périodes pour gagner l'énergie. En d'autres termes il faut mettre automatiquement les composants dans le mode qu'il faut. Il a été montré dans [PLM99] que théoriquement, on peut économiser jusqu'à 69 fois de l'énergie consommée, si les composants sont toujours dans le mode adéquat.

En pratique, ce n'est pas du tout évident d'atteindre ces performances, le basculement automatique entre les modes est un grand problème, et deux questions se posent : quand

faut-il éteindre un composant? et quand faut-il le rallumer ? Plusieurs politiques ont été proposées. On cite :

- Quand faut-il éteindre un composant ?
 - après une période donnée en mode veille «IDLE» : Cette politique possède une difficulté concernant la détermination de cette période. En plus, elle a l'inconvénient du gaspillage d'énergie durant cet intervalle.
 - En utilisant des estimations basées sur l'historique du fonctionnement : Cette politique n'est pas toujours efficace.
- Quand faut-il le rallumer ?
 - en répondant à un certain événement, tel que l'interaction d'utilisateur ou de ses applications, ou l'arrivée d'un paquet de données.
 - Après une durée à prédire par le système.

Toutes ces politiques peuvent être implantées soit d'une manière câblée par l'électronique, soit d'une façon programmée en ajoutant des gestionnaires de puissances (power managers) au système d'exploitation, ou en fournissant des APIs aux applications (pour la remise en marche).

Composant\mode	ON	STANDBY	STOPPED
ARM-based Ruby 2 CPU	150 mW	7.5 mW	0.15 mW
	Transmission	Réception	SLEEP
WaveLAN PCMCIA	3 W	1.48 W	0.18 W
GEC Plessey'sDE6003	1.8 W	0.6 W	0.05 W

Figure 1.11 Energie consommée en différents modes

2 Auto configuration

C'est la possibilité qu'a un terminal pour découvrir automatiquement les paramètres nécessaires dans son environnement, tel que l'adresse IP, les serveurs DHCP, DNS etc..., et se configurer automatiquement (SELF-CONFIGURE), et devenir opérationnel, aussi bien dans un réseau à distance (cellulaire), que dans un réseau WLAN . Ceci sans aucune intervention d'un administrateur,

3 Adaptabilité (« ADAPTIVITY »)

C'est la possibilité de s'adapter aux changements des conditions des canaux de communications durant le temps (« TIME-VARYING RADIO CHANNEL »), pour pouvoir fonctionner dans les environnements irréguliers.

4 Connaissances de la localisation et du contexte (« LOCATION AND CONTEXT AWARENESS »)

Où se trouve le terminal ? Quelle est sa vitesse ? Son accélération ? Quel est la température ? La pression ? La vitesse du vent ? Quels sont les utilisateurs (terminaux) qui se trouvent à son voisinage ? Grâce aux micro-détecteurs et au système GPS (Global Positioning System), il est possible au terminaux de répondre à toutes ces questions, et construire ainsi le contexte nécessaire.

5 Partitionnement fonctionnel entre le terminal et le réseau

Le partitionnement fonctionnel entre le terminal et le réseau est une décision architectural très importante. Il permet de préciser où les applications peuvent s'exécuter, et où les données seront sauvegardées. Ainsi que la complexité du terminal et le coût du service de la communication.

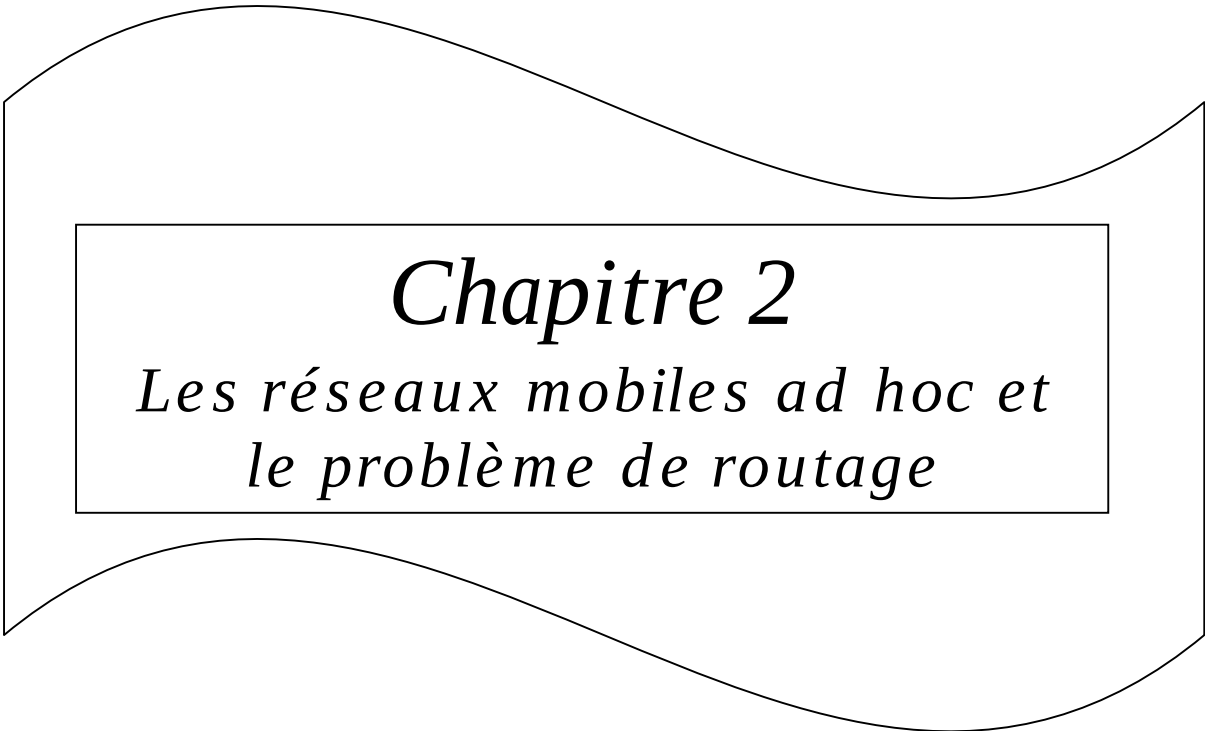
6 Sécurité

La sécurité a été toujours un facteur important dans la conception des réseaux, et elle est encore plus quand le réseau est sans fil (l'environnement des terminaux sans fil). La sécurité inclus aussi bien la sécurité des données contre des pertes et des corruptions, que leur confidentialité. Il faut donc protéger les données dans leurs points de stockages, ainsi

que durant les transmissions. Les techniques classiques telles que le cryptage et la duplication d'information peuvent être utilisées, mais, et à cause des contraintes de l'environnement, les solutions sont loin d'être évidentes.

7 interfaces utilisateurs

Les interfaces utilisateurs doivent prendre en considération les contraintes des terminaux, principalement la taille réduite de leurs écrans. Elles doivent alors être plus simples et petites, et contenir des parties invisibles (en arrière plan), pour ne pas condenser l'écran.



Chapitre 2

*Les réseaux mobiles ad hoc et
le problème de routage*

I. Introduction

L'évolution rapide de la technologie dans le domaine de la communication sans fil, a permis à des usagers munis d'unités de calcul portables, d'accéder à l'information indépendamment des facteurs : temps et lieu. Comme on l'a vu dans le chapitre précédent, ces unités, qui communiquent à travers des interfaces sans fil, peuvent être de diverses configurations : avec ou sans disque, des capacités de sauvegarde et de traitement plus ou moins modestes. Et il sont alimentés par des *ressources d'énergie autonomes* (batteries). L'environnement de calcul résultant est appelé *environnement mobile*. Cet environnement n'astreint plus l'utilisateur à une localisation fixe, mais lui permet une libre mobilité tout en assurant sa connexion au réseau.

Les réseaux mobiles ou sans fil, peuvent être classés en deux classes : les réseaux cellulaires avec infrastructure (INFRASTRUCTURED NETS), et les réseaux ad hoc sans infrastructure fixe (INFRASTRUCTURLESS NETS) [BAD98].

II. les réseaux cellulaires

Le modèle de système intégrant des sites mobiles et qui a tendance à se généraliser, est composé de deux ensembles d'entités distinctes: les sites fixes d'un réseau de communication filaire classique, et les sites mobiles. Certains sites fixes, appelés stations support mobile (Mobile Support Station) ou station de base (SB), sont munis d'une interface de communication sans fil pour la communication directe avec les sites ou unités mobiles (UM), localisés dans une zone géographique limitée, appelée cellule (voir figure 2.1).

A chaque station de base correspond une cellule, à partir de laquelle des unités mobiles peuvent émettre et recevoir des messages. Alors que les sites fixes sont interconnectés entre eux à travers un réseau de communication filaire, généralement fiable et d'un débit élevé, les liaisons sans fil ont une bande passante limitée qui réduit sévèrement le volume des informations échangées [DDN92].

Dans ce modèle, une unité mobile ne peut être, à un instant donné, directement connectée qu'à une seule station de base. Elle peut communiquer avec les autres sites à travers la station à laquelle elle est directement rattachée. L'autonomie réduite de sa ressource d'énergie, lui occasionne de fréquentes déconnexions du réseau; sa reconnexion peut alors se faire dans un environnement nouveau voire dans une nouvelle localisation. Un exemple typique des réseaux cellulaires est le réseau GSM (Global System Mobile)

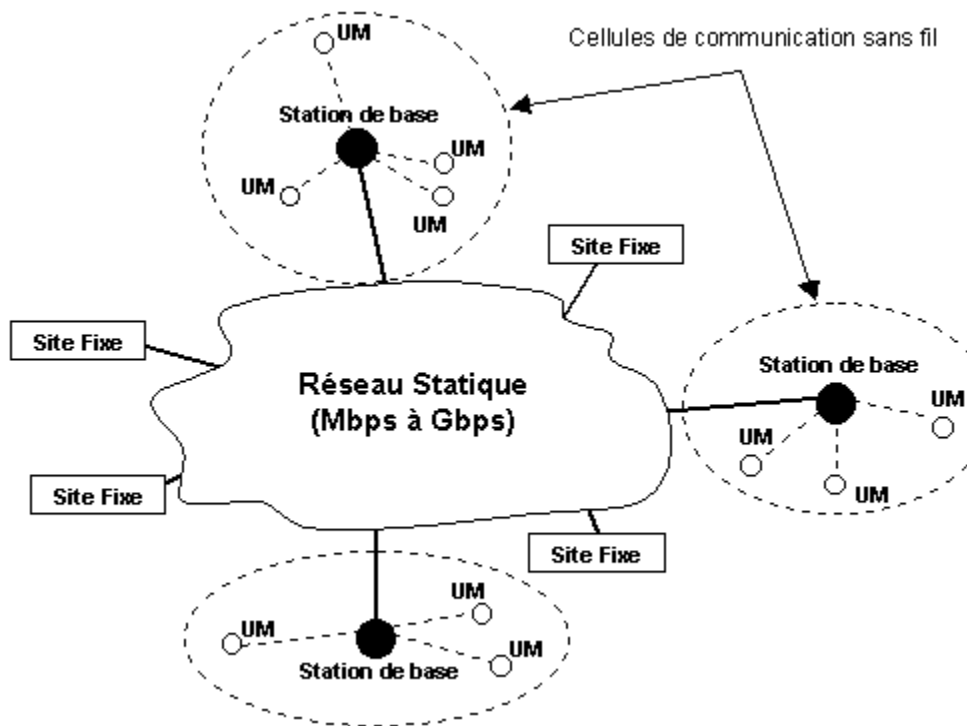


Figure 2.1 : Le modèle des réseaux mobiles avec infrastructure

III. Les réseaux mobiles Ad Hoc

L'évolution récente de la technologie dans le domaine de la communication sans fil et l'apparition des unités de calcul portables poussent aujourd'hui les chercheurs à faire des efforts afin de réaliser le but des réseaux : "l'accès à l'information n'importe où et n'importe quand" [BAD98].

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

Les réseaux ad hoc sont idéals pour les applications caractérisées par une absence (ou la non-fiabilité) d'une infrastructure préexistante, tel que les applications militaires et les autres applications de tactique comme les opérations de secours (incendies, tremblement de terre..) et les missions d'exploration.

3.1 Définitions

On trouve dans la littérature plusieurs définitions d'un réseau ad hoc:

- Un réseau mobile Ad hoc est une collection de calculateurs portable sans fil, formant un réseau temporaire sans aucune infrastructure filaire [KWJ01]
- Les réseaux sans fil ad hoc sont des réseaux à plusieurs sauts (MULTI HOP), où les nœuds maintiennent d'une manière coopérative la connectivité du réseau. Leur possibilité d'être installés rapidement, et devenir opérationnels sans avoir besoin d'infrastructure filaire fixe, rend ces réseaux des candidats prometteurs pour les applications militaires, et les opérations de secours [ASC02].
- Un réseau mobile ad hoc, appelé généralement MANET (Mobile Ad hoc NETwork), consiste en une grande population, relativement dense, d'unités mobiles qui se déplacent dans un territoire quelconque et dont le seul moyen de communication est l'utilisation des interfaces sans fil, sans l'aide d'une infrastructure préexistante ou administration centralisée [LEM00].

A un instant t , un réseau ad hoc peut être modélisé par un graphe $G_t = (V_t, E_t)$ où V_t représente l'ensemble des nœuds (i.e. les unités ou les hôtes mobiles) du réseau, et E_t modélise l'ensemble des connexions qui existent entre ces nœuds (voir la figure 2.2). Si $e = (u, v) \in E_t$, cela veut dire que les nœuds u et v sont en mesure de communiquer directement à l'instant t , ou plutôt chacun des nœuds est à la portée de communication de l'autre.

Si les nœuds utilisent des interfaces de communication qui ont des domaines de puissance différents, le graphe doit être orientée. Un arc $(u, v) \in E_t$ modélise un lien de u vers v , c à d que v est à la portée de communication de u , et l'inverse n'est pas forcément vrai.

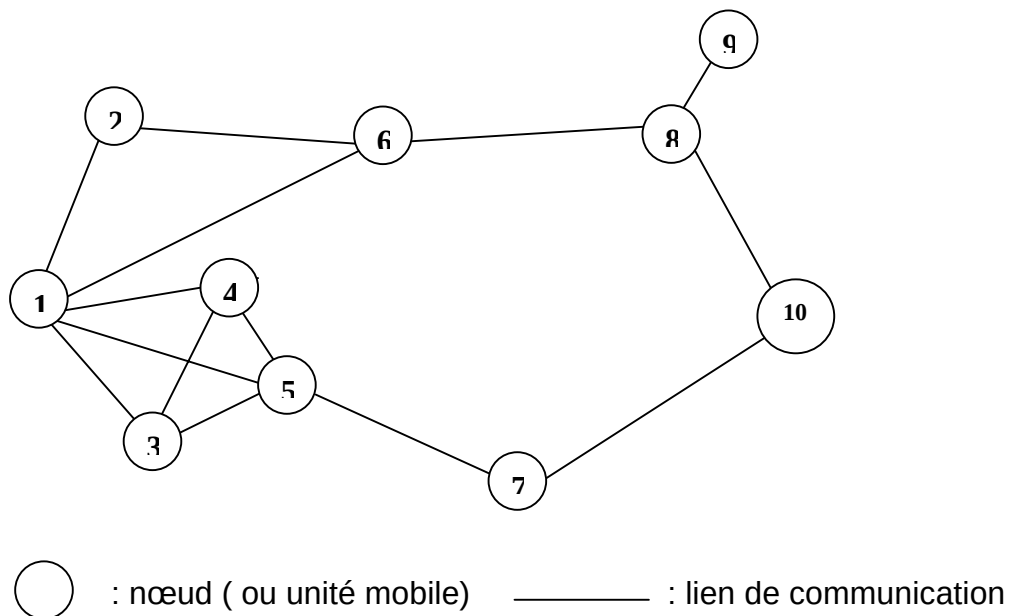


Figure 2.2 : La modélisation d'un réseau ad hoc

La topologie du réseau peut changer à tout moment (voir la figure 2.3), elle est donc dynamique et imprévisible, ce qui fait que la déconnexion des unités soit très fréquente.

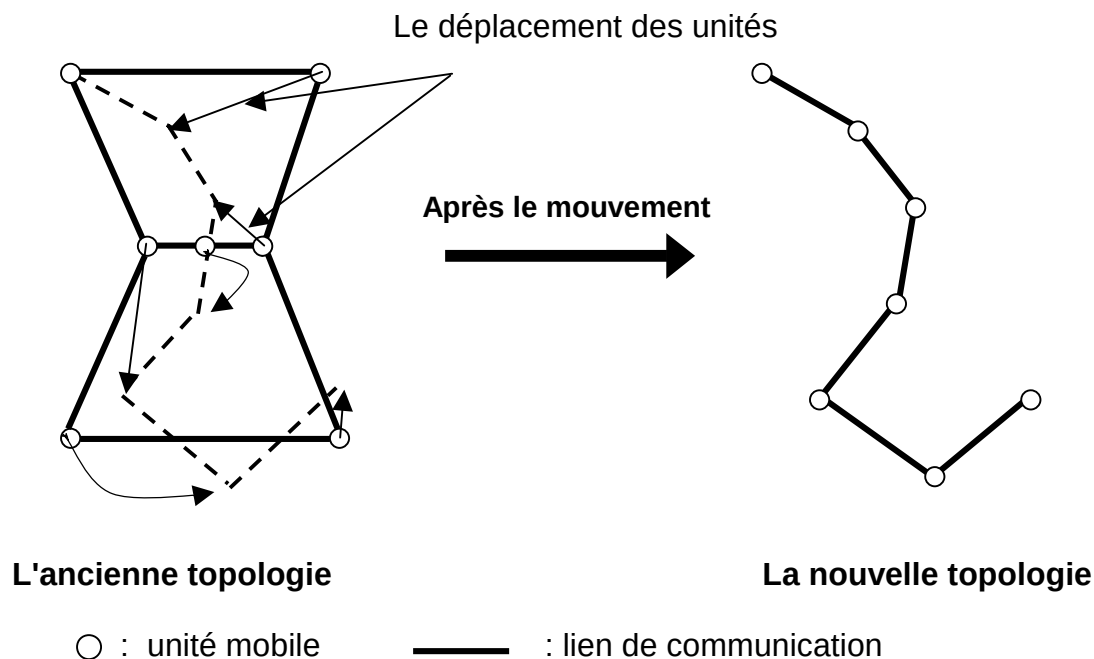


Figure 3.2 : Le changement de la topologie des réseaux ad hoc

Notons que les réseaux GSM ne représentent pas des réseaux ad hoc, car la communication entre les unités passe obligatoirement par des stations de base du réseau filaire.

3.2 Quelques types d'applications des réseaux mobiles ad hoc

Les applications ayant recours aux réseaux ad hoc couvrent un très large spectre. D'une façon générale, les réseaux ad hoc sont utilisés dans toute application où le déploiement d'une infrastructure réseau filaire est trop contraignant, soit à cause de la difficulté de la mise en place, ou parce que la durée d'installation du réseau ne justifie pas de câblage à demeure.

Parmi ces applications on trouve :

Applications militaires : Les équipements militaires (troupes, véhicules) peuvent prendre les avantages des réseaux Ad Hoc. Ces réseaux leur permettent de se déplacer et de communiquer entre eux dans les champs de bataille, en maintenant les informations entre les soldats et les chefs des troupes [GRA00] .

Opérations de secours : Les opérations de secours se passent dans des régions désastreuses (incendie, inondation, tremblement de terre) ou une infrastructure de réseau filaire ou station de base n'existent pas, et une communication entre les équipes de secours est nécessaire [GRA00]. Dans ce cas, une installation d'un réseau Ad Hoc peut résoudre ce problème.

Les conférences : Le réseau Ad Hoc est faisable pour un groupe de gens qui veulent créer un réseau temporaire. Par exemple un réseau Ad Hoc qui relie les différentes unités portables (palmtop, PDA, notebook) peut être utilisé pour propager et partager les informations parmi les participants dans une conférence [MFP00].

Contrôle d'environnement : Des petits véhicules équipés de caméras et des détecteurs de son peuvent être utilisés dans une région déterminée. Afin de collecter un ensemble d'information et de l'envoyer à travers un réseau Ad Hoc à une station qui traite ces informations. On peut avec cette méthode, par exemple, prévoir la pollution de l'eau [ERC99, MFP00] .

Utilisation privée : Les réseaux Ad Hoc peuvent être utilisés dans les Home Networks où les différents équipements échangent des informations (son, vidéo, alarme). Une des applications qui est dans ce contexte est: un réseau de robots dans une maison qui font plusieurs travaux (nettoyage, assurance de la sécurité) [MFP00] .

3.3 Les caractéristiques des réseaux ad hoc

Les réseaux mobiles ad hoc sont principalement caractérisés par ce qui suit :

Une topologie dynamique : Les unités mobiles du réseau, se déplacent d'une façon libre et arbitraire. Par conséquent, la topologie du réseau peut changer à des instants imprévisibles, d'une manière rapide et aléatoire.

Des contraintes d'énergie : Comme on l'a vu dans le chapitre précédent, Les hôtes mobiles sont alimentés par des ressources d'énergie autonomes très limité (batteries). Contrairement aux réseaux cellulaires, il n'y a pas d'infrastructure fixe qui peut profiter d'une ressource d'énergie illimité, et le rôle de stations de base et de tous les infrastructures fixes doit être prise en charge par les pauvres unités mobile elles-mêmes. Ceci peut amener fréquemment, à des déconnexions et des problèmes causés par l'écoulements des batteries. Le paramètre d'énergie est primordiale, et il doit être pris en considération dans toutes les applications, et les contrôles effectuées par le système.

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

Une sécurité physique limitée : Les réseaux mobiles ad hoc sont plus touchés par le paramètre de sécurité, et ils sont plus menacés par les attaques que les réseaux filaires classiques. Cela se justifie par les contraintes et limitations physiques qui font que le contrôle des données transférées doit être minimisé, et protégé.

L'absence d'infrastructure : Les réseaux ad hoc se distinguent des autres réseaux mobiles par la propriété d'absence d'infrastructure préexistante et de tout genre d'administration centralisée. Les hôtes mobiles sont responsables d'établir et de maintenir la connectivité du réseau d'une manière continue. Ceci représente un très grand challenge.

3.4 Le problème de routage dans les réseaux ad hoc

3.4.1 Définition

Généralement, le routage est une méthode d'acheminement des informations à la bonne destination à travers un réseau de connexion donné. Le problème de routage consiste pour un réseau dont les arcs, les nœuds et les capacités sur les arcs sont fixés à déterminer un acheminement optimal des paquets à travers le réseau au sens d'un certain critère de performance. Le problème consiste à trouver l'investissement de moindre coût en capacités nominales et de réserves qui assure le routage du trafic nominal et garantit sa survivabilité en cas de panne de lien ou de nœud.

Par exemple si on suppose que les coûts des liens sont identiques, le chemin indiqué dans la figure suivante est le chemin optimal reliant la station source et la station destination. Une bonne stratégie de routage utilise ce chemin dans le transfert des données entre les deux stations [LEM00].

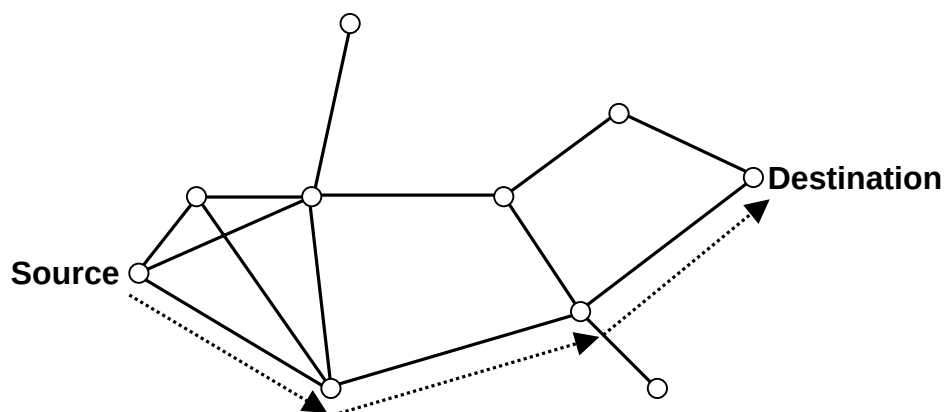


Figure 3.3 : Le chemin utilisé dans le routage entre la source et la destination.

3.4.2 La difficulté du routage dans les réseaux ad hoc

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

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

Chaque nœud participe donc à un protocole de routage qui lui permet de découvrir les chemins existants, afin d'atteindre les autres nœuds du réseau. Le fait que la taille d'un réseau ad hoc peut être énorme, souligne que la gestion de routage de l'environnement doit

être complètement différente des approches utilisées dans le routage classique. Le problème qui se pose dans le contexte des réseaux ad hoc est l'adaptation de la méthode d'acheminement utilisée avec le grand nombre d'unités existant dans un environnement caractérisé par de modestes capacités de calcul, d'énergie et de sauvegarde. Et d'une topologie dynamique.

Dans le cas où le nœud destination se trouve dans la portée de communication du nœud source, le routage devient évident et aucun protocole de routage n'est initié. Malheureusement, ce cas est généralement rare dans les réseaux ad hoc. Une station source peut avoir besoin de transférer des données à une autre station qui ne se trouve pas dans sa portée de communication, par exemple dans le réseau illustré par la figure 1.3.4.2 l'unité mobile W n'est pas dans la portée de communication de l'unité U (indiquée par le cercle d'origine U) et vice versa. Dans le cas où l'unité U veut transférer des paquets à W, elle doit utiliser les services de l'unité V dans l'envoi des paquets, puisque l'unité V contient dans sa portée de communication les unités U et W.

Le problème de routage est plus compliqué à cause de la non-uniformité de la transmission sans fil et de la possibilité du déplacement imprévisible de tous les nœuds concernés par le routage, et de ressources d'énergies limitées.

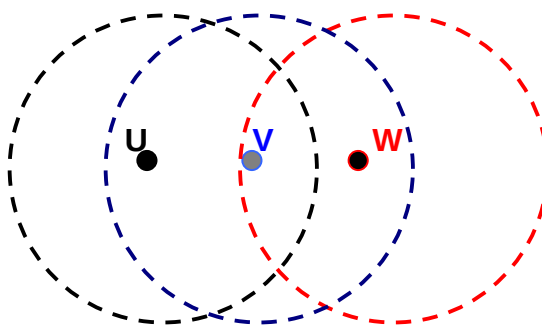


Figure 2.4 : Un simple réseau ad hoc constitué de trois unités mobiles

3.5 classification des protocoles de routage destinés aux réseaux mobiles Ad Hoc

La stratégie (ou le protocole) de routage est utilisée dans le but de découvrir les chemins qui existent entre les nœuds. Le but principal d'une telle stratégie est l'établissement de routes qui soient correctes et efficaces entre une paire quelconque d'unités, ce qui assure l'échange des messages d'une manière continue.

Vu les limitations des réseaux ad hoc, la construction des routes doit être faite avec un minimum de contrôle, et de consommation de la bande passante, du temps de calcul et surtout d'énergie. Suivant la manière de création et de maintenance de routes lors de l'acheminement des données, les protocoles de routage peuvent être séparés en deux catégories, les *protocoles pro-actifs* (*TABLE DRIVEN*) et les *protocoles réactifs* (*ON DEMAND*). Les protocoles pro-actifs établissent les routes à l'avance en se basant sur

l'échange périodique des tables de routage et/ou des informations de topologie, alors que les protocoles réactifs cherchent les routes à la demande.

3.5.1 Notions générales utilisé par les protocole de routage

3.5.1.1 Multihop: Les stratégies de routage utilisées dans les réseaux ad hoc sont caractérisées par le fait de pouvoir acheminer les paquets de données sans l'aide des stations de base utilisées dans la communication cellulaire.

Dans le modèle cellulaire, la communication entre deux nœuds est faite en utilisant les stations de base et le réseau filaire. Par conséquent, aucune unité mobile n'est utilisée comme routeur intermédiaire, le modèle cellulaire est dit alors "*single hop*" (i.e. le nombre de routeurs mobiles intermédiaires est nul). La contrepartie de ce modèle est le modèle de communication sans infrastructure. Dans ce modèle plusieurs nœuds peuvent participer au routage c'est pour cela que l'environnement des réseaux ad hoc est dit "*multihop*" (i.e. le nombre de stations mobiles qui peuvent être utilisées comme routeurs intermédiaires peut dépasser le un).

3.5.1.2 GPS : Le GPS (Global Positioning System) est un système de localisation composé de 24 satellites qui tournent autour de la terre à une hauteur de 10900 milles, rendent possible pour les utilisateurs muni de récepteurs terrestres de déterminer leur position géographique avec une précision de 10 à 100 mètres. Les satellites utilisent des calculs mathématiques simples pour diffuser des informations qui seront traduites en coordonnées géographiques par les récepteurs de base terrestre[COM03].

3.5.1.3 Groupe : Une des architectures qui est utilisée dans les réseaux Ad Hoc est la décomposition de ces réseaux en un ensemble de groupes. Chaque groupe désigne un nœud qui sera son représentant. Les représentants de groupe assurent le routage des données et contrôle l'utilisation du canal par leurs membres de groupe, ils sont prioritaires d'accéder au canal et donc ils ont plus de chance de transmettre que d'autres nœuds.

3.5.1.4 Hiérarchie : On peut étendre la notion de groupe vue précédemment pour former des niveaux hiérarchiques de groupes. Les représentant de groupes dans un niveau i , deviennent des membres de groupes dans un niveau $i+1$, ces nouveaux membres s'organisent en un ensemble de groupes de la même manière que le niveau bas, et ainsi de suite jusqu'à arriver à un niveau où il n'existe qu'un seul représentant de groupe.

3.5.1.5 source routing : c'est une technique de routage dans laquelle l'émetteur du paquet (source) détermine la séquence complète de nœuds à travers laquelle le paquet doit passer. L'émetteur met explicitement le chemin dans l'entête en identifiant chaque saut (hop) par l'adresse du prochain nœud à qui on doit envoyer le paquet .

3.5.1.6 DAG (Direct Acyclic Graph): on dit qu'un graphe orienté est un DAG pour le sommet (destination) j si à partir de n'importe quel autre sommet $i \neq j$, il existe au moins un chemin vers j .

3.5.1.7 inversement de liens (link reversal): c'est une technique utilisée par les algorithmes de routage basée sur la construction de DAGs. Elle est utilisée pour

reconstruire un DAG quand un sommet perd son dernier chemin vers la destination à cause d'une défaillance de lien, ceci est réalisé par inversement d'un ensemble de liens.

3.5.2 Les protocoles de routage pro-actifs

Les protocoles de routage pro-actifs des réseaux mobiles ad-hoc, sont basés sur la même philosophie des protocoles de routage utilisés dans les réseaux filaires conventionnels. Pour cela, nous allons examiner les deux principales méthodes utilisées dans le routage des réseaux filaires.

Les deux principales méthodes utilisées sont : la méthode Etat de Lien ("Link State") et la méthode du Vecteur de Distance ("Distance Vector"). Les deux méthodes exigent une mise à jour périodique des données de routage qui doit être diffusée par les différents nœuds de routage du réseau.

Les algorithmes de routage basés sur ces deux méthodes, utilisent la même technique qui est la technique des plus courts chemins, et permettent à un hôte donné, de trouver le prochain hôte pour atteindre la destination en utilisant le trajet le plus court existant dans le réseau. Généralement le calcul du plus court chemin entre deux stations, est basé sur le nombre de nœuds (on dit aussi le nombre de sauts) que comportent les différents chemins qui existent entre les deux stations. Mais on peut aussi associer des coûts aux liens de communication; ces coûts seront utilisés en appliquant une fonction de calcul qui est en général linéaire. Un coût peut matérialiser le taux de l'utilisation d'un lien, les délais relatifs de communication ou un autre facteur qu'on veut minimiser lors du routage des données.

Dans le protocole "Link State", chaque nœud de routage maintient sa propre vision de la topologie du réseau et qui inclut l'état de ses canaux de sortie. Pour que cette vision soit à jour, chaque nœud diffuse (par inondation) périodiquement l'état des liens de ses voisins à tous les nœuds du réseau. Cela est fait aussi quand il y a un changement d'état de liens.

Un nœud qui reçoit les informations concernant l'état des liens, met à jour sa vision de la topologie du réseau, et applique un algorithme de calcul des chemins optimaux afin de choisir le nœud suivant pour une destination donnée. Un exemple des algorithmes les plus connus appliqués dans le calcul des plus courts chemins, est celui de Dijkstra. Notons que le nœud de routage calcule la plus courte distance qui le sépare d'une destination donnée, en se basant sur l'image complète du réseau formée des liens les plus récents de tous les nœuds de routage. Cela veut dire que pour qu'un nœud p puisse déterminer le nœud de routage suivant pour une destination donnée, p doit recevoir les messages de la dernière mise à jour des liens, propagé par le réseau. Le protocole OSPF (Open Shortest Path First), est l'un des protocoles les plus populaires basé sur le principe "Etat de lien". L'algorithme "Distance Vector" de base a été adopté pour le routage dans les réseaux ad hoc sans fil, et cela en traitant chaque hôte mobile comme un nœud de routage.

Dans l'approche de routage "Distance Vector", chaque nœud de routage diffuse à ses nœuds de routage voisins, sa vision des distances qui le séparent de tous les hôtes du réseau. En se basant sur les informations reçues depuis tous ses voisins, chaque nœud de routage fait un certain calcul pour trouver le chemin le plus court vers n'importe quelle destination. Le processus de calcul se répète, s'il y a un changement de la distance

minimale séparant deux nœuds, et cela jusqu'à ce que le réseau atteigne un état stable. Cette technique est basée sur l'algorithme distribué de Bellman-Ford (DBF).

L'algorithme DBF est basé sur l'utilisation des messages de mise à jour. Un message de mise à jour contient un vecteur d'une ou plusieurs entrées dont chaque entrée contient, au minimum, la distance vers une destination donnée. Le principe du DBF est utilisé par une grande partie des protocoles de routage des réseaux filaires. Un problème de performance majeur de cet algorithme est qu'il prend beaucoup de temps pour mettre à jour les tables de routage des hôtes, après une partition du réseau, des défaillances de nœuds, ou quand il y a un grand nombre de nœuds dans le réseau. D'autres problèmes du DBF sont dus à l'absence de coordination entre nœuds, dans les modifications des tables de routage qui peuvent être faites en se basant sur des données erronées (le problème de "routing loops"). En plus de cela, le DBF ne possède pas de mécanisme précis qui peut déterminer quand est ce que le réseau doit arrêter l'incrément de la distance qui correspond à une destination donnée, ce problème est appelé : "counting to infinity".

La circulation inutile des paquets de messages, qui peut arriver avec le DBF, est intolérable dans les réseaux mobiles ad hoc, caractérisés par une bande passante limitée et des ressources modestes. En plus de cela, la mobilité fréquente des nœuds fait que la convergence du DBF prend beaucoup de temps, ce qui pénalise le routage dans de tels environnements. Durant ces dernières années, beaucoup d'efforts ont été faits dans le but de résoudre les problèmes du DBF, et de l'adapter dans le contexte des réseaux mobiles. Un de ces efforts est le protocole DSDV (Destination Sequenced Distance Vector) [CPP94]

Dans les algorithmes de routage basés sur le principe "Link State", le problème counting to infinity n'existe pas. En plus de cela, la convergence d'un nœud de routage, est moins lente par rapport au DBF, ce qui a fait que "Link State" est plus préféré et utilisé dans beaucoup de réseaux modernes. Cependant, l'approche du "Link State" exige que chaque nœud, doit maintenir une version mise à jour de la topologie complète du réseau, ce qui nécessite un grand espace de stockage et implique une surcharge d'échange de paquets de contrôle dans le cas des réseaux dynamiques. En outre, aucun algorithme, implémenté en se basant sur le principe "Link State", n'a pu éliminer - totalement - la création des boucles temporaires de routage.

Les protocoles de routage pro-actifs rassemblent les idées des deux approches précédentes, et essaient de les adapter pour les environnements mobiles en essayant de réduire ou d'éliminer leurs limitations tout en prenant en considération, les caractéristiques du nouvel environnement, voici quelques protocoles proactifs :

Protocole	Principes
DSDV (Destination Sequenced Distance Vector) [PER94]	Distance Vector
WRP (Wireless Routing Protocol) [MUR96]	Distance vector
GSR (Global State Routing) [CHE98]	Link State + principes de DSDV
FSR (Fisheye State Routing) [PEI00]	Amélioration de GSR (en ajoutant la technique d'œil de poisson)
CGSR (Cluster-Gateway Switching Routing) [CHI97]	Distance Vector + groupes
HSR (Hierarchical State Routing) [IWA99]	Hiérarchie
ZHLS (Zone –based Hierarchical Link State) [JOA 99]	Link State + Hiérarchie
DREAM Distance Routing Effect Algorithm for Mobility) [BAS98]	Localisation (GPS)

Figure 2.5 quelque protocoles proactif

Une représentation un peu plus détaillé peut être trouver dans [LEM00], ou dans [DDB01]. Nous avons remarqué, par simulation que les protocoles proactif, ne sont pas adaptatif à la mobilité, et ils sont souvent moins performants que les réactifs, et ils consomment toujours plus d'énergie que les réactifs [DDB01].

3.5.3 Les protocoles de routage réactifs

Comme nous l'avons vu dans la section précédente, les protocoles de routage pro-actifs essaient de maintenir les meilleurs chemins existants vers toutes les destinations possibles (qui peuvent représenter l'ensemble de tous les nœuds du réseau) au niveau de chaque nœud du réseau. Les routes sont sauvegardées même si elles ne sont pas utilisées. La sauvegarde permanente des chemins de routage est assurée par un échange continue des messages de mise à jour des chemins. Ce qui induit un contrôle excessif surtout dans le cas des réseaux de grande taille.

Les protocoles de routage réactifs (dits aussi : protocoles de routage à la demande (On Demand Routing)), représentent les protocoles les plus récents proposés dans le but d'assurer le service du routage dans les réseaux sans fil. La majorité des solutions proposées pour résoudre le problème de routage dans les réseaux ad hoc, et qui sont évaluées actuellement par le groupe de travail MANET (Mobile Ad Hoc Networking Working Groupe) de l'IETF (Internet Engineering Task Force), appartiennent à cette classe de protocoles de routage.

Les protocoles de routage appartenants à cette catégorie, créent et maintiennent les routes selon les besoins. Lorsque le réseau a besoin d'une route, une procédure de découverte globale de routes est lancée, et cela dans le but d'obtenir une information spécifiée, inconnue au préalable. Plusieurs approches peuvent être appliquées dans la découverte des routes. La majorité des algorithmes utilisés sont basés sur le mécanisme d'apprentissage en arrière (backward learning). Le nœud source, qui est à la recherche d'un chemin vers la

destination, diffuse par inondation une requête dans le réseau. Lors de la réception de la requête, les nœuds intermédiaires (ou de transit) essaient de faire *apprendre* le chemin au nœud source. Une fois que la destination est atteinte, elle peut envoyer une réponse en utilisant le chemin tracé par la requête, un chemin est alors établi entre le nœud source et le nœud destination. Si les lien sont bidirectionnel, le chemin est full duplex. Le travail peut être réduit, dans le cas où un nœud de transit posséderait déjà un chemin vers la destination. Une fois le chemin est calculé, il doit être sauvegardé et mis à jour, tant qu'il est en cours d'utilisation.

Dans la plus part des protocole réactive, et comme est le cas pour les proactifs. Un nœud qui reçoit un paquet pour le router, consulte sa table pour déterminer le prochain saut. Cependant, dans les protocole basé sur la technique "SOURCE ROUTING", ou routage de source, décrite précédent, un nœud qui reçoit un paquet pour le router n'a pas consulter l'entête du paquet pour déterminer le prochain saut. Ce mécanisme est utiliser par le protocole DSR, qu'on va voir dans les chapitre qui suit.

Le routage à la demande induit une lenteur à cause de la recherche des chemins, ce qui peut dégrader les performances des applications interactives (exemple: les applications des bases de données distribuées), ce qui justifie leur absence dans les réseaux filaire. Mais ces protocoles sont plus adaptatif pour les réseaux ad hoc que les protocoles de l'autres classe [DDB01].

Voici quelque protocoles réactifs, pour plus de détaille consultez [LEM00, DDB01].

Protocole	Principes
DSR (Dynamic Source Routing) [JDM96]	Source Routing
AODV (Ad hoc On-Demande Distance Vectore) [PER99]	Combinaison entre DSR et DSDV
CBRP (Cluster Based Routing Protocol) [JIA99]	Groupe
LMR (Lightweight Mobil Routing) [COR95]	DAG + Link Reversal
TORA (Temporally Order Routing Algorithm) [PAR97]	DAG + Link Reversal + GPS, (dérivé de LMR)
ABR (Associativity Based Routing) [TOH96]	Associativité (historique des lien)
SSR (Signal Stabilité Routing) [DUB97]	Puissance des signaux des liens
LAR (Located Aided Routing) [KOV98]	DSR + GPS(pour localiser)
RDMA (Relative Distance Micro-discovery Ad hoc Routing) [AGG99]	Micro découverte + distance relatif

Figure 2.5 quelque protocoles réactif

A decorative border with a wavy, undulating top and bottom edge, enclosing the central text box.

Chapitre 3

*Techniques et métriques proposées pour
minimiser la consommation d'énergie*

I. Introduction

Les unités mobiles sont alimentées par des sources d'énergie limitées, comme vu précédemment, il est alors nécessaire de minimiser leur consommation d'énergie.

L'énergie consommée par l'interface réseau est la plus importante, surtout pour les petites unités [FOR94, DDB01].

Les premières recherches sur la minimisation de l'énergie consommée par les unités mobiles étaient au niveau matériel (HARDWARE), et visaient les réseaux structurés autour d'infrastructures fixe (réseaux cellulaires), qui n'ont pas le problème de limitation de source d'énergie.

Cependant, un gain important d'énergie peut être obtenu au niveau logiciel, et plus particulièrement au niveau du protocole de routage pour les réseaux ad hoc, là où les paquets suivent des chemins multi-saut (MULTI-HOP). Les principaux axes de recherches actuels se concentrent sur la conception des applications et protocoles de la couche transport, MAC et surtout de routages, fixant à minimiser l'énergie consommée. Pour assurer la plus longue durée de vie aux batteries des terminaux, et éviter ainsi leurs déconnexions et le partitionnement du réseau.

Comme indiqué dans [BTS01, DDB02], les protocoles réactifs sont plus efficaces en consommation d'énergie que les protocoles proactifs. Donc, l'approche réactive (passive) est meilleure pour la conception d'un protocole de routage efficace en consommation d'énergie (ENERGY EFFICIENT ROUTING).

Dans ce chapitre on va voir d'abord quelques mécanismes proposés pour réduire l'énergie consommée par les terminaux mobiles dans les réseaux ad hoc, ensuite on parlera des caractéristiques désirables d'un protocole de routage de consommation d'énergie minimale, et on terminera par la présentation des métriques proposées pour être utilisées par ce type de protocoles.

II. Mécanismes pour réduire la consommation d'énergie

1 Contrôle de la puissance de transmission

Dans tous les protocoles de routage standards, aussi bien proactifs que réactifs (vus au chapitre 2), les paquets sont transmis en utilisant une puissance maximale fixe, pour couvrir tout le domaine de puissance. En fait les paquets de données ne sont généralement pas propagés, mais plutôt destinés à un voisin unique (soit pour le router, soit par ce qu'il est sa destination finale), qui se trouve à une distance inférieure au domaine de puissance. Au lieu d'envoyer sous cette puissance, il est très intéressant d'utiliser une puissance dynamique qui dépend de la distance entre les nœuds.

Modèle d'énergie

Pour bien comprendre le gain d'énergie provenant de l'utilisation de cette technique, il faut d'abord présenter un modèle de calcul d'énergie de transmission, nous présentons celui proposé dans [SDTa02, LFM01]:

L'énergie consommée pour transmettre un paquet de données de taille D Octets via un lien donné, peut être modélisé par la formule simplifiée suivante :

$$E(D) = K_1 D + K_2$$

Tel que :

$$K_1 = P_t^{\text{packet}} * 8 / BR$$
$$K_2 = (P_t^{\text{MAC}} * D^{\text{MAC}} + P_t^{\text{packet}} * D^{\text{header}}) * 8 / BR$$

BR : le débit

P_t^{MAC} : la puissance à laquelle les paquets de la couche MAC (de contrôle) sont envoyés

D^{MAC} : la taille (en Octets) des paquets de la couche MAC

P_t^{packet} : la puissance à laquelle les paquets de données sont envoyés

D^{header} : la taille (en Octets) des entêtes des paquets de données

Les protocoles de routage standards utilisent la valeur maximale de P_t^{packet} , pour couvrir tout le domaine de puissance, ainsi l'énergie maximale ne dépend que de la taille du paquet:

$$E_{\text{max}}(D) = K_{1\text{max}} D + K_{2\text{max}}$$

Tel que $K_{1\text{max}}$ et $K_{2\text{max}}$ sont respectivement obtenus à partir des expressions de K_1 et K_2 en remplaçant P_t^{packet} par sa valeur maximale.

La puissance de transmission minimale P_t requise pour une réception correcte est:

$$P_t(d) = P_r * d^n / K$$

Tel que :

d est la distance, et K est une constante

n : est un paramètre d'atténuation, il prend pratiquement la valeur de 4.

P_r : est une puissance seuil (puissance minimale pour une bonne réception)

L'énergie consommée en utilisant cette puissance minimale est:

$$E_{\text{min}}(D,d) = K_{1\text{min}} D + K_{2\text{min}}$$

Tel que $K_{1\text{min}}$ et $K_{2\text{min}}$ sont respectivement obtenus à partir des expressions de K_1 et K_2 en remplaçant P_t^{packet} par $P_t(d)$

Théoriquement, si on utilise cette puissance minimale le gain d'énergie pour la transmission d'un paquet de taille D à une distance d , par rapport à l'utilisation de puissance maximale est :

$$S(D,d) = E_{\text{max}}(D) - E_{\text{min}}(D,d)$$

La figure 3.1 inspirée de [SDBb02] montre le gain d'énergie par rapport à la distance, on remarque que plus les nœuds se rapprochent (la distance est petite), plus le gain est important.

Puisque cette courbe n'est pas linéaire, l'utilisation des chemins à plusieurs saut (plus longs), peut économiser l'énergie par rapport à l'utilisation des chemins les plus courts. Pour bien comprendre ce phénomène on va présenter un exemple.

Soit trois nœuds alignés a , b et c , et supposons que c est à la portée de communication de a , mais qu'il est plus loin de a que b (b est entre a et c), le protocole classique (qui choisit les

plus courts chemins) envoie les paquets directement de a à c, si le chemin choisi est a-b-c, alors l'énergie requise pour envoyer un paquet de taille D est :

$$E_{\text{multi}}(D,d,d1) = E(D,d1) + E(D,d-d1)$$

Tel que d1 est la distance entre a et b, et d celle entre a et c et $E(D,d1)$, $E(D,d-d1)$ sont respectivement l'énergie d'envoi du paquet de a à b et de b à c.

Si $E(D,d)$ est l'énergie d'envoi du paquet directement de a à c, alors le gain d'utilisation du chemin le plus long $S(D,d,d1)$ est :

$$S(D,d,d1) = E(D,d) - E_{\text{multi}}(D,d,d1)$$

$$\text{c-à-d : } S(D,d,d1) = E(D,d) - E(D,d1) - E(D,d-d1)$$

En fait l'énergie économisée obtenue en utilisant plusieurs sauts (MULTI-HOP) dépend des distances d et d1. La figure 3.2 inspiré de [SDBb02] montre le pourcentage d'énergie économisée par rapport à d1, pour différentes valeurs de d.

On remarque que, pour les grandes valeurs de d, le gain est important, et il diminue avec la diminution de d. On remarque aussi que le gain diminue quand b s'approche de a ou de c, et peut même devenir négative (c à d que le chemin le plus court a-c consomme moins d'énergie que a-b-c), par exemple : dans les cas : $d1 > 0.5$ pour $d = 150$, et $d1 > 0.8$ pour $d = 200$).

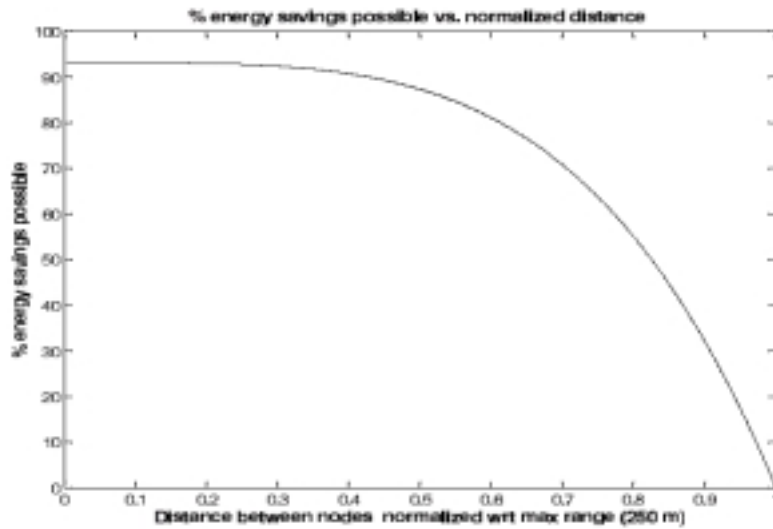


Figure 3.1 gain d'énergie par rapport à la distance [SDBb02]

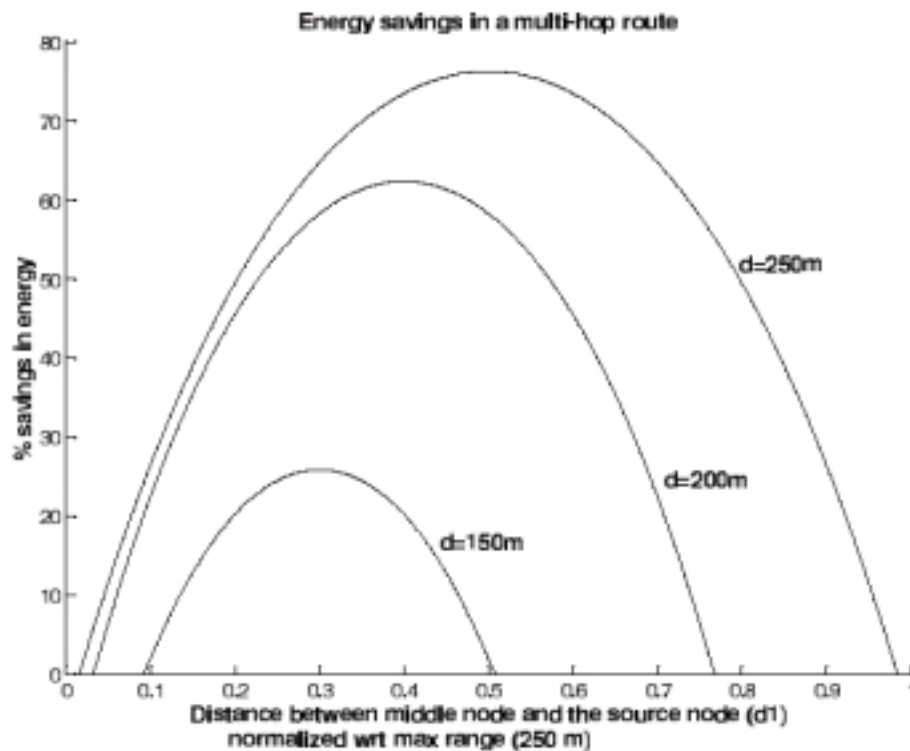


Figure 3.2 énergie économisé par l'utilisation de plusieurs sauts vs distances [SDBb02]

2 Eteindre l'interface réseau

Dans les protocoles MAC standards (tels que IEEE 802.11), les interfaces réseaux sont toujours allumées, et même si elles ne sont entrain ni de transmettre ni de recevoir des données, elles seront mise en mode IDLE. Les résultats des expériences réalisées par Jean-Pierre Ebert, Brian Burns et Adam Wolisz de l'université technique de Berlin, montrent que la consommation d'énergie dans ce mode est importante et très proche du mode de réception [JBA02].

Dans la figure 3.3, la transmission du nœud A au nœud B est entendue par hasard (OVERHEARD) par le nœud C, qui est un voisin de A. Le nœud C va perdre son énergie en recevant un paquet qui n'est pas destiné à lui, ou en restant en mode IDLE durant la période de transmission. Dans ce cas, il est clair que le nœud C doit être éteint durant cette transmission pour conserver son énergie. La grande question qui se pose est : quand faut-il rallumer l'interface ?, autrement dit, quelle est la durée pour laquelle l'interface restera éteinte ?

Ce mécanisme peut être assuré par le protocole de la couche MAC, et c'est la stratégie utilisée par ce protocole qui peut répondre à cette question.

Dans [SSM98] on propose un protocole dit PAMAS (Power-Aware Multiple Access Protocol with Signaling).

Dans ce protocole, comme dans tous les protocoles MAC des réseaux ad hoc, les nœuds vérifient le canal en échangeant les messages RTS/CTS (Ready To Send et Clear To Send).

Ainsi l'émetteur envoie un message RTS, et le récepteur répond par un message CTS si il reçoit correctement RTS, et par la suite l'émetteur commence la transmission

Dans PARMAS, cet échange de messages RTS/CTS prend place dans des canaux spéciaux de telle sorte à ne pas effectuer les transmissions de données en cours. Et ces messages contiennent la taille du paquet à émettre par l'émetteur, de cette façon n'importe quel nœud dans le voisinage peut utiliser cette information. Encore un autre problème reste posé, c'est quand un nœud s'éteint, s'allume pendant une nouvelle transmission de données (il ne capte pas les messages RTS/CTS de cette transmission). Dans ce cas, le nœud doit être capable d'estimer la durée de transmission restante.

Une présentation détaillée de ce protocole est disponible dans [SSM98].

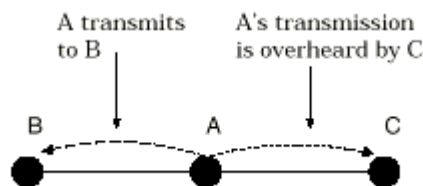


Figure 3.3 consommation inutile

3 Utilisation des antennes dirigées (DIRECTIONAL ANTENNAS)

Les protocoles et algorithmes proposés supposent que les nœuds sont équipés d'antennes omnidirectionnelles, c'est à dire que les nœuds ont des angles de couverture de 360°. Alors ils n'ont pas besoin de pointer leurs antennes pour communiquer. Bien que cette méthode est très simple, elle cause un gaspillage important d'énergie, puisque l'énergie est propagée vers toutes les directions. Il est donc avantageux d'utiliser des antennes dirigées pour économiser l'énergie.

Le gain d'énergie des antennes dirigées par rapport aux omnidirectionnelles dépend des angles (d'élévation et de largeur) utilisés. Plus l'angle est petit, plus le gain est important, ce gain dépend aussi de la suppression de lobe secondaire [CAB97].

Si on suppose que l'énergie dégagée dans lobe secondaire est négligeable c'est à dire que toute l'énergie est propagée dans le lobe principal, et que toute l'énergie consommée par l'antenne est convertie en radiation (l'efficacité de l'antenne est 100%). Alors le gain d'énergie est : [ASCR2002]

$$\text{Gain} = 4\pi / (\theta * \phi)$$

Où θ et ϕ sont respectivement les angles d'élévation et de largeur (AZIMUTH) en radian.

Si l'émetteur et le récepteur utilisent, tous les deux, des antennes dirigées, le gain total d'énergie est alors :

$$\text{Gain} = \text{Gain(TX)} * \text{Gain (RX)}$$

Où Gain (TX) et Gain (RX) sont donnés par la formule précédente.

Bien que ce gain est dans le cas idéal, et qu'en pratique les contraintes posées ne sont pas satisfaisable, ce qui fait diminuer le gain d'énergie. L'utilisation des antennes dirigées permet aux nœuds d'un réseau ad hoc de communiquer en consommant beaucoup moins

d'énergie par rapport à l'utilisation des antennes omnidirectionnelles. En plus, cette technique permet d'assurer plus de sécurité, et particulièrement de confidentialité, de données, en diminuant la probabilité de détection (de sniffer).

Mais tout ceci nécessite un protocole (qui réside normalement au niveau MAC) pour diriger les antennes, et synchroniser les utilisateurs. L'exécution de ce protocole cause une consommation d'énergie supplémentaire, ce qui cause un compromis avec le gain.

Dans [ASC02] on propose des solutions basées sur cette technique.

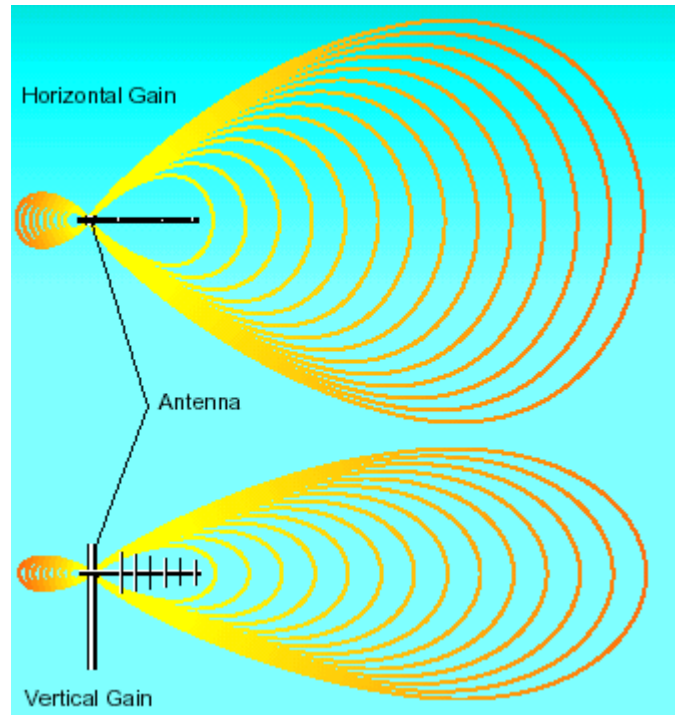


Figure 3.4 gain des antennes dirigées

4 Utilisation d'un protocole de routage efficace

Dans les premiers protocoles de routage construits pour les réseaux ad hoc, les chemins sont choisis en se basant sur la minimisation du nombre de sauts (HOPS), ou de délais. De cette façon un ensemble limité de nœuds peut être utilisé dans plusieurs routes (OVERUSED), ce qui cause une perte importante d'énergie, et ainsi une déconnexion rapide à ces nœuds.

Beaucoup d'études théoriques ont montré que l'énergie consommée pour le routage des paquets de données dans les réseaux ad hoc peut être énormément réduite, par rapport aux protocoles classiques (MIN-HOP ROUTING). Mais en pratique ce but est très difficile à réaliser. En fait même sans tenir compte de la limitation en sources d'énergie, la mobilité posait, et pose toujours un grand problème aux protocoles de routage. Nous avons montré dans [DDB03] que l'augmentation de la mobilité cause une forte dégradation de performances des protocoles de routage, et que la conception d'un protocole de routage qui

s'adapte rapidement au changement fréquent de la topologie causé par la mobilité, est un grand challenge.

Maintenant si on tient compte du facteur d'énergie, le problème de routage devient encore plus compliqué. Le but est non seulement de trouver des chemins stables et résistants à la mobilité, mais aussi de router les paquets, d'une part via des nœuds qui ont suffisamment d'énergie, et d'autre part via des chemins qui cause le minimum de consommation d'énergie.

Les travaux actuels ajoutent des modifications aux protocoles classiques, en proposant des nouvelles techniques et critères pour la découverte et le choix des chemins, ce qui permet de créer de nouvelles versions de protocoles existants nommés (POWER-AWARE ROUTING, MINIMUM ENERGY ROUTING, ou aussi ENERGY EFFICIENCY ROUTING).

Il est à noter que la première technique (contrôle de la puissance de transmission) peut être utilisé pour la conception d'un tel un protocole de routage.

III. Caractéristiques d'un protocole de routage efficace en consommation d'énergie

Dans ce paragraphe, nous allons reprendre les caractéristiques souhaitées d'un protocole de routage avec une consommation minimale d'énergie (MINIMUM ENERGY ROUTING) et présentées dans [SDTa02] :

1 Calcul du coût d'un lien en terme d'énergie et contrôle de la puissance de transmission de paquets de données

Le protocole de routage doit prendre en considération le coût, en énergie, d'un chemin pour choisir une route parmi plusieurs autres disponibles.

Comme on a vu précédemment, La puissance de transmission minimale requise pour une réception correcte dépend de la distance entre les deux nœuds (émetteur et récepteur), ainsi que l'énergie minimale nécessaire.

Le protocole de routage doit associer à chaque lien un coup représentant l'énergie $E(\text{lien})$ (ou la puissance) nécessaire, et choisir les routes en minimisant le coûts du chemin, qui est le cumule de coups des liens formants le chemin : formellement parlé :

$$E_{\text{route}} = \sum_{\text{lien} \in \text{chemin}} E(\text{lien})$$

Le protocole choisit le chemin qui minimise E_{route} , et transmet les données en utilisant des puissances adaptées (qui s'approche le plus de la puissance minimale), au lieu d'utiliser des puissances maximales fixes. En d'autre terme, il doit implanter le mécanisme de variation (contrôle) de la puissance de transmission (voir paragraphe 2.1).

2 Permettre de prendre en compte toutes les routes possibles, pour découvrir la route à énergie minimale

Pour comprendre cette caractéristique, prenons un exemple:

soit quatre nœuds a, b, c, d alignés, les nœud b et c sont à la portée de communication du nœud a (voir la figure 3.5). Pour trouver une route du nœud a au nœud d, le nœud a initialise une découverte de route en propageant un paquet de requête, ce paquet sera reçu par les nœuds b et c, ces derniers continuent la propagation. Tous les mécanismes de découverte de routes des protocoles (réactif) actuels se ressemblent du fait que chaque nœud ne propage pas plus d'une fois une même requête. Dans notre cas, puisque le nœud c a déjà reçu la requête, le paquet diffusé par b et reçu par c, sera ignoré, ce qui empêche la découverte du chemin a-b-c-d, et à la fin de la phase de découverte de route, les chemins trouvés sont : a-b-d et a-c-d. Bien qu'en terme de nombre de sauts, ils sont plus optimal que a-b-c-d, ce dernier peut être plus optimale en consommation d'énergie. Donc le protocole de routage doit permettre de prendre toutes les routes (ou le maximum de route) possibles en considération, y compris les routes longues qui incluent d'autres plus courtes, pour découvrir la route optimale en consommation d'énergie.

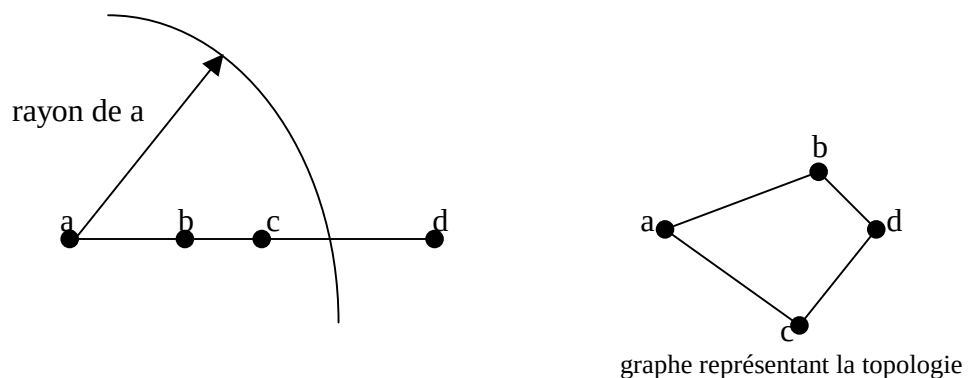


Figure 3.5 Topologie du réseau

3 Suivre la trace de coûts des liens

Dans les versions courantes des protocoles réactifs, la maintenance de route est effectuée après la défaillance de liens, et aucune maintenance n'est réservée au changement de qualité des liens. En plus aucun mécanisme utilisé ne permet de mettre à jour le changement de puissances de transmission nécessaire qui se produit par la mobilité de nœuds.

Même après la découverte d'une route optimale en consommation d'énergie, le changement de coûts des liens doit être suivi afin de consommer toujours le minimum d'énergie possible.

Dans le cas où le coût d'un certain lien augmente (à cause de l'éloignement des nœuds formant le lien), le chemin peut ne pas devenir optimal. Alors ce changement doit être communiqué au nœud source, pour que ce dernier puisse choisir un autre chemin s'il le faut.

Alors le protocole doit posséder un mécanisme pour suivre la trace de changements des coûts des liens, dans les routes utilisées.

La mobilité peut aussi causer la création des nouvelles routes plus optimaux que celles utilisées, par exemple: à un instant donné, le chemin optimal entre deux nœuds a et c est a-c. Après un moment donné (Δt), et à cause de la mobilité, un autre nœud, soit b, se place entre les deux nœuds précédents (voir figure 3.6), et le chemin a-b-c sera plus optimale que a-c. Alors une maintenance supplémentaire qui permet la découverte de nouvelles routes est préférable.

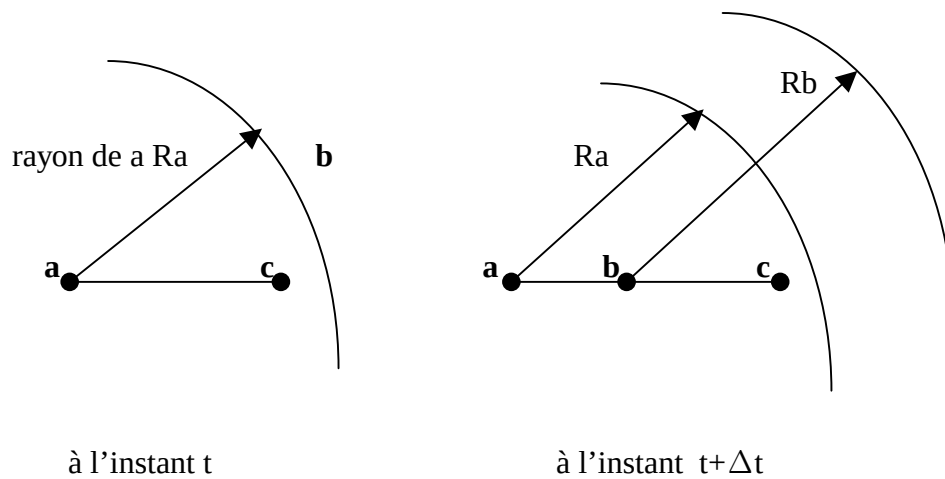


Figure 3.6

4 S'adapter à l'augmentation du nombre de nœuds (SCALING WITH NODES NUMBER)

Les messages de contrôle (OVERHEAD) nécessaires pour assurer le bon fonctionnement du protocole de routage, doivent s'adapter à l'augmentation (SCALE) du nombre de nœuds. En fait, l'augmentation du nombre de nœuds (n) dans le réseau implique l'augmentation du nombre de sauts dans les chemins utilisés par le protocole, ainsi que le nombre de messages de contrôle utilisés pour la découverte et la maintenance des routes. Le nombre de messages de contrôle doit être proportionnel au nombre de nœuds dans le réseau ($O(n)$) pour assurer l'adaptation (THE SCALING).

IV. Métrique d'un protocole de routage de consommation d'énergie minimale

La conservation de l'énergie et le partage prudent du coût énergétique entre les nœuds pour le routage des paquets, augmente la durée de vie des nœuds et du réseau. Les métriques utilisées par les protocoles classiques ne permettent pas d'atteindre cet objectif.

Dans ce paragraphe on va présenter les métriques proposées dans [SSM98], nommées (POWER-AWARE METRICS), et qui sont proposées pour être utilisées pour concevoir un protocole de routage efficace en consommation d'énergie. Ces métriques sont :

1 Minimiser l'énergie consommée par paquet

C'est la métrique la plus évidente et intuitive, qui reflète la conservation d'énergie.

Supposons qu'un paquet j traverse les nœuds $n_1, \dots, n_k$, tels que n_1 est le nœud source, et n_k la destination. Et soit $T(a,b)$ l'énergie consommée pour transférer (transmettre et recevoir) un paquet entre les nœuds a et b , alors l'énergie consommée pour le routage du paquet j est :

$$e_j = \sum_{i=1}^{k-1} T(n_i, n_{i+1})$$

le but de cette métrique est de minimiser e_j , $\forall$ le paquet j

Discussion : il est clair que cette métrique minimise l'énergie moyenne consommée par un paquet.

En fait il faut noter que, dans le cas d'une charge légère de données, les routes sélectionnées par cette métrique seront identiques à celles sélectionnées par les protocoles classiques (SHORTEST-HOP ROUTING). Ceci n'est pas surprenant, car dans ce cas, l'énergie pour transférer un paquet via un lien est pratiquement constante, si on suppose que $\forall (a,b) \in E$, où E est l'ensemble de liens : $T(a,b) = T$ (une constante), alors l'énergie consommée (e_j) est $(k-1) * T$.

Minimiser cette valeur, revient simplement à minimiser k , ce qui est équivalent à trouver le chemin le plus court. Mais dans le cas où un ou plusieurs nœuds sont plus ou moins chargés, le coût de transfert d'un paquet à travers un saut ne sera plus constant, et les routes sélectionnées par cette métrique vont être différentes des routes les plus courtes.

Un inconvénient majeur de cette métrique est, qu'un sous-ensemble de nœuds peut être plus utilisé dans les chemins choisis. Ceci peut causer une différence remarquable entre les consommations des nœuds, et alors des déconnexions rapides pour ces nœuds.

Considérons le réseau de la figure 3.7, et supposons que le réseau n'est pas chargé. Le nœud 6 sera sélectionné pour router les paquets de 0 à 3, de 1 à 4 et de 2 à 5, par conséquent ce nœud consommera sa batterie plus rapidement que les autres nœuds. Ainsi cette métrique n'assure pas la plus longue vie à tous les nœuds, et au réseau.

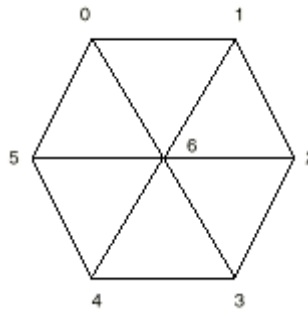


Figure 3.7 [SSM98]

2 Maximiser le temps avant le partitionnement du réseau

Cette métrique est très importante pour certains types d'application, l'objectif est d'éviter le partitionnement le plus longtemps possible.

Discussion : pour une topologie donnée, en utilisant le théorème MAX-FLOW-MIN-CUT, on peut trouver un ensemble minimal de nœuds (le CUT-SET), tel que la suppression de l'un d'eux cause un partitionnement du réseau. Les routes entre ces partitions doivent contenir un de ces nœuds critiques, le protocole doit alors diviser la tâche entre ces nœuds pour maximiser la durée de vie du réseau. Ce problème est similaire à celui de l'équilibre de charge (LOAD BALANCING), où les tâches sont à envoyer à un des serveurs disponibles ; le choix du serveur doit minimiser le délai, ce problème est NP-complet.

En fait si la taille des paquets est constante, on peut résoudre le problème (pour une topologie fixe) par la sélection des nœuds en appliquant la politique ROUND-ROBIN, mais en pratique cette condition n'est pas vérifiée. En plus la mobilité rend le problème beaucoup plus compliqué.

3 minimiser la variance d'énergie entre les nœuds

Cette métrique suppose que tous les nœuds du réseau ont la même importance, et aucun nœud ne doit être plus pénalisé qu'un autre. De cette façon, les nœuds vont demeurer opérationnels ensemble, le plus longtemps possible.

Discussion : ce problème est similaire à celui de partage de charge (LOAD SHARING) dans les systèmes distribués. Dont l'objectif est de minimiser le temps de réponse tout en gardant le travail incomplet le même dans tous les nœuds.

Atteindre cette optimisation est un problème très compliqué à cause des temps d'exécution inconnus des tâches futures. Allouer un ensemble de N tâches de durée variables à trois machines (ou plus), est problème NP-complet.

Une politique utile qui peut être utilisée pour s'approcher du but est celle nommée rejoindre la plus petite file (JOIN THE SHORTEST QUEUE (JSQ)). On peut faire adopter cette idée à un protocole de routage, ainsi le voisin (parmi ceux possédant une route vers la destination désirée) qui a la plus petite quantité de données en attente de transmission, sera

choisi par le protocole pour router les paquets de données. On peut améliorer cette idée en vérifiant le trafic en attente au niveau d'un certain nombre de nœuds formant les chemins, pour choisir le meilleur chemin.

La variation de la charge entre les nœuds, et la mobilité de ces derniers, rendent l'implémentation de cette politique un problème très compliqué.

Notons que la minimisation de la variance d'énergie entre les nœuds peut maximiser (ou au moins augmenter considérablement) le temps avant le partitionnement du réseau.

4 Minimiser le coût par paquet

On a vu que la métrique définie dans 4.1 (minimiser la consommation d'un paquet), peut pénaliser un sous-ensemble de nœuds. Si le but est de maximiser la durée de vie de tous les nœuds dans le réseau, une autre métrique doit être utilisée, de telle sorte que les nœuds qui ont épuisé leur réserve d'énergie ne participant pas dans plusieurs chemins.

Soit $F_i(X_i)$ la fonction qui dénote le coût (le poids) du nœud i , tel que X_i représente l'énergie totale consacrée par le nœud i . On définit le coût total de l'envoi d'un paquet à travers un chemin par la somme des coûts des nœuds formant le chemin. Formellement, le coût d'envoi d'un paquet j du nœud n_1 au nœud n_k via les nœuds intermédiaires $n_2, \dots, n_{k-1}$ est :

$$c_j = \sum_{i=1}^{k-1} F_i(X_i)$$

le but de cette métrique est de minimiser c_j , $\forall$ le paquet j

Discussion: intuitivement, F_i représente la répugnance d'un nœud pour router les paquets. On peut remarquer qu'avec un choix convenable de F_i , on peut atteindre plusieurs objectifs. par conséquent, si F_i est une fonction monotone croissante, alors les nœuds (comme par exemple le nœuds 6 de la figure 3.7) ne seront pas trop utilisés (OVERUSED), ce qui augmente leurs durées de vie. F_i peut être aussi adaptée pour qu'elle reflète fidèlement la durée de vie restante de la batterie.

Beaucoup de batteries ont une courbe de décharge similaire à celle de la figure 3.8 [SGO97]. Sur cette courbe on mesure le voltage de la batterie par rapport au taux d'énergie consommée, on remarque que : quand 80% de la capacité de la batterie est consommée, le voltage est 3.6, et que pour un voltage de 2.8 V la batterie est complètement déchargée. Un choix intéressant de F_i est :

$$F_i(X_i) = 1 / (1 - g(z_i))$$

Où z_i dénote le voltage mesuré (qui donne une bonne indication de l'énergie utilisée) et $g(z_i)$ est le taux d'énergie consommée, en fait ($g(z_i)$, z_i) est un point de la courbe de décharge.

Pour cet exemple on peut donner l'expression alternative de F_i suivante:

$$F_i(X_i) = 1 / (z_i - 2.8)$$

Cette fonction a un coût de nœud raisonnable pendant 80% de la durée de vie de la batterie (quand le voltage varie entre 4 V et 3.6 V), mais après, le coût augmente rapidement.

Intuitivement, cette forme de F_i assure que, les chemins les plus courts soient utilisés au début (quant les batteries sont neuves), mais quand les batteries commencent à perdre leurs

capacités, les chemins sont choisis plus prudemment, de tel sorte d'éviter qu'un nœud épuise sa batterie avant les autres.

Enfin on note que les courbes de décharges de quelques batteries alcalines sont presque linéaires, on peut alors les associer des fonctions de coût linéaires, de la forme :

$F_i(z_i) = c * z_i$. Ou c est un paramètre à déterminer.

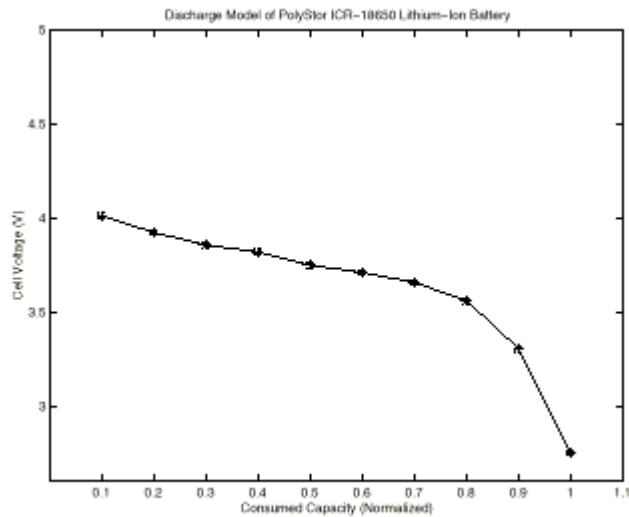


Figure 3.8 Modèle de Décharge de la batterie PolyStar ICR_18650 Lithium_Ion [SSM98]

5 Minimiser le coût maximal

Soit $C_i(t)$ le coût de routage d'un paquet via le nœud i à l'instant t , et $\hat{C}(t)$ le maximum des $C_i(t)$.

Une définition alternative, est de minimiser le coût maximal après le routage de N paquets à leurs destinations, ou après T secondes, toutes ces variances assurent l'équilibre de consommation d'énergie entre les nœuds. Malheureusement, aucun mécanisme n'est actuellement proposé pour implanter cette rigoureuse métrique, mais la minimisation du coût par nœud (la quatrième métrique) peut réduire d'une manière significative le coût maximal.

Les cinq métriques présentées, peuvent conserver l'énergie utilisée pour router les paquets, mais comment implémenter ces métriques?

Les protocoles classiques (qui calculent les chemins les plus courts) peuvent être utiles, il suffit d'ajouter des poids aux nœuds (ou aux liens), et calculer ainsi le chemin optimal (en se basant sur la minimisation de la somme des poids et non pas de la longueur du chemin), et de cette façon, les métriques 1 et 4 peuvent être implémentées.

Ces métriques (ou plus tôt une d'elles) sont utilisées pour envoyer les paquets qui n'ont aucun privilège particulier, quant au paquet qui sont estimés plus prioritaires (et qui sont de nombres limités), ils peuvent être routés via les plus courts chemins.

Nous notons qu'aucune des métrique proposé, ne tire profit de la technique de contrôle de puissance. Et que tous les métrique (excepté la quatrième), sont trop générale, et nécessite des définitions plus formelle pour être implémenté.



Chapitre 4

*Proposition de nouvelles métriques et
d'un protocole de routage*

I. Introduction

Les études récentes montrent que le protocole de routage à consommation d'énergie minimale peut être désigné à partir des protocoles classiques existants, en ajoutant de nouveaux mécanismes et techniques, et que l'approche réactive est plus adaptative que celle proactive.

Comme on l'a vu dans la partie précédente, un des mécanismes prometteurs proposés pour réduire la consommation d'énergie, est l'utilisation des fréquences d'amplification adaptatives. Dans [SDTb02], Sheetal Kumar Doshi et al proposent une implémentation de ce mécanisme. En fait, ils ont modifié le protocole de routage DSR en lui ajoutant principalement ce mécanisme, pour concevoir des versions consommant moins d'énergie par rapport au DSR standard.

Suresh Singh et al proposent dans [SSM98] des métriques générales sur lesquelles un protocole de routage peut se baser afin de réduire la consommation d'énergie. Entre autre, on trouve la métrique qui définit la notion de répugnance des nœuds pour router les paquets, selon l'état de leurs batteries.

D'une part, les métriques [SSM98] ne prennent pas en compte la possibilité d'utiliser des puissances d'amplification dynamiques, et ne tirent pas profit de la technique de contrôle de puissances. Et d'autre part, tous les protocoles proposés dans [SDTb02] (versions de DSR), ne prennent pas en considération les états des batteries des nœuds, et les chemins sont choisis en se basant sur la minimisation de l'énergie requise sur les chemins disponibles pour le transfert de chaque paquet, sans tenir compte des états énergétiques des nœuds formant le chemin, ni les états des canaux de communication.

Le choix des chemins qui nécessitent le moins d'énergie pour router les paquets n'est pas toujours optimal. Car ce choix n'empêche pas l'utilisation d'un sous-ensemble de nœuds d'une manière abusive par rapport à d'autres. Par conséquent, ces nœuds vont perdre leur charges avant les autres. De cette façon, le risque de partitionnement du réseau augmente, ce qui diminue la durée de vie du réseau.

Cependant, pour assurer la plus longue durée de vie au réseau, il faut distribuer la tâche de routage sur l'ensemble des nœuds, et il faut aussi prendre en compte les états des batteries des nœuds formant les chemins.

Dans cette partie, nous allons définir de nouvelles métriques qui tentent de résoudre le compromis qui existe entre : minimisation d'énergie pour router chaque paquet, en choisissant les chemins qui nécessitent le moins d'énergie d'une part, et assurer la plus longue durée de vie à tous les nœuds du réseaux, en choisissant les chemins contenant les nœuds possédant les batteries les plus fraîches, d'autre part. Nous allons aussi présenter une nouvelle technique qui permet d'exploiter toutes les routes disponibles et distribuer ainsi la charge sur plusieurs chemins. En se basant sur toutes ces nouvelles métriques et techniques, nous allons ensuite modifier le protocole de routage DSR, afin de l'améliorer et de concevoir un protocole conscient sur la consommation d'énergie (ENERGY AWARE).

Ce chapitre comporte les parties suivantes :

Après cette introduction, la deuxième partie est une présentation de DSR. Dans la troisième nous allons présenter une implémentation du mécanisme permettant d'économiser l'énergie. Dans le paragraphe qui suit nous allons présenter nos nouvelles métriques que nous proposons suivies de notre nouvelle technique round robin sur les chemins. Et enfin, nous terminerons par la présentation du protocole.

II. Présentation générale du protocole DSR (Dynamic Source Routing)

Comme son nom l'indique, le protocole DSR [JDM96, PER01]. est basé sur la notion de routage de source. Ainsi, pour envoyer un paquet, le nœud source construit le chemin complet à suivre nommé *source route*. Il met les adresses des nœuds intermédiaires formant ce chemin dans l'entête du paquet. Par la suite, il envoie le paquet via son interface sans fil au premier nœud (*saut*) qui figure sur ce chemin. Quand ce dernier reçoit le paquet, et si il n'est pas sa destination finale, il n'a qu'à l'envoyer au prochain saut figurant sur l'entête, et ainsi de suite.

Chaque nœud qui participe à l'exécution de l'algorithme, maintient une *cache de route*, là où il sauvegarde les routes qu'il a apprises. Quand un nœud source veut envoyer un paquet à un autre nœud destinataire, il commence par vérifier s'il possède déjà un chemin dans son cache, vers ce dernier. Si c'est le cas, le nœud source peut utiliser ce chemin ou choisir le plus **optimal en nombre de sauts** (si il y en a plusieurs). Sinon, il doit tenter de trouver un chemin, en initiant une *découverte de route*, comme le font tous les protocoles réactifs.

Pendant cette période de découverte de route, le nœud peut continuer à traiter et échanger les paquets avec d'autres nœuds. Il peut aussi bien sauvegarder le paquet pour le transmettre quand un chemin sera trouvé une fois la découverte de route terminée, que compter sur les protocoles des couches supérieures (application et transport) pour le renvoyer.

A chaque chemin dans le cache de route, on associe un délai de garde *TIME OUT*, qui représente la durée maximale pendant laquelle ce chemin peut être considéré valide.

Cependant, avant cette période, le chemin peut devenir invalide, à cause de la mobilité ou la déconnexion d'un ou de plusieurs nœuds formant le chemin, causant ainsi une *défaillance de lien*. Quand un nœud détecte une défaillance d'un lien au cours de son utilisation, il initialise une *maintenance de route*. Pour remplacer le chemin invalide par un autre valide.

Dans ce qui suit, nous allons présenter avec un peu plus de détail le protocole DSR proposé dans [JDM96] et [PER01].

1. Découverte de route

La découverte de route permet à n'importe quel nœud du réseau de découvrir un chemin (ou plusieurs) vers un autre nœud du réseaux. Que ce soit franchissable directement par une simple diffusion si il est à sa portée de communication (*voisin*), où bien via un ensemble de nœuds intermédiaires.

Le nœud qui initialise une découverte de route, diffuse un *paquet de requête (ROUTE REQUEST)*, à tous ces *voisins*. Chaque requête est identifiée par : l'initiateur de la découverte, le destinataire (*TARGET*), et un numéro de séquence unique (*REQUEST ID*) géré par l'initiateur. Le paquet de requête contient aussi une séquence d'adresses représentant les nœuds intermédiaires par lesquels le paquet a passé, notons cette séquence, qui est initialement vide, par *record*. Pour détecter la duplication des requêtes reçues, chaque nœud maintient une liste des paires (*adresse de l'initiateur (initadr)*, *identité de*

requête(reqid)) des requêtes reçues récemment, soit l'ensemble recent_req. Quand un nœud *i* reçoit un paquet de requête, il le traite en appliquant l'algorithme suivant:

- a. si la paire (initadr, reqid) $\in$ recent_req alors ne pas traiter le paquet et arrêter sa propagation.
- b. Sinon, si *i* $\in$ record, alors ne pas traiter le paquet et arrêter sa propagation.
- c. Sinon, si *i* est la destination de la requête, alors la séquence record contient le chemin menant de la source à la destination. *i* envoie alors un paquet de réponse (REPLY) au nœud source.
- d. Sinon, ajouter *i* a record et continuer la propagation de la requête.

De cette façon, la requête va être propagée sur le réseau jusqu'à atteindre la destination, qui va répondre en envoyant le paquet REPLY au nœud source. La deuxième condition assure que les chemins conçus ne contiennent pas de boucle. Quant à la première, elle permet d'éviter de propager plusieurs fois le même paquet, même si ce dernier **a suit un chemin différent**.

Pour envoyer le paquet de réponse, le nœud destinataire (source de la réponse) doit posséder un chemin vers le nœud source. Si il possède déjà un chemin dans sa cache, il n'a qu'à l'utiliser pour envoyer le paquet. Sinon, il peut inverser le chemin résultant de la découverte et l'utiliser, mais ceci nécessite que les chemins soient bidirectionnels, ce qui n'est pas toujours valable. Une alternative qui peut résoudre le problème dans le cas de réseaux hétérogènes, est d'attacher (PIGGYBACK) le paquet de réponse à un paquet de requête destiné au nœud source. Afin d'éviter une boucle de requête infinie.

2. Maintenance de route

Contrairement aux protocoles proactifs, DSR n'utilise aucun message périodique. Cependant, pour maintenir les routes, la procédure distribuée de maintenance contrôle les chemins qui sont au cours d'utilisation, et informe l'émetteur des erreurs possibles.

Plusieurs protocoles de la couche MAC (tel que IEEE 802.11), utilisent des accusés de réception de paquets. Dans ce cas, la maintenance peut être facilement conçue.

Quand un lien ne devient plus valide, le protocole MAC du nœud qui est entrain d'envoyer des données via ce lien, signale un problème de transmission. Par la suite, le nœud envoie un paquet d'erreur au nœud source du paquet de données qui a rencontré l'erreur. Le paquet d'erreur contient les identités des deux nœuds qui forment le lien défaillant. Le nœud qui reçoit le paquet d'erreur supprime de son cache tous les chemins contenant le lien en question.

Si le protocole de la couche inférieure ne permet pas de telle détection de défaillance de lien, le protocole peut utiliser ce qu'on appelle *acquiescement passif*, qui consiste à écouter la transmission du même paquet par le nœud suivant. Et pour s'assurer de la bonne réception sur le dernier lien, il faut suivre les acquiescements des couches supérieures. Mais ce mécanisme nécessite que l'interface soit opérationnelle en mode PROMISCUOUS.

La dernière alternative est d'utiliser des acquiescements au niveau du protocole de routage, soit pour chaque saut (HOP BY HOP), soit point à point (END TO END). L'absence d'acquiescement dans le premier cas implique une défaillance de lien, quant au deuxième, il ne permet au nœud source que de déduire la défaillance du chemin, sans savoir le lien qui a causé la défaillance.

3. Optimisation

Dans [JDM96], on propose aussi les optimisations suivantes:

3.1 utilisation complète du cache (FULL USE OF CACHE) : elle consiste à utiliser le cache pour répondre aux requêtes. Quand un nœud reçoit un paquet de requête, il vérifie si il possède une route pour la destination en question dans son cache. Si c'est le cas, il répond à cette requête et arrête la propagation.

Discussion : bien que cette option minimise le nombre de paquets de contrôle, elle peut augmenter le risque d'utilisation des routes invalides. Il faut donc que les durées de vie maximales des routes en cache soient bien choisies, afin d'éviter l'utilisation des routes invalides.

3.2 Attacher des données aux paquets de requêtes (PAGGYBACKING) : elle consiste à attacher (PAGGYBACK) les données aux paquets de requête, afin de minimiser les délais de transfert, et le nombre de paquets envoyés. Cependant la quantité d'informations attachées doit être limitée.

Discussion : les paquets de requête sont propagés, et la propagation des données sur le réseau peut dégrader les performances. Nous pensons que cette technique peut être utilisée pour envoyer des paquets de faibles tailles et surtout de hautes priorités.

3.3 Utiliser le mode PROMISCUSE et snifer sur les paquets (de contrôle et de données), pour mettre à jour les caches de route

Discussion : l'utilisation de ce mode dégrade la sécurité. En plus, il cause plus de temps CPU et de consommation d'énergie, et il empêche d'éteindre l'interface réseaux quand le nœud n'a pas de transfert de données à faire.

3.4 Utiliser un buffer pour sauvegarder les paquets en attente de réponses aux requêtes.

Discussion : ce principe est pratiquement indispensable pour assurer plus de fiabilité, et perdre le minimum de paquets. De cette, façons on minimise le nombre de paquets à retransmettre par les couches supérieurs.

3.5 Utiliser un TIME OUT avant de retransmettre une requête vers une même destination

Discussion : ce mécanisme est aussi indispensable, pour minimiser les paquets de contrôles. Mais la détermination de la valeur optimale de TIME OUT n'est pas évidente.

3.6 Optimisation de maintenance : Quand le nœud source reçoit un paquet d'erreur pour une route r , et si le paquet a suivi un chemin différent de r (les liens ne sont pas bidirectionnels), le nœud source envoie le paquet d'erreur sur la route r pour informer tous les nœuds formant le chemin de cette défaillance.

Discussion : si les chemins ne sont pas bidirectionnelles, cette technique peut être très utile pour éviter d'utiliser des chemins incorrectes.

3.7 Utiliser des caches d'informations négatives : quand un nœud reçoit, ou rencontre un paquet d'erreur, il enregistre l'information du lien en question pour éviter de l'utiliser ultérieurement.

Discussion : Cette technique peut être très utile pour éviter d'utiliser des chemins incorrectes, mais ces informations sont temporaires. Il faut alors les éliminer quand elles ne seraient plus valables. Sinon on risque d'empêcher l'utilisation des liens valables.

Plus de détails sur le protocole sont disponibles dans [JDM96] et même dans [PER01].

Ce protocole tel qu'il est défini, ne prend pas en considération la contrainte d'énergie. Aucune technique n'est utilisée pour minimiser la consommation d'énergie, ni pour choisir le meilleur chemin en consommation d'énergie.

Avant de présenter les modifications que nous proposons, nous commençons par présenter une implémentation du mécanisme de contrôle de puissance présenté précédemment, que nous allons utiliser.

III. Implémentation du mécanisme d'utilisation de puissances de transmission adaptatives

Dans les protocoles de routage standards, tels que DSR présenté précédemment, les paquets sont transmis en utilisant une puissance maximale fixe, pour couvrir tout le domaine de puissance. En fait, un paquet de données n'est pas propagé, mais plutôt destiné à un voisin unique (soit pour le router, soit parce qu'il est sa destination finale), qui se trouve à une distance inférieure au domaine de puissance. Au lieu d'émettre sous cette puissance, il est très intéressant d'utiliser une puissance dynamique qui dépend de la distance entre les nœuds, comme on l'a vu dans le chapitre précédent.

Dans [SDTb02], on trouve une implémentation de ce mécanisme de variation de puissance. En fait, dans cet article Sheetal Kumar Doshi et al proposent des modifications des protocoles existants (DSR et IEEE 802.11), pour concevoir des versions du protocole de routage DSR efficaces en consommation d'énergie.

Ils proposent de modifier l'entête du paquet de requête, pour qu'il contienne la puissance utilisée pour la transmission du paquet par l'émetteur, soit P_{tx} cette puissance. Supposant aussi que le récepteur reçoit le paquet à la puissance P_{rx} , et soit $P_{threshold}$ la puissance minimale requise pour une réception correcte, notons que cette puissance représente une caractéristique du hardware utilisé (voir chapitre 1). La puissance de transmission minimale pour une réception correcte P_{txmin} peut être calculée par le nœud récepteur, elle est donnée par :

$$P_{txmin} = P_{tx} + P_{threshold} - P_{rx}$$

Pour surmonter le problème de l'instabilité des liens due à la fluctuation des canaux et à la mobilité des nœuds, une marge M est ajoutée. Ainsi la valeur de P_{txmin} utilisée devient :

$$P_{txmin} = P_{tx} + P_{threshold} - P_{rx} + M$$

Cette puissance est calculée pour chaque saut, et incluse dans l'entête du paquet de requête durant la phase de découverte de route. Le nœud destinataire (ou celui qui répond à la requête) inclut toutes ces valeurs dans le paquet de réponse. Ainsi le nœud source sauvegarde le chemin avec les informations de puissances requises sur chaque saut. Pour

envoyer les paquets de données, le nœud source met dans l'entête du paquet le chemin à suivre en indiquant la puissance à utiliser pour chaque saut.

Tous les paquets à propager (les requêtes et les signaux de la couche MAC) sont envoyés en utilisant la puissance maximale (FULL POWER), pour s'assurer qu'ils seront reçus par tous les voisins après un seul envoi. Les autres paquets sont envoyés à des puissances contrôlées.

On suppose que la propagation du signal est la même pour tous les nœuds, et que les domaines de puissance des nœuds sont égaux. Dans le cas contraire, des informations supplémentaires peuvent être incluses dans les entêtes pour qu'ils soient utilisées dans les calculs.

Ainsi chaque route $I_0, I_1, \dots, I_{n-1}$ dans le cache peut être caractérisée par la métrique suivante :

$$métrique = \sum_{i=1}^{n-1} P_i$$

Où P_i est la puissance à utiliser pour envoyer les paquets sur le lien (I_i, I_{i+1}) .

Les chemins sont choisis de telle sorte à minimiser la valeur de *métrique*.

Discussion :

Bien que l'implémentation de cette technique montre un gain très important par rapport au DSR standards, le choix du chemin en se basant sur cette métrique n'est pas toujours optimale. Car il ne prend pas en considération les états des batteries des nœuds sur les chemins, et ceci risque d'épuiser les ressources d'un sous ensemble de nœuds.

L'exemple suivant montre l'inconvénient de cette métrique :

Exemple : soit le réseau statique de la figure suivante : où les poids des liens représentent les puissances à utiliser sur les liens :

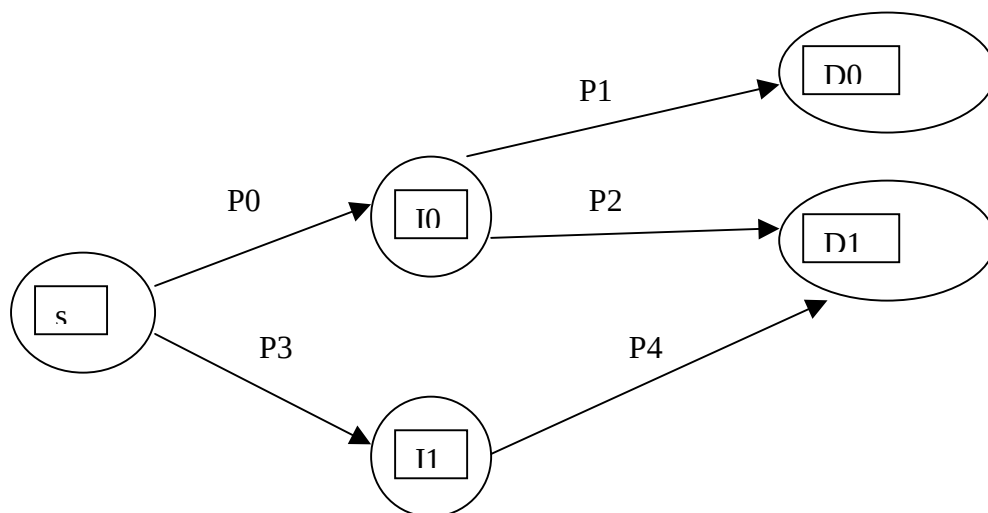


Figure 4.1

Supposons qu'il y a une session entre s et D0 et une autre entre s et d1, et que $P_0 + P_2 < P_3 + P_4$.

Si s choisit les routes en se basant sur la métrique présentée, I0 sera utilisé pour envoyer les paquets vers D0 et D1. Quant à I1, il ne sera pas exploité.

De cette façon, le nœud I0 risque de perdre son énergie plus rapidement que les autres, ce qui cause un partitionnement du réseau (D0 ne sera plus franchissable). Ce partitionnement rapide peut être évité si les ressources du nœud I1 sont exploitées.

IV. Métriques proposées

Afin de résoudre le compromis qui existe entre le choix du chemin qui nécessite le minimum d'énergie, d'une part, et celui qui contient les nœuds alimentés par les batteries les plus fraîches d'autre part. Nous proposons des nouvelles métriques, sur lesquelles se base le choix des chemins. Au début, supposons que le réseau est fixe et essayons de résoudre le problème d'une manière centralisée, à un instant t donné.

Commençons par définir la structure de données suivante, qui représente le réseau à l'instant t :

Soit (S,L,P,E) un quadruplet tel que :

S : est un ensemble de sommets

L : est un ensemble d'arcs, $L = \{(s_i,s_j) / s_i \in S \text{ et } s_j \in S\}$. En fait (S,L) est un graphe représentant la topologie du réseau à l'instant t .

P : une fonction qui associe à chaque élément (s_i,s_j) de L , une image dans R (ensemble des nombres réels), nommée poids de (s_i,s_j)

$$\begin{aligned} P : L &\rightarrow R \\ (s_i,s_j) &\rightarrow P(s_i,s_j) \end{aligned}$$

E : une fonction qui associe à chaque élément s de S , une image dans R (ensemble des nombres réels) nommée état de s .

$$\begin{aligned} E : S &\rightarrow R \\ s &\rightarrow P(s) \end{aligned}$$

Un exemple est montré sur la figure 4.2, il représente la structure associée au réseau de l'exemple précédent, (de la figure 4.1)

Sur un chemin donné $C = I_0, I_1, \dots, I_{n-1}$, nous définissons la métrique suivante:

$$M_C = \sum_{i=0}^{n-2} \alpha * E(I_i) + \beta * P(I_i, I_{i+1}) \dots \dots \dots (1)$$

le but est de choisir le chemin C qui a la métrique M_C minimale

où

α : représente le taux des états ($0 \leq \alpha \leq 1$)

β : représente le taux des lien ($0 \leq \beta \leq 1$)

tel que, $\alpha + \beta = 1$, donc $\beta = 1 - \alpha$

Il reste à définir les fonctions E et P et le paramètre α

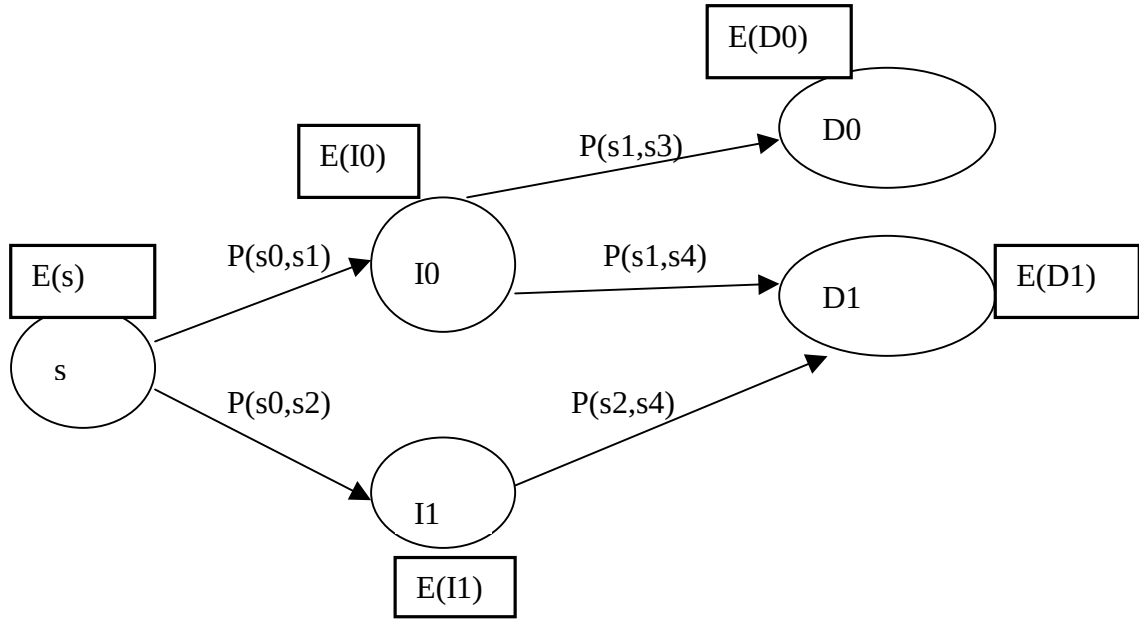


Figure 4.2

La fonction E doit refléter les états des batteries. Quant à P, elle doit refléter les coûts des puissances de transmission sur les liens.

Pour E, nous choisissons la fonction suivante :

$$E(s_i) = \frac{1}{1 - \text{energ}(s_i)} \dots\dots\dots(2)$$

ou $\text{energ}(s_i) \in [0,1]$ est le taux d'énergie consommée par le nœud s_i

notons que cette fonction est monotone sur l'intervalle $[0,1]$. Donc plus le taux de consommation du nœud augmente, plus sa valeur d'état augmente.

Pour P, nous proposons la fonction suivante :

$$P(s_i, s_j) = \frac{1}{1 - \text{PowRate}(s_i, s_j)} \dots\dots\dots(3)$$

$$\text{PowRate}(s_i, s_j) = \frac{\text{Pow}(s_i, s_j)}{\text{MaxPow}(s_i)}, \text{ PowRate}(s_i, s_j) \in [0,1]$$

où $\text{Pow}(s_i, s_j)$ est la puissance nécessaire sur le lien (s_i, s_j)

$\text{MaxPow}(s_i)$ est la puissance maximale (qui permet de recouvrir tout le domaine de puissance), du nœud s_i . A vrais dire, PowRate est le taux de la puissance requise par rapport à la puissance maximale. Cette fonction augmente avec l'augmentation de PowerRate

Plus α augmente, plus les chemins qui contiennent les batteries les plus chargées sont favorisés. Et plus elle diminue, plus les chemins minimisant le coût cumulatif par paquet sont favorisés. Ainsi nous proposons la politique suivante.

Proposition : Plus l'écart entre les états des batteries des nœuds augmente, plus il faut faire augmenter le paramètre α . Et plus les états se rapprochent, plus il faut le faire diminuer.

Définition de l'écart entre les états de batteries

Nous définissons l'écart entre les états de batteries à l'instant t par :

$$\text{EnergDiff} = \max_s(\text{energ}(si)) - \min_s(\text{energ}(si))$$

C'est la différence entre le taux d'énergie consommée maximal, et celui minimale, des nœuds du réseau.

Propriété :

$$\text{EnergDiff} \in [0,1]$$

Preuve :

On a d'une part :

$$\begin{aligned} \max(\text{energ}(si)) &\geq \min(\text{energ}(si)) \rightarrow \max(\text{energ}(si)) - \min(\text{energ}(si)) \geq 0 \\ &\rightarrow \text{EnergDiff} \geq 0 \dots\dots\dots(a) \end{aligned}$$

Et d'autre part : $0 \leq \max(\text{energ}(si)) \leq 1$ Et $0 \leq \min(\text{energ}(si)) \leq 1$ car c'est des taux

Donc :

$$\begin{aligned} \max(\text{energ}(si)) &\leq 1 \\ \text{et} \\ 0 &\leq \min(\text{energ}(si)) \end{aligned} \} \rightarrow \max(\text{energ}(si)) - \min(\text{energ}(si)) \leq 1 - 0 \rightarrow \text{EnergDiff} \leq 1 \dots(b)$$

de (a) et (b) on déduit que $\text{EnergDiff} \in [0,1]$

Il reste à trouver une fonction monotone croissante sur l'intervalle $[0,1]$, pour quelle représente le paramètre α

Nous proposons les deux fonctions suivantes:

$$F1 = \alpha(\text{EnergDiff}) = (1 - \alpha_0) * \text{EnergDiff} + \alpha_0 \dots\dots\dots(4)$$

$$F2 = \alpha(\text{EnergDiff}) = \sqrt{(1 - \alpha_0)^2 * \text{EnergDiff} + \alpha_0} \dots\dots\dots(5)$$

tel que α_0 est une valeur initiale

les deux fonctions sont monotones croissantes sur l'intervalle $[0,1]$. elles diffèrent dans leurs manières de croissance. La première est linéaire. Quant à la deuxième elle est racine, elle est plus croissante sur cette intervalle que la première (voir figure 4.3).

Donc plus l'écart énergétique augmente, plus le paramètre α augmente.

Mais de quelle façon, l'augmentation donne mieux de performance ? c'est pour cette raison que nous avons choisi ces deux types de fonctions différentes.

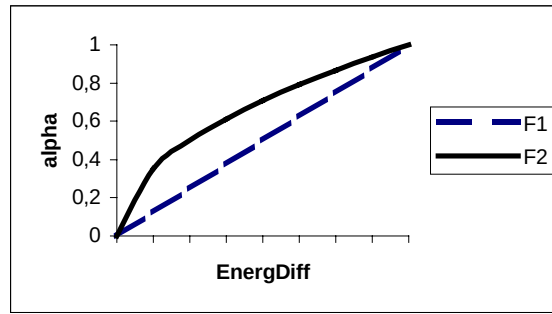


figure 4.3 Courbe des deux fonctions sur l'intervalle[0,1]

Récapitulation :

La métrique Opt_C qui représente l'optimalité d'un chemin $C = I_0, I_1, \dots, I_n$, reliant le nœud I_0 au nœud I_n est :

$$Opt_C = \sum_{i=0}^{n-1} \frac{\alpha}{1 - Energ(I_i)} + \frac{1 - \alpha}{1 - \frac{Pow(I_i, I_{i+1})}{MaxPow(I_i)}} \dots \dots \dots (6)$$

tel que

$\alpha = (1 - \alpha_0) * EnergDiff + \alpha_0$ ou bien $\alpha = \sqrt{(1 - \alpha_0)^2 * EnergDiff + \alpha_0}$ où α_0 est une constante

Le chemin optimale est celui qui possède Opt_C minimale.

A l'instant t : la valeur $pow(s_i, s_j)$ est connue pour chaque lien. $Energ(s_i)$ est connue pour chaque nœud s_i , donc $EnergDiff$ peut être calculée ainsi que α . Ceci permet de déterminer le chemin optimal (selon notre métrique) reliant n'importe quel couple de nœuds à l'instant t (en appliquant l'algorithme de Djikstra par exemple).

Avec le temps, les valeurs de $Energ(s_i)$ changent, ainsi que la valeur de α . Il faut alors recalculer les paramètres ainsi que les routes.

En utilisant cette approche, le nœud I_0 de l'exemple précédent, ne sera pas utilisé durant toutes les sessions entre le nœud s et les nœuds D_0 et D_1 . Car, quand le nœud I_0 commence à consommer son énergie pour router les paquets, $E(I_0)$ augmente. Alors, la route (s, I_0, D_1) devient moins optimale que (s, I_1, D_1) . Cette dernière va être utilisée pour router les paquets vers d_1 , et un partitionnement rapide sera évité.

Cette solution centralisée n'est utilisée que pour expliquer les métriques proposées. Dans la pratique, il est clair qu'aucun nœud ne peut avoir une vue globale exacte sur le réseau (topologie, états des batteries).

Mais il est possible d'appliquer ces métriques sur n'importe quelle algorithme de routage distribué (comme on va voir par la suite).

V. Politique round robin sur les chemins

Plusieurs protocoles de routage réactifs, à savoir DSR, sont multi-route. C'est à dire qu'après une découverte de route, le nœud source peut obtenir dans son cache plusieurs routes vers sa destination. Mais pour router les paquets, il utilise toujours le chemin optimal (en nombre de sauts). Les autres ne seront que des chemins alternatifs, utilisés en cas de défaillance du chemin optimal.

Pourquoi ne pas exploiter le maximum de chemins, et distribuer ainsi le trafic de données sur plusieurs nœuds ?

Pour ceci, nous proposons une politique analogue à une politique utilisée par les gestionnaires de processus dans les systèmes d'exploitations (scheduler). Les chemins sont analogues au processus et le protocole de routage au scheduler.

Soit nbr_paths chemins dans le cache du nœud S pour la destination D, ordonné selon leurs optimalités (le chemin 0 est le plus optimale)

Et soit $nbr_paquets$ le nombre de paquets à envoyer, rangés dans un buffer.

Pour exploiter le maximum de chemins, et pour que les chemins les plus optimaux soient plus utilisés que les autres, nous proposons de router sur chaque chemin i , le nombre de paquets suivant :

$$nb_paquets_i = \frac{2^{nbr_paths-i-1} * nbr_paquets}{2^{nbr_paths} - 1} \dots\dots\dots(7)$$

Ainsi, sur le premier chemin on envoi les $\frac{2^{nbr_paths-1} * nbr_paquets}{2^{nbr_paths} - 1}$ premiers

paquets, et sur le second on envoi les $\frac{2^{nbr_paths-2} * nbr_paquets}{2^{nbr_paths} - 1}$ suivants.

Et ainsi de suite, jusqu'à l'envoi de tous les paquets.

Théorème : Le nombre de paquets envoyés sur chaque chemin i , est deux fois plus grand que le nombre de paquets envoyés sur le chemin suivant $i+1$. Et tous les chemins sont nécessaires et suffisants pour envoyer les paquets.

Preuve : pour simplifier l'écriture notons: nbr_paths par n et $nbr_paquets$ np

- a) Le nombre de paquets envoyés sur chaque chemin i , est deux fois plus grand que le nombre de paquets envoyés sur le chemin suivant $i+1$

c à d qu'il faut montrer que :

$$\forall 0 \leq i \leq n-1 \quad nbr_paquets_i = 2 * nbr_paquets_{i+1}$$

$\forall 0 \leq i \leq n-1$ on a :

$$\begin{aligned} nbr_paquets_i &= 2^{n-i-1} * np / (2^n - 1) = 2^{(n-i-1)+1-1} * np / (2^n - 1) \\ &= 2^{1+(n-i-1)-1} * np / (2^n - 1) \\ &= 2 * 2^{(n-i-1)-1} * np / (2^n - 1) \\ &= 2 * 2^{n-(i+1)-1} * np / (2^n - 1) \end{aligned}$$

$$2 * 2^{n-(i+1)-1} * np / 2^{n-1} \underset{\text{definition}}{=} 2 * nbr_paquets_{i+1}$$

alors : $nbr_paquets_i = 2 * nbr_paquets_{i+1}$

b) tous les chemins sont nécessaires et suffisants pour envoyer les paquets

il faut montrer que la somme des paquets envoyés sur les chemins (en appliquant la formule), égale la taille (en nombre de paquets) du buffer.

Formellement parler :

$$\sum_{i=0}^{n-1} nbr_paquets_i = np$$

Lemme:

$$\forall 0 \leq i \leq n-1 \quad nbr_paquets_i = 2^{n-i-1} * nbr_paquets_{n-1}$$

Preuve:

$$nbr_paquets_{n-1} = 2^{n-(n-1)-1} * np / (2^n - 1) \text{ (par définition)}$$

$$nbr_paquets_{n-1} = np / (2^n - 1)$$

$$\text{par définition on a : } nbr_paquets_i = 2^{n-i-1} * np / (2^n - 1)$$

$$\text{donc : } nbr_paquets_i = 2^{n-i-1} * nbr_paquets_{n-1}$$

Ce lemme va être utilisé dans ce qui suit, pour la preuve du théorème

Suite de la preuve de théorème : On a :

$$\sum_{i=0}^{n-1} nbr_paquets_i = \sum_{i=0}^{n-1} nbr_paquets_i = SUM, \text{ en appliquant le lemme on aura :}$$

$$SUM = \sum_{i=0}^{n-1} 2^{n-i-1} * nbr_paquets_{n-1} = \sum_{i=0}^{n-1} 2^{n-1} * 2^{-i} * nbr_paquets_{n-1}$$

$$SUM = nbr_paquets_{n-1} + 2 * nbr_paquets_{n-1} + \dots + 2^{n-1} * nbr_paquets_{n-1}$$

C'est une série géométrique de base $nbr_paquets_{n-1}$, de raison 2, et de n termes. Cette somme égale [PIS80] :

$$nbr_paquets_{n-1} * \frac{1-2^n}{1-2} = nbr_paquets_{n-1} * (2^n - 1) = np / (2^n - 1) * (2^n - 1) = np$$

alors :

$$\sum_{i=0}^{n-1} nbr_paquets_i = np$$

Notons que l'expression précédente représentant le nombre de paquets à envoyer est une expression qui donne un nombre réel. Or, on ne peut pas parler de nombre de paquets rationnel. Le nombre de paquets à envoyer sur chaque chemin doit être plutôt un entier.

Pour résoudre ce petit problème, nous proposons de changer l'expression en:

$$[nb_paquets_i = \frac{2^{nbr_paths-i-1} * nbr_paquets}{2^{nbr_paths-1}}] \dots\dots\dots(8)$$

tel que $\lceil \cdot \rceil$ est la partie entière supérieure, définie de la manière suivante :

$\forall r \in \mathbb{R}, \lceil r \rceil = m \in \mathbb{N} / m \geq r \text{ et } m-1 < r$

de cette façon:

sur le premier chemin on envoie les $\lceil \frac{2^{nbr_paths-1} * nbr_paquets}{2^{nbr_paths-1}} \rceil$ premiers paquets, et sur

le second on envoie les $\lceil \frac{2^{nbr_paths-2} * nbr_paquets}{2^{nbr_paths-1}} \rceil$ suivants.

Et ainsi de suite, jusqu'à l'envoi de tous les paquets.

Ainsi la partie a) du théorème devient approximativement vrai (à une différence d'un paquet), formellement :

$$nbr_paquet_i \approx 2^{\pm 1} * nbr_paquet_{i+1}$$

Quant aux chemins disponibles. Ils restent suffisants pour envoyer les paquets, mais pas forcément nécessaires. Cela veut dire qu'on peut envoyer tous les paquets avant d'arriver au chemin n-1 (le moins optimale), ou plutôt sans exploiter tous les chemins.

Pour montrer cela (le nombre de chemins est suffisant) il suffit de montrer que:

$$\sum_{i=0}^{n-1} nbr_paquets_i \geq nbr_paquets$$

on a montré que

$$\sum_{i=0}^{nbr_paths-1} \frac{2^{nbr_paths-i-1} * nbr_paquets}{2^{nbr_paths-1}} = nbr_paquets$$

$\forall 0 \leq i \leq n-1$ on a par définition :

$$\lceil \frac{2^{nbr_paths-i-1} * nbr_paquets}{2^{nbr_paths-1}} \rceil \geq \frac{2^{nbr_paths-i-1} * nbr_paquets}{2^{nbr_paths-1}}$$

donc :

$$\begin{aligned} \sum_{i=0}^{nbr_paths-1} \lceil \frac{2^{nbr_paths-i-1} * nbr_paquets}{2^{nbr_paths-1}} \rceil &\geq nbr_paquets \\ \Rightarrow \sum_{i=0}^{n-1} nbr_paquets_i &\geq nbr_paquets \end{aligned}$$

Cette technique permet de distribuer les paquets sur le maximum de routes, en tenant compte de l'optimalité de ces routes.

Pour l'implémentation, notons que : $2^n = 2 \ll n$ (2 qui est $(10)_2$ décalé à gauche par n bits), cette opération peut être effectuée en une seule étape par n'importe quelle machine. Ceci peut être exploité pour simplifier la complexité de calcul par rapport à la fonction exponentielle qui nécessite n opérations pour calculer 2^n .

VI. Protocole proposé

6.1 Principales modifications

Notre protocole est basé sur DSR présenté précédemment, en lui ajoutant principalement les modifications suivantes:

- implémentation de la technique de contrôle de puissances présentée précédemment : Les paquets de données seraient envoyés en utilisant des puissances contrôlées sur chaque lien. Ces paquets sont calculés durant la phase de découverte de route comme dans [SDTb02]
- tenir compte des états énergétiques des batteries: pour ceci, nous supposons que chaque nœud peut déterminer à tout instant son taux d'énergie consommée. Pour ceci il suffit juste d'implémenter un mécanisme de bas niveau qui permet de fournir le voltage de la batterie, et la courbe de décharge[SSM98]. En plus, chaque nœud doit avoir une vue sur les états énergétiques des autres nœuds. Ceci est assuré en incluant l'information des états énergétique dans les paquets de contrôle, comme on va voir par la suite.
- les chemins sont classés par chaque nœud dans son cache, en se basant sur la métrique définie précédemment (de formule 6)
- ajout du paquet E_state: chaque nœud a une vue globale sur les états énergétiques des autres nœuds. Cette vue est mise à jour à chaque réception d'un paquet de réponse (RPLY). Pour que cette vue soit plus adéquate, nous proposons que chaque nœud diffuse à ses voisins un paquet de type E_state, après chaque changement d'un taux donné, là où il met sa valeur énergétique actuelle (taux de l'énergie consommée par rapport à la capacité de la batterie). Notons que la taille de ce paquet est très petite. Ce paquet est envoyé après chaque changement d'un taux donné. Si par exemple ce taux est 25%, chaque nœud envoie 3 paquets durant toute sa vie. Le premier quand il consomme 25% de sa batterie, le second quand il consomme 50%, et enfin le dernier quand il consomme 75%. Ce qui donne un coût supplémentaire, en paquets de contrôles propagés, de complexité $3*n$ durant toute la durée de vie du réseaux (ou n est le nombre de nœuds). D'une manière générale, si le taux est $1/x$ ($x \in \mathbb{N}$), le coût est $(x-1)*n$. Notons que le taux peut être changé au niveau de chaque nœud. Par exemple, il peut apparaître intéressant de le faire augmenter quand la charge de la batterie diminue. Car à ce moment, la validité de l'information sur l'état énergétique devient plus critique. Mais ceci nécessite plus de paquets de contrôle, et pose encore un compromis entre la fraîcheur de l'information, et la propagation des paquets de contrôles.
- modification des formats des paquets qui existent : pour implanter les mécanismes et métriques proposés, les formats des paquets ont été étendus pour contenir les informations supplémentaires nécessaires. Principalement la partie record des paquets

(contrôles et données), est étendue. Au lieu que chaque champ contient l'adresse IP d'un nœud i (saut) donné, il contient le triplet suivant ($@node_i$, P_{i+1} , E_i), tel que : $@node_i$ est l'adresse IP du nœud i , P_{i+1} est la puissance à utiliser sur le lien $i, i+1$ et E_i est l'état énergétique du nœud i . Il faut aussi ajouter un flag *LinkChange* au paquet REQ pour préciser si la requête est de type *LinkChange* ou pas. Et un flag *ALL_ROUTES* pour spécifier si toutes les routes sont à exploitées ou pas. Ainsi qu'un flag maintenance pour les paquets RPLY, et un champ *cost* (la nouvelle puissance nécessaire sur un lien) pour les paquets d'erreurs (à expliquer par la suite).

- Ajouter la possibilité d'exploiter toutes les routes possibles durant une découverte de routes. Ainsi, un nœud qui reçoit un paquet déjà reçu, peut continuer sa propagation, si le chemin suivi par ce paquet est différent de celui dans la copie rencontrée. Ceci dépend du flag *ALL_ROUTES* de la requête. Bien que cette technique permet théoriquement de découvrir tous les chemins possibles vers une destination, elle propage beaucoup de paquets de contrôle par rapport à la première, surtout quand la connectivité augmente. C'est pour cette raison que notre protocole n'utilise pas cette technique d'une manière absolue, mais ceci dépend d'un paramètre d'entrée.
- modification de la maintenance : si on utilise des puissances adaptatives aux distances, la mobilité peut rendre une puissance utilisée sur un lien insuffisante, sans que ce lien soit défaillant. Ceci en cas d'éloignement des deux nœuds formant le lien, l'un de l'autre. Dans ce cas, il faut ajouter un mécanisme permettant de découvrir la nouvelle puissance nécessaire. Nous proposons de procéder de la manière suivante : Quand un paquet ne passe pas sur un lien sélectionné, le protocole de la couche MAC du nœud émetteur le retourne au protocole de routage. Quand ce dernier le reçoit, il le renvoie en utilisant la puissance maximale *FULL_POWER*, et en le mettant dans un paquet de *requête non propagé (dirigé)* de type *LinkChanged*. Et met à jours ces variables afin d'éviter d'envoyer les paquets via ce lien. Si le voisin est toujours à la portée de communication de l'émetteur, il recevra cette requête, et répond par un paquet de réponse de type *maintenance*. A la réception de ce paquet, l'émetteur détermine la nouvelle puissance requise, et envoie un paquet d'erreur à tous les nœuds qui utilisent actuellement ce lien. Si le nœud successeur n'est plus à la portée de communication de l'émetteur, ce dernier reçoit à partir de sa couche MAC le paquet de données envoyé dans la requête dirigée. Et c'est à ce moment qu'il détecte la défaillance de lien.
- pour distribuer la tâche de routage sur le maximum de nœuds, et exploiter les chemins disponibles dans le cache. Les paquets sont acheminés en appliquant une politique ROUND ROBIN avec priorité sur les chemins.

6.2 Protocole

6.2.1 formats des paquets

Ici nous donnons des formats généraux des paquets, sans entrer dans des détails techniques. nous nous concentrons sur les champs utilisés dans l'algorithme présenté par la suite. En pratique (pour l'implantation), les formats détaillés respectent les normes IP [IET02].

1) paquet de données

Data	Header
------	--------

Tel que Data est la partie qui contient les données, et Header est l'entête du paquet. Il est composé des entêtes des différents protocoles utilisés, aussi bien des couches supérieures (application et transport), que la couche inférieure (MAC). Nous nous intéressons particulièrement à l'entête DSR, qui est ajouté à l'entête IP par derrière (PAGYBACKED) [IET02]. Il contient principalement :

Source : l'adresse de l'émetteur (le nœud source)

Destination : la destination finale du paquet

PacketId : un numéro de séquence géré localement par le nœud source

Record : ensemble de sauts formant le chemin à suivre. Chaque élément contient l'adresse du nœud, la puissance à utilisée à partir de ce nœud, et l'état énergétique de ce nœud. c à d $\text{Record} = \{(@\text{node}_i, P_{i+1}, E_i) / i, i+1 \in \text{chemin} \}$

Dans ce qui suit, nous notons $\text{record}[x]$ le triplet qui possède l'adresse x, et non pas le x'iem élément dans la liste.

2) paquet de requête REQ

Source	Destination	PacketId	Link changed Flag	ALL_ROUTES	Record
--------	-------------	----------	-------------------	------------	--------

Link changed Flag : Un flag pour préciser si la requête est pour une défaillance de lien, afin de réaliser une maintenance.

ALL_ROUTES: pour préciser si toutes les routes possibles sont à exploitées ou pas.

3) paquet de réponse RPLY

Sender	Destination	Source	PacketId	Maintenance Flag	record
--------	-------------	--------	----------	------------------	--------

Sender: Est l'adresse de l'émetteur du paquet

Destination: Est l'adresse de la destination de la requête pour laquelle ce paquet de réponse est envoyé

Source: Est l'adresse de la source de la requête pour laquelle ce paquet de réponse est envoyé. En fait c'est la destination de ce paquet de réponse.

Maintenance Flag: Pour préciser si la réponse est pour une requête de maintenance

4) paquet d'erreur

Source	Destination	BackId	UpId	PacketId	Cost	Record
--------	-------------	--------	------	----------	------	--------

Ce paquet est envoyé quand un lien échoue, ou quand la puissance utilisé sur un lien deviens insuffisante.

BackId : L'adresse du nœud prédécesseur du lien en question (pour lequel le paquet d'erreur est envoyé)

UpId : L'adresse du nœud successeur du lien en question

Cost : La nouvelle puissance nécessaire sur le lien. Si le lien devient défaillant, la valeur de ce champ est ∞

5) paquet de mise à jours d'état énergétique E_state

Source	Energy_state
--------	--------------

Energy_state : la valeur énergétique du nœud Source (celui qui propage le paquet)

6.2.2 constantes et variables

Chaque nœud I possède les constantes et variables suivantes :

REP_FROM_CACHE: booléen indiquant si les réponses à partir du cache sont activées.

FULL_POWER: valeur de la puissance de transmission maximale du nœud

EnergyStateThreshold: seuil à partir duquel, le changement d'état énergétique est considéré important

Pthreshold : puissance minimale nécessaire pour une réception correcte.

Pmargin : puissance marge. à ajouter pour surmonter une marge de mobilité

AllRoutes: booléen, indiquant si les requêtes seront propagées de telle sorte à découvrir tous les chemins

WaitForRoutes : délai utilisé après la réception d'une réponse. C'est pour attendre la découverte de plusieurs chemins, et exploiter la technique round robin. Ce délai doit être aussi petit que possible afin de ne pas influencer le délai de transfert des paquets

Id :numéro de séquence des paquets envoyés

Energ_state[] : chaque élément Energ_state[j] contient la vision de I sur l'état énergétique de j.

MinEnerg, Maxenerg, α : respectivement la valeur de energ_state[] minimale, la valeur de energ_state[] maximale, et le taux des états énergétiques (voir paragraphe précédent)

buffer[] : chaque élément buffer[D], est un buffer contenant les paquets de données à envoyer à D

sauv[] : chaque élément sauv[X] est un buffer contenant les paquets en attente de retransmission sur le lien menant à X. dans le cas où la puissance sur ce lien est insuffisante

route[] : la table de routage (cache). Chaque entrée route[D], contient une liste de routes ordonnée vers D. Chaque route contient principalement :

un champ metrique : contenant la valeur de la métrique de la route

Hops[] : le record (le chemin), formé des triplets décrits précédemment.

current_packet []: chaque élément current_packet[D] est un pointeur vers le paquet courant dans buffer[D], en attente de routage.

Req_sent{}: ensemble, dont chaque élément contient l'adresse d'un nœud à qui une requête est envoyée récemment

Recent_Req_seen {}: ensemble de tuplets, où chaque élément contient l'identification d'une requête envoyée récemment. Cette identité est soit le tuple (source, destination, identité du paquet, route) si le champ ALL_ROUTES de la requête est activé, ou bien (source, destination, identité du paquet), sinon.

Recent_Full_Send {}: ensemble de couples (Adresse, source), tel que

Adresse : est l'adresse d'un nœud, auquel un paquet a été envoyé en utilisant la puissance maximale. Ceci afin de découvrir la nouvelle puissance requise en cas où la puissance actuelle ne suffit pas.

source : est un nœud source qui utilise actuellement le lien menant au nœud Adresse.

Récent_Failed_links {}: ensemble des liens (nœuds successeurs) rompus récemment

6.2.3 évènements : chaque nœud I traite les évènements suivants :

Have a packet to send to D: cet événement est déclenché quand le nœud possède un (ou plusieurs) paquet de données à router au nœud D. si le nœud possède au moins un chemin vers D il peut router le paquet. Sinon, il le met dans le buffer en attendant la découverte d'un chemin, et lance une découverte de route si aucune requête n'a été envoyée récemment vers D.

receiving REQ: Quand un nœud reçoit une requête, il répond par la nouvelle puissance à utiliser si il s'agit d'une requête de type LinkChanged. Si non, il procède pratiquement de la même manière que dans DSR standard, avec quelques traitements supplémentaires pour l'implantation des mécanismes présentés.

When receive RPLY: Quand un nœud reçoit un paquet de type RPLY, et si il s'agit d'une réponse de type maintenance, il envoie (sauve) les paquets en attente en utilisant la nouvelle puissance (précisée dans l'entête de RPLY). Il envoie aussi un paquet d'erreur à chaque source d'un paquet sauvé, et met à jours son cache. Si non (si la réponse n'est pas de type maintenance), il procède pratiquement de la même manière que dans DSR standard, avec quelques traitements supplémentaires pour l'implantation des mécanismes présentés.

When receive a data packet to forward : cet événement est déclenché quand un nœud reçoit un paquet de données, dont il n'est pas sa destination finale. Si le lien menant au prochain saut précisé dans l'entente (la partie record) n'est pas en attente de maintenance (d'une mise à jours de puissance requise), et si le lien est toujours valide, il continue l'acheminement du paquet comme dans DSR standard. Si non il le met dans le buffer sauv, en attendant la maintenance du lien.

When receive LNKERROR (FailAdr,S,D,PacketId, Packet, type) message:

Cet événement est déclenché par un envoi de message par le protocole de la couche MAC (du même nœud I), quand le paquet Packet envoyé par le nœud S vers D ne passe pas sur le lien menant au nœud FailAdr. Si ce dernier appartient à l'ensemble de nœuds à qui I a envoyé une requête dirigée (maintenance de lien), et si le paquet retourné est de type requête (REQ), alors le nœud réagit à une détection d'une défaillance de lien. Si non (le paquet retourné est alors un paquet de données), il le sauvegarde dans le buffer sauv[FailAdr]. Si le paquet retourné n'appartient pas à l'ensemble de nœuds à qui I a envoyé une requête dirigée, il envoie un paquet de requête dirigé en lui attachant le paquet de données retourné. Momentanément, il considère que la puissance nécessaire sur ce lien est la puissance maximale (FULL_POWER), en attendant la découverte de la nouvelle puissance.

When receive ERROR

Le paquet d'erreur peut être envoyé à cause d'une défaillance de lien, ou même pour un changement sur la puissance requise sur un lien, selon la valeur du champ cost. Si la valeur est ∞ , alors il s'agit d'une défaillance. ainsi I réagit comme dans DSR standard. Sinon il met à jours son cache, et continue l'envoi du paquet si il n'est pas la destination finale.

When (Energy state change significantly)

Cet évènement aura lieu, chaque fois que l'état énergétique du nœud change d'une manière significative (voir au début de ce chapitre). Par conséquent, le nœud I diffuse un paquet E_state pour que les autres nœuds voisins mettent à jours leurs visions sur l'état de I.

When receive E_state

À la réception de ce message, le nœud met à jours ses paramètres utilisés dans les calculs des métriques, ainsi que sa table de routage.

6.2.4 l'algorithme

Chaque nœud I exécute l'algorithme distribué suivant :

When have packets to route to D

```

If (there is an entry in cache to D) route(D,0);
Else If(D  $\notin$  req_sent)
    Form a REQ packet; /*request packet*/
    REQ.Source = I;
    REQ.dest = D;
    REQ.Id= Id;
    If (AllRoutes) REQ.All_Routes = true;
    End if
    REQ.record = ((I, FULL_POWER,
                    energy_state[I]));
    broadcast REQ;
    Id ++;
    req_sent = req_sent  $\cup$  {D}
End if
Save packet in the buffer[D];
End if
End

```

When receiving REQ(J as source, D as destination, ID as PacketId, record) from K

```

If (REQ.LinkChanged = true
    If(I=D)
        Form a RPLY packet;
        Prec = ComputePowerRequired(REQ);
        RPLY.source=J;
        RPLY.replier=I;
        RPLY.dest=D;
        RPLY.record = (J, Prec,);
        RPLY.Maintenance = true;
        Send REPLY to K;
    End if
Else If (I  $\in$  record) /*loop!!!*/
    discard packet;

```

```

Break;
Else If (REQ.ALL_ROUTES = true)
    if (J,D,ID,REQ.record)  $\in$ 
        Recent_Req_Seen then
            Discard packet;
            Break;
        Else Recent_Req_Seen =
            Recent_Req_Seen +
            (J,D,ID,REQ.record)
        End if
    Else if (J,D,ID)  $\in$ 
        Recent_Req_Seen then
            Discard packet;
            Break;
        Else Recent_Req_Seen =
            Recent_Req_Seen + (J,D,ID)
        End if
    End if
End if
Prec = ComputePowerRequired();
record[K].P = Prec; /* put Prec in the last
element of record*/
If (I=d)
    reply(REQ);
    Break;
Else if (REP_FROM_CACHE = true)

    if ( $\exists$  an entry to D in cache)
        append cache route to
        record;
        record[I].energy_state[I]=
        energy_state ;
        reply(REQ);
        Break();

```

```

        End if
    end if
    if (REP_FROM_CACHE = false or
        not  $\exists$  any entry to D)
        record = record  $\cup$  (I, FULL_POWER,
            energy_state[I]);
        continue broadcasting REQ;
    End if
End if
End if
END

```

When receive RPLY(R as sender , D as destination , S as source , record)

```

If (RPLY.maintenance = true)
    NewPrec = RPLY.record.P;
    For each packet Pack in sauv[R] do
        Pack.DSRHeader.record[I].P = NewPrec;
        Send Pack to R;
    Od
    For each (x  $\in$  Recent_Full_Send/ x.adress=R)
    do
        Send ERROR(S,I,R,NewPrec) to
        x.PacketId.source;
        /*remove couple x*/
        Recent_Full_Send = Recent_Full_Send -
        {(x)};
    Od
    For (Each route Rou containing R as first
    hop) Do
        Rou.Hops[0].Power = NewPre;
        Update R.metrrique;
        Reorder R;
    Od
    Else if (S=I) /*I is the request's source*/
        Remove D from req_sent;
        For each node j in record do
            If (record[j].E - energy_state[j] >
                EnergStateThreshold)
                energy_state[j] = record[j].E;
            End if
        Od
        Update MinEnergy, MaxEnergy and  $\alpha$ 
        For each route in cache
            Compute route metric;
            reorder route;
        Od
        Compute metric of record's route;
        Insert record's route in route[D]
        regarding its metric;
        route(D) after WaitForRoutes;
    Else forward packet;
    End if
End if
END

```

When receive a data packet to forward

```

If (DSRHeader.record.nexthop  $\in$ 
    Recent_Full_Send)
    Put it in sauv[DSRHeader.record.nexthop];
Else if (DSRHeader.record.nexthop
     $\notin$  Recent_Failed_links)
    send it to next hop;
End

```

When receive LNKERROR (FailAdr, S, D, PacketId, Packet, type) message from MAC protocol

```

If (FaillAdr  $\in$  Recent_Full_Send)
    If (type = RREQ) /*link fail*/
        For each Src in Recent_Full_Send/
        Src.Adresse
            = FailAdr)
            Do
                Recent_Full_Send = Recent_Full_Send - {
                    (FaillAdr, (S,D,PacketId)) };
                Send ERROR(S, D, I, FaillAdr,  $\infty$ )
                to Src;
                Remove from cache each route
                containing FaillAdr as first hop;
            Od
            Recent_Failed_links = Recent_Failed_links
            + { FaillAdr};
        Else save packet in sauv[FaillAdr];
        End if
    Else
        Recent_Full_Send = Recent_Full_Send + {
            (FaillAdr, (S,D,PacketId)) };
        /*Resend packet using full power*/
        Packet.DSRHeader.record[I].P =
            FULL_POWER;
        Pagyback packet to a REQ packet;
        REQ.LinkChanged = true;
        Send REQ(I, FaillAdr,Id) to FaillAdr;
        Id++;
        // Update routes in cache
        For (Each route R containing Id as first hop)
        Do
            R.Hops[0].Power = FULL_POWER;
            Update R.metrrique;
            Reorder R;
        Od
    End if
End

```

When receive ERROR(Source, Dest ,BackId, UpId, Cost, Record)

```

If (Cost =  $\infty$ )
    remove all routes containing (BackId, UpId)
    link from cache;
    If (Dest  $\neq$  I) then forward Packet;
End if
Else
    For each route R containing (BackId,
        UpId) link in cache

```

```

Do
    R.hops of address BackId.P = cost;
    Update R.métrique;
    Reorder R;
Od
If (Dest ≠ I) then forward Packet;
End if
End if
End
When (Energy_state change significantly)
    Update energy_state[I];
    broadcast E_state(I, energy_state);
    id ++;
End
When receive E_state(J, energy_state)
    Update (MinEnergy, MaxEnreg, α);
    energy_state[j] = energy_state;
    For each route R in cache do
        Update R.métrique;
        Reorder R;
    Od
End

Procedures

route(D, num)
{
    if (there is no packet to route in buffer[D])
        return;
    else
        nbr_paths = number of routes to D;
        nbr_paquets = number of packets of
            buffer[D];
        
$$n = \lfloor \frac{2^{nbr\_paths - num - 1} * nbr\_paquets}{2^{nbr\_paths} - 1} \rfloor$$

        send n packets from current_packet[D]
        to D on route num;
        num ++; Update current_packet[D];
        route(D, num);
    end if
}

reply(REQ)
{
    If (links are bi-directional)
        /*send via reversed path*/
        Send RPLY(I, REQ.Dest, REQ.Source,
            (REQ.record)) to REQ.Source via
            (REQ.record)-1
    Else if (∃ a route to S in cache)
        Send RPLY(I, REQ.Dest, REQ.Source,
            REQ.record) to REQ.Source via best
            route;
    Else
        Piggyback RPLY(I, REQ.Dest,
            REQ.Source, REQ.record) to a new
            request packet REQ';
        broadcast REQ';
    End if;
End if
}

ComputePowEerRequired(REQ)
{
    lasthop = address of last element in
    REQ.record;
    Precep = value of reception power;
    val = REQ.record[lasthop].P - Precep +
    Pthreshold + Pmargin;
    return val;
}

```

6.2.5 Etude de complexité

Dans ce paragraphe, nous allons présenter une étude de complexité de notre protocole, en se basant sur la complexité de DSR présenté dans [ROY99, PAR97].

Notons :

Dest : Le nombre de nœuds destinataires

Nei : Le nombre maximum de nœuds voisins d'un nœud

MaxPath : le nombre de chemins (sans boucle) maximale, qui existent entre deux paires de nœuds

N: nombre de nœuds dans le réseau

Diam: diamètre du réseaux, c'est le nombre de saut maximal qui sépare deux nœuds.

Dans [ROY99], on donne la complexité sans aucune preuve. Avant de donner la complexité, nous proposons la propriété suivante.

Propriété: Supposant que la méthode de découverte de route de DSR est utilisée, et supposant que Le nœud D a V voisins. Alors :

Le nœud D recevra au maximum V routes pour une même requête.

Preuve:

Supposant que le nœud D reçoit plus de V routes différentes.

Alors au moins deux routes ont le dernier saut en commun, soit $P1 = \langle s, v_1, \dots, v_n, D \rangle$ et $P2 = \langle s, u_1, u_2, \dots, v_n, D \rangle$

Alors le nœud v_n propage deux fois la même requête. Ce qui est contradictoire avec la méthode utilisée. Notons que dans la méthode de propagation de requêtes utilisée par DSR, une même requête ne peut pas être propagée deux fois par un même nœud. Donc le nœud D ne peut recevoir pas plus de V route.

Cette propriété sera utilisée pour expliquer l'étude de complexité

6.5.2.1 Complexité de mémoire (MC)

La complexité de mémoire permet de déterminer l'espace de stockage nécessaire dans la mémoire de chaque nœud source.

Dans DSR, chaque nœud sauvegarde, pour chaque destination, tous les routes qu'il obtiens après la procédure de découverte de route.

Au pire des cas, chaque nœud obtiens pour chaque destination N_{ei} routes (voir propriété précédente).

Alors il maintient dans sa cache N_{ei} routes pour chaque destination, se qui donne $Dest * N_{ei}$ entrées [ROY99].

Dans cette étude, on ne tient pas compte de la taille d'une entrée, on la considère de complexité $O(1)$. Mais en fait, elle dépend de la longueur du chemin.

Pour le protocole que nous proposons (soit DSRPA). Si on n'utilise pas l'option ALL ROUTES, la seule différence avec DSR est dans la taille de l'entrée dans la table de routage. Si on considère quelle est de complexité $O(1)$ pour DSR, alors pour DSRPA elle est $O(3)$. Donc la complexité en mémoire est $dest * N_{ei} * 3$.

Cependant, si on utilise l'option ALL ROUTES, au pire des cas on va avoir $MaxPath$ routes pour chaque destination, donc la complexité est : $Dest * MaxPath$.

Mais en pratique, quand le réseau a une forte connectivité, beaucoup de requêtes seront perdu à cause des collisions, ce qui permet de découvrir moins de routes, donc moins de ressources mémoires seront consommées.

On plus de la table de routage que nous avons étendue, les structures de données que nous avons ajoutées dans DSRPA ont les complexités suivantes :

Energ_state	N
Sauv	N_{ei}
Recent_Full_Send	$Sour * N_{ei}$
Recent_Failed_links	N_{ei}

Tel que Sour est le nombre de nœud sources.

6.5.2.2 Complexité de communication (CC)

Elle permet de déterminer le nombre de messages de contrôle envoyés dans tous le réseau, pour obtenir une route, ou pour qu'un nœud source sera informé d'une défaillance de route (complexité de maintenance).

Pour DSR : Quand le nœud source initialise une requête, au pire des cas, tous les nœuds du réseau vont propager une fois la requête, une même requête ne sera pas propagée plus qu'une seule fois, ceci nécessite $N-1$ diffusion de la requête. et puisque dans ce cas il y aura $N-1$ nœuds qui participent dans l'ensemble des routes, il y aura $N-1$ diffusions pour la réponse. En tous, on a une complexité de $2*N$ Diffusions

Après une défaillance de lien, le paquet d'erreur envoyé sera diffuser par $N-1$ nœuds au maximum avant d'informer le nœud source de cette défaillance [ROY99].

Pour DSRPA : si on n'utilise pas l'option ALL ROUTES, le même raisonnement est valable, donc DSRPA aura la même complexité.

Toute fois, si on utilise cette option, Au pire des cas, chaque route a une longueur Diam. Donc au maximum, on aura: $\text{MaxPath} * \text{Diam}$ diffusions de la requête et le même nombre de diffusions pour la réponse, ce qui donne une complexité de $2 * \text{MaxPath} * \text{Diam}$.

La complexité pour la maintenance est toujours $O(N)$ diffusions.

Notons que cette complexité ne prend pas en compte la taille des paquets, mais plutôt leur nombre. Car c'est ce dernier qui montre le nombre des diffusions. la taille des paquets de contrôle est triplet pour DSRPA, mais généralement, les paquets de contrôle ont une taille très limitée.

6.5.2.3 complexité de temps (TC)

C'est le temps nécessaire pour que l'algorithme converge vers un état stable après une initiation de recherche de route, ou après une défaillance. Dans le premier cas, il s'agit du nombre de messages nécessaires pour que le nœud source détecte une route après l'envoi d'une requête. Quant au deuxième cas, c'est le temps qui sépare la détection d'une défaillance, et l'information du nœud source de cette défaillance.

Pour détecter une route, au pire des cas, la route (si elle existe) qui sépare le nœud source et la destination contient Diam sauts. La requête passe par Diam nœuds, avant d'arriver à la destination, et puis la réponse envoyée par la destination passe par Diam nœuds. Ce qui donne une complexité de $O(2*\text{Diam})$ étapes.

Pour une maintenance, le paquet de requête passe par Diam nœuds avant d'arriver à la source.

Cette complexité est la même pour tous les protocoles.

Le tableau suivant résume la complexité des protocoles, nous notons DSRPA0 la version qui n'utilise pas l'option ALL ROUTES, et DSRPA1 la version qu'il l'utilise.

	MC	CC(initialisation)	CC(maintenance)	TC(init)	TC(maint)
DSR	$O(\text{Des} * \text{Nei})$	$O(2 * N)$	$O(N)$	$O(2 * \text{Diam})$	$O(\text{Diam})$
DSRPA0	$O(3 * \text{Des} * \text{Nei})$	$O(2 * N)$	$O(N)$	$O(2 * \text{Diam})$	$O(\text{Diam})$
DSRPA1	$O(\text{MaxPath} * \text{Dest})$	$2 * \text{MaxPath} * \text{Diam}$	$O(N)$	$O(2 * \text{Diam})$	$O(\text{Diam})$

Figure 4.4

A decorative border with a wavy, undulating top and bottom edge, enclosing the central text box.

Chapitre 5

Environnement de simulation

I. Introduction

Pour évaluer les performances du protocole proposé dans les chapitres précédents, et le comparer avec le protocole DSR standard, nous avons effectué des simulations. Avant de parler de ces simulations, nous allons d'abord présenter, dans ce chapitre, l'environnement de simulation utilisé. Il s'agit du simulateur GloMoSim développé en langage PARSEC. Ce chapitre est composé, principalement, des parties suivantes : après cette brève introduction, nous allons donner quelques notions générales sur la simulation. Ensuite nous allons présenter le langage PARSEC. Cette présentation est suivie par la présentation du simulateur GloMoSim.

II. Généralités sur la simulation

La simulation connaît de nos jours un essor considérable. Ceci est dû aussi bien à l'intérêt théorique que présente la modélisation des systèmes simulés, que par les besoins croissants de simuler par ordinateur des réalisations de plus en plus complexe.

On constate en effet que les conditions d'expérimentation sont aujourd'hui toujours plus coûteuses, notamment aux techniques dites virtuelles. Celle ci consiste à représenter la réalité simulée sur ordinateur, à donner à celle ci des moyens de perception réalisés artificiellement, puis à simuler son fonctionnement.

C'est sur ce dernier point que la simulation joue un rôle important en mettant à la disposition de l'utilisateur un environnement d'expérimentation dont on peut faire varier les paramètres.

Les conceptions et les techniques de programmation ont fait de leur côté d'important progrès, en particulier dans les domaines de l'abstraction et du parallélisme. C'est ainsi que les langages de programmations d'aujourd'hui offrent des moyens performants, ce qui rend possible la réalisation d'un simulateur dans un environnement de programmation existant [ERA 96].

2.1 Système réel et objectif de simulation :

La simulation est une technique de modélisation du monde réel. Elle permet de représenter le fonctionnement d'un système composé de différents centres d'activité, de mettre en évidence les caractéristiques de ceci et les interactions entre eux, de décrire la circulation de différents objets traités par ces processus et en fin d'observer le comportement du système dans son ensemble et dans son évolution dans le temps.

On appelle communément *système réel* le phénomène que l'on veut observer. On y distingue d'une part les centres d'activités ou tâches d'activités, c-à-d les endroits où il se passe quelque chose ou un objet est créé, traité, transformé ou détruit, d'autre part les objets qui circulent dans le système et qui sont manipulés par différentes tâches actives.

Nous considérons que le but de l'opération est d'étudier un tel système et pour comprendre les relations entre ces composantes ou pour prédire ces performances dans un environnement donné [ERA 96].

2.2 Limite de l'expérimentation directe

Une expérimentation directe effectuée sur le terrain peut se révéler coûteuse, irrationnelle ou même impossible.

Il serait par exemple irrationnel de fermer arbitrairement l'un des guichets d'une banque et de laisser le public s'accumuler dans le hall pour le seul intérêt d'observer le phénomène.

Il serait de même inconcevable dans notre étude de mettre en œuvre un réseau Ad Hoc, de déplacer les nœuds et changer les différents paramètres pour évaluer les performances de notre algorithme de routage. Car l'expérimentation, en plus qu'elle est coûteuse, elle ne donne pas beaucoup de souplesse, et ne permet pas de changer les paramètres d'une manière souple.

C'est à cause de difficulté liée à l'expérimentation directe, dans le but aussi de pouvoir examiner facilement et rapidement des variantes de système étudié que l'on cherche à réaliser un modèle de ce système dont on peut analyser numériquement le comportement et sur la base de cette analyse, et inférer le comportement du système réel lui-même.

2.3 Systèmes discret et continue

Selon leurs structures on divise les systèmes de simulation en deux catégories, les systèmes discrets et les systèmes continus.

Un *système discret* est un système dans lequel les variables décrivant un état ne changent de valeur qu'en un nombre fini de points sur l'axe du temps. La station de service est un exemple de système discret, puisque les variables qui y figurent ne changent de valeur qu'à des instants précis. Par exemple le nombre de clients dans la station ne se modifie qu'à l'arrivée ou au départ d'un client.

Un *système continu* est un système dans le quel le temps s'écoule de façon continue et où les variables peuvent changer de valeur à tout instant. La simulation du vol d'un avion est un système continu en ce sens que les coordonnées de la vitesse de l'avion sont des fonctions qui prennent une valeur en tout point de l'axe du temps. Cela ne veut pas dire bien sûr que les fonctions elles mêmes soient continues. Le terme continu se rapporte à la perception du temps dans la simulation.

Il est possible de rendre un système discret, en le décomposant en activités pouvant être considérées comme instantanées et séparées par des intervalles de temps pendant lesquels il ne se passe rien de significatif pour l'étude.

Cette démarche est la base de la modélisation de la simulation par événements discrets.

2.4 Modèles de simulation

On distingue plusieurs modèles de simulation selon qu'il soient statiques ou dynamiques, déterministes ou aléatoires, continus ou discrets.

Un modèle de simulation est dynamique ou statique selon que les valeurs de ses variables se modifient dans le temps ou non.

Un modèle de simulation est déterministe s'il ne contient aucune variable aléatoire, il est aléatoire dans le cas contraire.

Dans le cas d'un système aléatoire, les résultats de la simulation sont eux même aléatoires et ne donnent qu'une estimation du comportement du système simulé.

Enfin, un modèle de simulation est continu si l'ensemble des instants considérés forme un intervalle compact sur l'axe des temps, il est discret si celui ci se compose d'un nombre fini ou dénombrable de valeurs isolées.

2.5 Gestion du temps

Une simulation consiste à gérer le temps ainsi que les actions qui sont liées aux différents instants du système réel, et à faire fonctionner abstraitement le modèle qui représente ce système.

Dans un modèle discret, le temps est géré par un échéancier et par une horloge centrale ou une horloge du système. L'échéancier est une liste d'événements ordonnés chronologiquement selon l'heure à laquelle ils doivent être activés. L'horloge du système donne à chaque instant l'heure du système de simulation. Elle est réglée au départ sur l'heure d'activation du premier événement. Chaque événement inséré dans l'échéancier en est retiré lorsque son heure d'activation est égale à celle du système de simulation. Des actions associées à cet événement sont alors exécutées, ensuite l'horloge du système est avancée à l'heure d'activation de l'événement suivant.

Dans un modèle continu le temps est discrétisé selon un pas donné, à chaque avance du temps, les valeurs des variables du système sont mises à jour [ERA 96].

2.6 Simulation par événements discrets

La simulation par événements discrets désigne la modélisation d'un système réel tel qu'il évolue dans le temps, par une représentation dans laquelle les grandeurs caractérisant le système (variable) ne changent qu'en un nombre fini ou dénombrable de points isolés dans le temps. Ces points sont les instants où se passent les événements, c-à-d le phénomène capable de modifier l'état de système et on appelle comme événement tout changement d'état du système réel se produisant à un instant donné, ainsi que les actions qui accompagnent ou caractérisent ce changement.

La simulation par événement discret consiste alors à prendre en compte dans la modélisation des tâches actives, les seuls instants où un événement se produit et à concentrer l'activité des tâches simulées sur ces instants là. On considère comme système de simulation par événements discrets tous les systèmes basés sur cette abstraction[ERA96].

2.7 Simulateur

Nous appelons simulateur par événements discrets, un programme qui met en œuvre un modèle de simulation par événements discrets. La tâche première d'un simulateur est d'assurer que la chronologie des événements soit respectée. À chaque occurrence d'un événement, les actions qui sont associées à celui ci sont exécutées[ERA 96].

Les applications de la simulation sont innombrables. Parmi les domaines dans lesquels elle est le plus utilisée, on s'intéresse à la simulation des réseaux mobiles (réseaux ad hoc). L'un des simulateurs de ce type est GloMoSim développé en utilisant le langage PARSEC.

III. PARSEC

3.1 Introduction

PARSEC (PARallel Simulation Environment for Complex system) est un langage dérivé d'un autre langage appelé MAISIE. Ils sont basés sur le langage C et destinés à la simulation des événements discrets, ils sont développés à l'université UCLA (University California Los Angeles), dans le laboratoire UCLA PARALLEL COMPUTING LABORATORY.

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

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

3.2 Entité

Une entité représente un objet spécifique dans le système physique, chaque programme dans PARSEC doit avoir une entité principale « driver ». Cette entité initialise l'exécution de la simulation, elle a en fait, le rôle de la fonction « main » dans le langage « C ». La déclaration d'une entité est similaire à la déclaration d'une fonction en « C », mais l'entité ne retourne pas de valeur, cette déclaration définit un type d'entité « ename » sur lequel on peut créer plusieurs instances, en exécutant l'instruction 'new'.

3.3 Message

PARSEC utilise des objets de type « message » composés de nom et une liste de paramètres, la définition d'un message est similaire à la définition de structure en « C », Les paramètres de message peuvent inclure des tableaux de taille fixe et qui passent par valeur.

Il est possible d'utiliser les pointeurs comme des paramètres de message, mais ceci est dangereux et déconseillé dans une architecture de mémoire distribuée.

Les entités communiquent entre elles en utilisant les messages. Chaque entité a un buffer unique, des primitives d'envoi et de réception asynchrone de messages sont fournies pour respectivement le dépôt et l'enlèvement des messages du buffer.

Ces primitives sont :

send () : Une entité envoie un message par l'instruction send, chaque message sera estampillé et déposé dans le buffer de l'entité destinataire.

receive() : Une entité accepte les messages de son buffer en exécutant l'instruction receive.

3.4 évènement

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

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

3.5 L'exécution parallèle :

En plus de l'algorithme séquentiel, parsec support deux types d'algorithmes de synchronisations parallèles : conservatrice et optimiste, qu'on va voir par la suite. Dans les simulations, nous avons utilisé la version séquentielle, mais nous allons comme même donné une présentation générale de l'exécution parallèle.

Pour une exécution parallèle certaines restrictions doivent être suivies:

1-variables globales : la simulation parallèle en PARSEC ne doit pas utiliser les variables globales, il est interdit pour deux entités d'avoir un pointeur sur la même donnée.

L'état d'une entité doit être seulement déterminé par des variables déclarées dans cette entité, et ces variables ne doivent pas être partagées avec d'autres entités. Ces restrictions sont pour les raisons suivantes :

- PARSEC assume que tout changement dans l'état de l'entité est le résultat de message, cette règle est violée avec un état d'entité globale.
- La simulation optimiste à besoin que l'état d'entité sauvegardé ne fonctionne pas avec l'état partagé, les variables globales ne sont pas contrôlées par le système.
- La mémoire partagée a une copie des variables globales quant au mémoire distribuée elle a N copies non consistantes.

2- Les pointeurs ne peuvent pas être passés dans des paramètres de message ou les paramètres d'entité.

3.5.1 Partitionnement :

Dans une implémentation, le modèle de simulation doit être partitionné par l'allocation d'entité dans les processeurs.

PARSEC permet au programmeur de spécifier le nœud dans lequel l'entité sera créée en utilisant l'option « at » au moment de la création.

La simulation parallèle utilise deux modèles :

3.5.2 Simulation conservatrice :

Pour l'utilisation des protocoles conservateurs, il faut prendre en compte les deux points suivant :

a) *topologie de communication*: Le système doit être informé sur la topologie de communication entre les entités, c-à-d savoir l'ensemble de source et l'ensemble de destination de chaque entité.

b) *lookahead* : C'est le temps entre le temps courant de l'entité et le temps dans lequel va envoyer son prochain message, « lookahead » doit être spécifié dans le programme conservateur de PARSEC .

Le système réinitialise la valeur de « lookahead » après chaque instruction « hold ».

« setlookahead » ou « setdestlookahead » doit être exécutée avant la première instruction « receive » et après chaque instruction « hold ».

3.5.3 Simulation optimiste :

L'algorithme de synchronisation optimiste n'est pas encore disponible pour PARSEC, les variables globales ne sont pas contrôlées. Si les variables globales sont modifiées durant l'exécution, cela conduit à des erreurs de programme.

Toute variable déclarée dans chaque entité sera contrôlée par le système dans quelques points de contrôle.

3.6 Exécution de parsec :

Parsec génère des fichiers temporaires en langage C. pour la compilation de ces fichiers et leur édition de lien il utilise un compilateur c. Pour l'environnement Windows, il utilise Visual C++. Ainsi après son installation, PARSEC nécessite une configuration du system.

- Sous Windows (95/98) :

Pour exécuter parsec sous la plate-forme Windows (95/98), il faut avoir le chemin de Visual C++ dans les variables PATH, INCLUDE et LIB. Alors on rajoute au fichier 'Autoexec.bat' les lignes suivantes :

```
Set path = c:\devstudio\vc\bin; c:\devstudio\sharedide\bin; c:\parsec\bin
```

```
set INCLUDE = c:\devstudio\vc\include; c:\parsec\include
```

```
set LIB = c:\devstudio\vc\lib; c:\parsec\runtime
```

Telque c:\devstudio est le chemin d'installation de Visual c++, et c:\parsec est celui de parsec. Si ces chemins ont des valeurs différentes, on a cas substituer les sous chaînes c:\devstudio et c:\parsec par ces valeurs dans les lignes précédente.

- Sous WindoswNT/2000:

La même chose que Windows mais on utilise le panneau de configuration au lieu de 'Autoexec.bat'.

- Sous Linux : on exécute les commandes suivantes

```
# !/bin/csh
```

```
setenv PCC_DIRECTORY "le chemin du répertoire parsec"
```

```
${PCC_DIRECTORY}/bin/parsecc $*
```

On met le chemin de parsecdir/bin dans la variable PATH.

Dans l'annexe on trouve une description générale de la syntaxe du langage PARSEC, une description plus détaillé est disponible dans [PAR]

IV. GloMoSim

4.1 Introduction

GloMoSim (Global Mobile Information System Simulator) est un environnement de simulation pour les réseaux sans fil et filaires. Il a été conçu à l'université californienne UCLA, dans la laboratoire UCLA PARALLEL COMPUTING LABORATORY[ZEN98]. Ceci en utilisant le langage de programmation PARSEC (développé dans le même laboratoire).

Actuellement GloMoSim peut simuler un réseau purement sans fil, plus particulièrement un réseau ad hoc. Dans le futur, des nouvelles versions pourront simuler un réseau filaire aussi bien qu'un réseau hybride.

La plupart des systèmes réseaux sont construits en utilisant une approche basée sur les couches qui est semblable à l'architecture à sept couches de la norme OSI.

Des APIs standards sont utilisées entre les différentes couches de simulateur. Ceci autorisera l'intégration rapide et facile des modèles et protocoles développés aux différentes couches par différents développeurs.

Le tableau suivant montre les différentes couches, et quelque protocole disponible dans chaque couche. Ces protocoles seront expliqués ultérieurement.

Le transfert de messages dans GloMoSim est montré dans la figure 5.2

Couches	Protocoles
Mobilité	Random drunken et Random waypoint
Propagation	Free space
Radio	radio capture et sans capture
Liaison de donnée (MAC)	CSMA, FAMA, IEEE 802.11 et MACA
Réseau (Routage)	IP avec AODV, Bellman-Ford, DSR, Fisheye, LAR scheme 1, NS DSDV, OSPF or WRP
Transport	TCP et UDP
Application	CBR, FTP, HTTP et Telnet

Figure 5.1 Architecture en couches de Glomosim

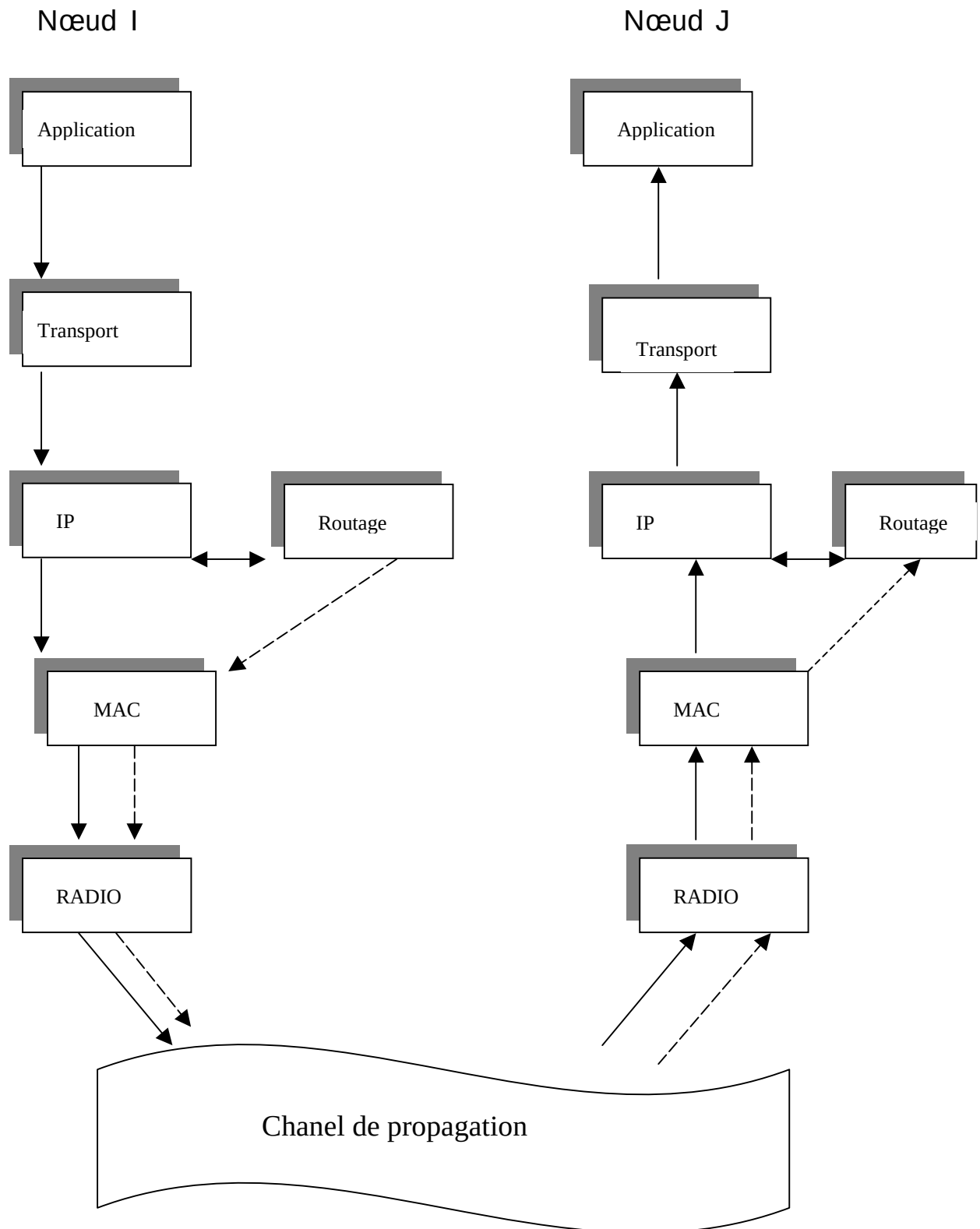


Figure 5.2 Transfert des paquets dans GloMoSim

4.2 Agrégation des nœuds

Dans PARSEC, une approche simple à concevoir un modèle de simulation du réseau est de créer pour chaque nœud du réseau une entité. Bien que cette approche soit facile à comprendre, elle cause des problèmes.

Si une entité doit être créée pour chaque nœud, les exigences de la mémoire augmentent pour un modèle avec un grand nombre des nœuds, puisque chaque entité est un processus indépendant qui exige une mémoire additionnelle de travail.

La performance de la simulation se dégrade à cause du changement de contexte entre plusieurs entités.

Pour circonvenir ces problèmes, l'agrégation des nœuds a été introduite dans GloMoSim. Avec l'agrégation des nœuds, une seule entité peut simuler plusieurs nœuds du réseau dans le système. Quand le code de simulation pour un nœud particulier est exécuté, il n'a pas accès aux structures de données des autres nœuds.

La technique de l'agrégation des nœuds implique que le nombre des nœuds dans le système peut être augmenté en maintenant le même nombre des entités dans la simulation.

En fait, la seule exigence est que nous avons besoin seulement autant d'entités que le nombre des processeurs sur lequel la simulation est exécutée. D'ici, une simulation séquentielle a besoin seulement d'une seule entité.

Dans GloMoSim, chaque entité représente une région géographique de la simulation. Cette région est rectangulaire et régulière (partition), une partition peut avoir huit partitions voisines au plus.

Quand un nœud du réseau envoie un message, le message doit être envoyé au plus aux huit entités voisines.

Si chaque entité représente un seul nœud, l'envoi d'un message devient très difficile. La première option d'envoi de message est:

Chaque entité garde trace des autres entités dans son domaine de puissance (portée), cette option est difficile car la topologie du réseau change constamment lorsque la mobilité est introduite dans la simulation.

La deuxième option est quand un nœud envoie un message, il sera envoyé à toutes les autres entités dans la simulation, l'entité de la réception acceptera le message s'il est dans la portée de l'envoyeur.

D'ici une transmission du message devient très compliqué quand l'agrégation du nœud n'est pas utilisée.

4.3 Structure des répertoires de GloMoSim

Après avoir télécharger et décompresser GloMoSim (glomo.tgz), il devrait contenir les répertoires suivants:

application: contient le code pour la couche d'application

bin: pour l'exécutable et les fichiers d'entrée/sortie.

doc: contient une documentation pour le simulateur.

include: contient les fichiers commun d'inclusion (.h).

mac: contient le code pour la couche mac.

main: contient les fichiers principaux tel que driver.pc (le point d'entrée)

network: contient le code pour la couche réseau.

radio: contient le code pour la couche radio.

transport: contient le code pour la couche transport.

scenario: contient les fichiers d'entrée et d'initialisation pour les différents scénarios.

Cette structure facilite la tâche de modification et d'amélioration telle que l'ajout d'un nouveau protocole dans une couche particulière, et l'ajout d'un nouveau calcul statistique.

4.4 Exécution de GloMoSim

une fois téléchargé est décompressé, le code de GloMoSim sera présent et structuré dans les répertoires présentés dans le paragraphe précédent. Cependant, pour obtenir l'exécutable, il faut d'abord le générer, en compilant les fichiers obtenus. pour ceci il faut avoir la dernière version de Parsec.

Les étapes de la création de l'exécutable sont :

a) Sous Windows Nt:

- Il suffit d'utiliser le fichier batch Makent.bat (situé dans le répertoire main) pour créer l'exécutable. Ce fichier contient toutes les commandes pour compiler et lier les fichiers sources, on a qu'à exécuter makent. Comme on l'a déjà, il faut s'assurer que le chemin de pcc.exe ainsi que le fichier cl.exe du compilateur visual c++ apparaît dans la variable système "path", car makent fait des appels directs à pcc.exe pour compiler et lier les fichiers « .pc » et cl.exe pour compiler les fichiers ".c" et ".cpp".

Remarque: pour l'environnement Windows, Glomosim fonctionne sous les plates formes professionnelles utilisant la technologie NT, à savoir "WindowsNt et Windows2000". Il n'est pas possible de générer l'exécutable sous les plate forme "windows 95" ou "Windows 98".

Si l'étape de la création est terminée avec succès on obtient l'exécutable "Glomosim.exe" dans le répertoire "bin". Pour l'exécution, « Glomosim.exe » a besoin d'un paramètre de la ligne de commande, qui est un fichier d'entrée. Un exemple de fichier est CONFIG.IN dans le répertoire "bin".

Un fichier appelé "GLOMO.STAT" est produit à la fin de la simulation et contient toutes les statistiques (demandées dans le fichier d'entrée) produites par la simulation.

b) Sous linux:

- Exécuter "make depend" afin de créer la liste de dépendances dans le "Makefile".
- S'assurer que le bon chemin pour le compilateur Parsec est spécifié dans le Makefile pour la variable "par".
- Lancer "make" pour créer l'exécutable

Un exemple de fichier est CONFIG.IN dans le répertoire "bin", un fichier appelé "GLOMO.STAT" est produit à la fin de la simulation et contient toutes les statistiques (demandées dans le fichier d'entrée) produites par la simulation.

4.5 Les APIs de GloMoSim

Le noyau de GloMoSim offre des APIs sous forme de fonctions, et de messages à échangés entres les différentes couches. L'utilisation de ces APIs facilite au développeur l'ajout de nouveaux protocoles.

4.6 Description du fichier d'entrée (de configuration)

Cette section explique les différents paramètres du fichier d'entrée à fourni à GloMoSim. Ces paramètres représentent la configuration du réseau à simuler. Dans ce fichier, n'importe quelle expression qui suit un "#" dans une ligne est traitée comme un commentaire, et chaque ligne représente un paramètre.

Le terrain

Ces deux paramètres représentent la longueur et la largeur du terrain physique dans lequel les nœuds sont simulés.

TERRAIN-RANGE-X 100

TERRAIN-RANGE-Y 100

Cet exemple représente une région de dimension 100 mètres par 100 mètres.

Tous les paramètres sont en mètres.

Domaine de puissance

Ce paramètre représente le domaine de puissance des nœuds dans la simulation.

POWER-RANGE 50

Dans cet exemple, un nœud peut atteindre tout autre nœud sur un rayon de 50 mètres.

Temps de simulation

Ce paramètre représente le temps maximum de la simulation, le nombre peut être suivi par une lettre facultative pour modifier l'unité de temps de la simulation.

SIMULATION-TIME 100M

Cette ligne de configuration fixe le temps à 100 minutes, on peut aussi utiliser les unités suivantes:

NS : nano seconde

MS : milli-seconde

S : seconde

M : minute

H : heure

D : jour

Si on ne précise pas l'unité, la seconde est utilisée par défaut.

Nombre de nœuds

Ce paramètre représente le nombre de nœuds à simuler.

NUMBER-OF-NODES 10

Dans ce cas on simule 10 nœuds.

L'emplacement initial des nœuds

Ce paramètre représente la stratégie du placement des nœuds, les stratégies qui sont implémentées actuellement sont:

- RANDOM: Les nœuds sont placés aléatoirement dans le terrain physique.
- UNIFORM: Basé sur le nombre des nœuds dans la simulation, le terrain physique est divisé en plusieurs cellules, Dans chaque cellule, un nœud est placé aléatoirement.
- GRID: Si on choisit cette stratégie il faut aussi préciser un autre paramètre (GRID-UNIT).

Le placement des nœuds commence à (0, 0), et les nœuds sont placés dans un format de grille, la distance qui sépare chaque nœud de ses voisins est précisée par le paramètre GRID-UNIT.

Pour cette stratégie le nombre de nœuds doit être le carré d'un nombre entier.

- FILE: Les positions des nœuds sont lues d'un fichier, NOEUD-PLACEMENT-FILE est le paramètre qui prend ce nom.

Chaque ligne du fichier contient les coordonnées x et y d'un nœud séparées par au moins un espace.

```
NODE-PLACEMENT    RANDOM
#NODE-PLACEMENT    UNIFORM
#NODE-PLACEMENT    GRID
#GRID-UNIT          30
#NODE-PLACEMENT    FILE
#NODE-PLACEMENT-FILE nodes.input
```

Dans ce cas la stratégie choisie est RANDOM, pour la changer il suffit de mettre la première ligne en commentaire et activer le mode voulue et le paramètre qu'il faut (si c'est le cas).

La bande passante

Le paramètre suivant représente la bande passante utilisée par les nœuds pour la transmission des messages.

```
BANDWIDTH          2000000
dans ce cas la bande est 2000000 bps
```

Protocoles et modèles à utiliser

Ces paramètres spécifient le protocole à utiliser dans chaque couche, Glomosim peut simuler différents protocoles. Par exemple, pour la couche radio, nous avons une interface de communication avec et sans capacité de capture. Pour la couche MAC, nous avons des protocoles pour CSMA, MACA, et IEEE802.11. Pour le protocole du routage nous avons Bellmanford et OSPF, etc....

```
RADIO-TYPE          RADIO-NO-CAPTURE
#RADIO-TYPE          RADIO-CAPTURE
#
#MAC-PROTOCOL        CSMA
#MAC-PROTOCOL        MACA
MAC-PROTOCOL         802.11
#
#
```

```
#ROUTING-PROTOCOL  BELLMANFORD
#ROUTING-PROTOCOL  AODV
ROUTING-PROTOCOL  DSR
#ROUTING-PROTOCOL  LAR1
#ROUTING-PROTOCOL  OSPF
#ROUTING-PROTOCOL  WRP
#ROUTING-PROTOCOL  FISHEYE
#ROUTING-PROTOCOL  NS_DSDV
```

```
#ROUTING-PROTOCOL  STATIC
#STATIC-ROUTE-FILE  ROUTES.IN
```

```
#
NETWORK-PROTOCOL  IP
```

Dans cette configuration nous avons choisi: le modèle RADIO-NO-CAPTURE pour la couche radio, le protocole IEEE802.11 pour la couche mac, DSR pour le protocole de routage, et IP pour la couche réseau.

Pour la couche du transport les protocoles peuvent être utilisés individuellement ou ensemble.

Il suffit de mettre 'YES' pour le protocole choisi, et 'NO' pour les autres.

```
TRANSPORT-PROTOCOL-TCP  YES
TRANSPORT-PROTOCOL-UDP  YES
```

Dans ce cas les deux protocoles TCP et UDP sont choisis.

Mobilité

S'il n'y a aucun mouvement des nœuds dans le modèle, le paramètre MOBILITY est mis à NO, sinon il prend comme valeur le type de mobilité.

Si un nœud est placé dans la position (x, y), il peut être déplacé à (x-1, y), (x+1, y), (x, y-1), et (x, y+1), si et seulement si la nouvelle position est dans le terrain physique.

Les types de mobilité sont:

-RANDOM-WAYPOINT: un nœud aléatoirement sélectionne une destination du terrain physique, il se déplace vers cette destination avec une vitesse entre MOBILITY-WP-MIN-SPEED et MOBILITY-WP-MAX-SPEED. Après qu'il atteigne sa destination, le nœud reste pour une période de temps égale à MOBILITY-PAUSE.

-RANDOM-DRUNKEN: un nœud se déplace aléatoirement à une de ses positions avoisinantes chaque MOBILITY-INTERVAL. Le paramètre MOBILITY-PAUSE n'est pas utilisé dans ce modèle.

Notons que les valeurs de MOBILITY-INTERVAL et MOBILITY-PAUSE sont représentées dans le même format vu pour le paramètre SIMULATION-TIME.

-TRACE: Dans ce cas le mouvement des nœuds est lu à partir du fichier présenté par MOBILITY-TRACE-FILE.

exemple :

```
MOBILITY TRACE
```

```
MOBILITY-TRACE-FILE mobility.in
```

dans cet exemple le fichier qui contient l'historique du déplacement est nommé mobility.in.

Le format de chaque ligne du fichier est:

<temps de déplacement> <nœud> <position de la destination en x> <position de destination en y> <vitesse de déplacement>

Pour un nœud donné, les temps de déplacement doivent être chronologiques

Applications

Le paramètre suivant contient le nom de fichier qui précise les applications à établir entre les nœuds.

exemple : si le fichier est nommé app.conf on met :

APP-CONFIG-FILE app.conf

Dans ce fichier, chaque ligne représente une application entre deux nœuds, en précisant le type de la application et les paramètres nécessaires. Les syntaxe pour chaque type d'application sont.

- *FTP* : <source> <destination> <items à envoyer> <temps de début>
- *TELNET* : <source> <destination> <durée> <temps de début>
- *CBR* : <source> <destination> <items à envoyer> <taille de l'item> <intervalle>
<temps de début> <temps de terminaison>

<source> : identificateur du nœud source

<destination> : identificateur du nœud destination

<items à envoyer> : le nombre de paquets à envoyer

<taille de l'item> : représente la taille du paquet

<intervalle> : durée entre 2 envois de paquets de la couche d'application.

<temps de début> : temps de début de session

<temps de terminaison> : temps de fin de session

<durée> : la durée de la session

Type statistique

Les paramètres suivants déterminent les statistiques à obtenir dans le fichier de sortie (glomo.stat), chaque type de statistique est présenté par un paramètre qui prend la valeur 'YES' si le type est sélectionné, et 'NO' dans le cas contraire.

```
#
APPLICATION- STATISTICS NO
TCP-STATISTICS          NO
UDP-STATISTICS          NO
ROUTING-STATISTICS      NO
NETWORK-LAYER-STATISTICS NO
MAC-LAYER-STATISTICS    NO
RADIO-LAYER-STATISTICS  YES
CHANNEL-LAYER-STATISTICS NO
#
```

Ici seules les statistiques de la couche radio seront trouvées dans le fichier de sortie.

4.7 Les statistiques obtenues dans GloMoSim

Le fichier résultat de la simulation contient des statistiques sur les paramètres et les protocoles choisis.

Chaque ligne dans ce fichier a le format suivant

Node : < numéro de nœud>, *Layer* : <nom de la couche>, <statistique>

Exemple:

Node: 9, Layer: RadioCapture, Total number of collisions is 0

Ceci veut dire que le nombre total de collision pour l'interface réseau du nœud 9 qui utilise le model radio CAPTURE est 0. c à d que tous les paquets sont bien reçus par ce nœud

Statistiques de la Couche radio (physique) :

Nombre total de paquets provenant de la couche mac

Nombre total de paquets de provenant du canal

Nombre total de collisions

L'énergie consommée.

Statistique de la couche MAC :

CSMA

Nombre de paquets provenant de la couche réseau

Nombre de paquets perdus dus au débordement du buffer

Nombre de paquets UNICAST envoyés a la couche canal

Nombre de paquets BROADCAST envoyés a la couche canal

Nombre de paquets UNICAST reçus correctement

Nombre de paquets BROADCAST reçus correctement

MACA

Nombre de paquets provenant de la couche réseau

Nombre de paquets perdus dus au débordement du buffer

Nombre de paquets UNICAST envoyés au canal

Nombre de paquets BROADCAST envoyés au canal

Nombre de paquets UNICAST reçus correctement

Nombre de paquets BROADCAST reçus correctement

Nombre de paquets RTS (Request To Send) envoyés

Nombre de paquets CTS (Clear To Send) envoyés

Nombre de paquets RTS obtenus

Nombre de paquets CTS obtenus

Nombre de paquets erronés «NOISY PACKETS» obtenus

802.11

Nombre de paquets provenant de la couche réseau

Nombre de paquets perdus dus au débordement du buffer

Nombre de paquets UNICAST (non-fragmenté) envoyés au canal

Nombre de paquets BROADCAST envoyés au canal

Nombre de paquets UNICAST reçus correctement

Nombre de paquets BROADCAST reçus correctement

Nombre de paquets retransmis dus à une expiration de délai des paquets CTS

Nombre de paquets retransmis dus à une expiration du délai des paquets ACK

(acquittement)

Nombre de paquets retransmis dus à une expiration du délai de paquet FRAG ACK

Nombre de paquets perdus dus au dépassement du nombre maximum de retransmissions

Statistiques de la Couche du réseau

Nombre de paquets provenant de TCP

Nombre de paquets envoyés à TCP

Nombre de paquets provenant d'UDP

Nombre de paquets envoyés à UDP

Nombre de paquets provenant d'OSPF

Nombre de paquets envoyés à OSPF

Nombre de paquets TCP perdus dus au dépassement ttl (time to live)

Nombre de paquets UDP perdus dus au dépassement ttl

Nombre de paquets OSPF perdus dus au dépassement ttl

Nombre moyen de sauts que les paquets TCP ont traversé

Nombre moyen de sauts que les paquets UDP ont traversé

Nombre moyen de sauts que les paquets OSPF ont traversé

Statistiques de routage

Bellmanford

Nombre total de paquets qui ont bouclé

Nombre total de table de routage transmis

Nombre total de mises à jour des tables envoyées par événement

Nombre total de mises à jour des tables de routage

Nombre total de paquets reçus de la couche MAC

Nombre total de paquets reçus de la couche transport

Nombre total de paquets envoyés

Nombre total de paquets qui appartiennent à ce nœud

Nombre total de sauts traversés par les paquets du nœud

Nombre total de paquets perdus

Nombre total de paquets perdus dus à l'absence d'information de routage

OSPF

Le nombre de paquets HELLO envoyés

Le nombre de paquets HELLO reçus

Nombre de paquets LSA retransmis

Nombre total de paquets LSA envoyés

Nombre de paquets LSA Reçus

Nombre de paquets ACK d'état de lien envoyés

Nombre de paquets ACK d'état de lien reçus

Nombre de mises à jour de la Table de routage

DSR

- Nombre de paquets de requête transmis
- Nombre de paquets de réponse transmis
- Nombre de paquets d'erreur transmis
- Nombre total de paquets de contrôle transmis
- Nombre de paquets de données transmis
- Nombre de paquets de données générés
- Nombre de paquets de données reçus

AODV

- Nombre de paquets de requête transmis
- Nombre de paquets de réponse transmis
- Nombre total de paquets de contrôle transmis
- Nombre de paquets de données transmis
- Nombre de paquets de données générés
- Nombre de paquets de données reçus

LAR

- Nombre de paquets de requête générés
- Nombre de paquets de requête transmis par les nœuds intermédiaires
- Nombre de paquets de réponse générés
- Nombre de paquets de réponse transmis par les nœuds intermédiaires
- Nombre de paquets d'erreur générés
- Nombre de paquets d'erreur transmis par les nœuds intermédiaires
- Nombre de paquets de données générées
- Nombre de paquets de données transmis par les nœuds intermédiaires

WRP

- Nombre de paquets de routage envoyés
- Nombre de paquets de routage reçus

FSR

- Nombre de paquets de mises à jours de 1^{ier} niveau (intra scope)
- Nombre de paquets de mises à jours de 2^{ime} niveau (inter scope)
- Nombre de paquets reçus du protocole UDP
- Messages de contrôle en octets

Statistiques de la Couche du transport

TCP

- Nombre de paquets envoyés à la couche réseau
- Nombre de paquets de données envoyés
- Nombre de paquets de données en séquence
- Nombre de paquets de données retransmis
- Nombre de paquets ACK-ONLY envoyés
- Nombre de paquets de contrôle (SYN|FIN|RST) envoyés

Nombre de paquets de Fenêtre mises à jour envoyés
Nombre de paquets de données reçus
Nombre de paquets ACK reçus en séquence
Nombre de paquets ACK dupliqués reçus
Nombre de paquets de contrôle (SYN|FIN|RST) reçus
Nombre totale de paquets erronés
Nombre de paquets reçus dont le champ CCKSUM indique une erreur
Nombre de paquets reçus avec une mauvaise taille

UDP

Nombre de paquets provenant de la couche application
Nombre de paquets envoyés à la couche application

Générateurs de trafic

FTP client / serveur

Le temps de début de session
Le temps de fin de session
Nombre d'octets envoyés
Nombre d'octets reçus
Le débit

Telnet client / serveur

Le temps de début de session
Le temps de fin de session
Nombre d'octets envoyés
Nombre d'octets reçus
Le débit

CBR client / serveur

Le temps de début de session
Le temps de fin de session
Nombre de paquets/octetes envoyés
Nombre de paquets/octetes reçus
Le débit
Le délai moyen de transfert de données

A decorative border with a wavy, undulating top and bottom edge, enclosing the central text box.

Chapitre 6

Démarche et Résultats de Simulation

I. Introduction

Pour évaluer les performances du protocole proposé dans le chapitre 4, nous avons effectué des simulations en utilisant le simulateur GloMoSim. A partir du protocole proposé (DSRPA), nous avons dérivé quatre versions. Ceci en faisant varier le paramètre ALL ROUTES (activé, désactivé), et la fonction Alpha (linéaire, racine). Et nous avons les comparé avec DSR standard implémenté dans GloMoSim. Que nous avons utilisé comme benchmark.

L'objectif de notre protocole de routage est de minimiser la consommation d'énergie. Et surtout, d'assurer une longue vie aux batteries, et d'éviter le partitionnement le plus longtemps possible.

Les résultats montrent une amélioration considérable pour la consommation d'énergie, ainsi pour les durées de vie des batteries.

II. Environnement de simulation

Nous avons utilisé l'environnement de simulation GloMoSim décrit dans le chapitre précédent. Cependant, plusieurs modifications (extensions) ont été nécessaires pour nos simulations. Dans ce qui suit, nous allons décrire ces modifications, ainsi que les paramètres utilisés.

2.1 Extension de GloMoSim

nous avons principalement ajouté à GloMoSim :

1. **Implémentation du protocole Proposé DSRPA:** le protocole proposé a été ajouté aux protocoles de routage disponibles dans GloMoSim. Ceci en ajoutant principalement les fichiers : DSRPA.pc (3500 lignes de code), qui représente le code source en Parsec du protocole. Et DSRPA.h (500 lignes de codes), qui représente les structures de données. Cependant, pour intégrer le protocole dans le simulateur, et assurer son bon fonctionnement, les modifications suivantes sont nécessaires.
2. **Modification des structures de données globales:** afin de permettre l'échange des informations d'énergie (ajoutées) entre les protocoles des différentes couches, nous avons modifié le format principale du message, utilisé dans GloMoSim. Ceci en ajoutant des champs à la structure *Message* située dans le fichier *message.h*. Les structures de données qui représentent les informations d'énergie, ont été ajoutées aux fichiers : *main.h*, *message.h*, et *structmsg.h*.
3. **Modifications au niveau du protocole IP:** le protocole de routage est situé au niveau de la couche réseau, le protocole IP est le protocole maître dans cette couche. Tous les messages (paquets) provenant des couches supérieures, passent obligatoirement par ce protocole, avant de passer au protocole de routage. En faite, c'est le protocole IP qui lance le protocole de routage sélectionné dans la configuration, en lui passant les messages. Pour intégrer notre protocole, nous avons

besoin d'ajouter du code aux fichiers : nwip.pc, network.pc, ip.h, nwip.h, network.h, et nwcommon.h qui représentent le code et les structures de données du protocole IP.

4. **Modifications au niveaux de la couche MAC :** Pour permettre le transfert des informations d'énergie ajoutées au message à travers la couche MAC, que ce soit de haut en bas (réseaux → MAC, et MAC → RADIO), ou de bas en haut (RADIO → MAC, et MAC → réseaux). Nous avons besoin de modifier l'implémentation du protocole de la couche MAC IEEE 802.11 que nous avons choisie. Ceci en ajoutant du code au fichier 802_11.pc.
5. **Modifications au niveau de la couche RADIO:** le modèle de l'interface réseaux implémenté dans GloMoSim est basé sur l'utilisation d'une puissance de transmission fixe. Pour implémenter la technique de contrôle de puissance, nous avons ajouté des extensions à cette couche. Nous avons aussi ajouté le déclenchement de l'événement E_State quand la charge de la batterie change considérablement. Ainsi que les calculs de l'énergie consommée, et de la durée de vie des batteries. Nous avons ajouté aussi le transfert des informations d'énergie à travers cette couche; que ce soit de haut en bas (MAC → RADIO, et RADIO → canal), ou de bas en haut (canal → RADIO, et RAIO → MAC). Tout ceci en modifiant les fichiers : radio.pc, radio_has.pc, et radio .h.
6. **Modifications au niveau la couche canal :** le modèle de propagation utilisé dans GloMoSim ne tient pas en considération la puissance de transmission. Il la considère fixe. Nous avons choisi le model proposé dans [SDTa02], que nous avons implémenté en ajoutant le fichier PropagatePA.pc (700 lignes du code).
7. **Ajout de la récupération des paramètres ajoutés, et du calcul des statistiques:** plusieurs paramètres supplémentaires doivent être ajoutés dans les fichiers d'entrée. Ces paramètres sont nécessaires pour notre protocole de routage. Il se composent en deux catégories :
Les paramètres globaux: tel que la charge de la batterie, le taux de transmission du message E_state, ...etc. Ces paramètres sont à ajoutés dans le fichier d'entrée principale.
Les paramètres utilisés par DSRPA : telle que la fonction alpha, la réponse à partir du cache, l'option ALL ROUTES...etc. Ces paramètres doivent être ajoutés dans un fichier d'entrée séparé, dont le nom doit figuré dans le fichier d'entree principale. Plusieurs statistiques utilisées, tel que la durée de vie des batteries, ne sont pas implémentés. Nous avons alors ajouté le calcul de ces paramètres.
afin de récupérer (lire) tous les paramètres d'entrée, et de les mettre dans les variables qui conviennent, et afin de calculer et d'afficher les statistiques, nous avons modifié les fichiers: Glomo.pc. et radio.pc. Le fichier DSRPA.pc ajouté contient aussi des procédures de lecture, de calcul de statistiques et d'affichage.

2.2 métriques de comparaison

Un protocole de routage est efficace en consommation d'énergie par rapport à un autre, si d'une part, il cause moins de consommation d'énergie moyenne, et d'autre part, il assure que les batteries des nœuds restent *ensemble* en vie le plus longtemps. Ceci afin d'éviter le

partitionnement le plus longtemps possible. La deuxième condition n'est pas toute à fait liée à la première (comme on a vu dans le chapitre 4). Nous proposons alors deux classes de métriques de comparaisons :

a) Energie consommée : le calcul de l'énergie consommée est déjà implémenté dans GloMoSim, ceci en se basant sur le model NCR wavelan [WAV00]. La formule de l'énergie consommée dans la batteries d'un nœud i ($Power_consumed_i$) est:

$$Power_consumed_i = \sum_{transmitted\ packet\ i} trans_delay * (RTR * p/p_{max} - RSR) + \sum_{received\ packet\ i} trans_delay * (RRR - RSR) + RSR * (radio_turnOffTime - radio_turnOnTime) \dots\dots\dots (1)$$

Tel que :

transmitted packet_i : est l'ensemble des paquets envoyés par le nœud i

received packet_i : est l'ensemble des paquets reçus par le nœud i

trans_delay = packetSize / bandwidth + synchronizationTime

packetSize : c'est la taille du paquet

bandwidth : la bande passante

synchronizationTime : constante égale à 192 micro seconde

RSR (Radio_Sleep_Rate) = 0.18 / second

RRR (Radio_Receive_Rate) = 1.48 / second

RTR (Radio_Transmit_Rate) = 3 / second

Radio_turnOnTime : le temps de la mise en marche de l'interface réseaux (début de la simulation)

radio_turnOffTime : le temps d'arrêt de l'interface réseaux (fin de simulation).

P: la puissance de transmission

p_{max} : la puissance maximale, elle égale 72.321 mW

La première partie est la partie consommée pour la transmission des paquets. La second est la partie consommée pour la réception. Ces deux parties ensemble, forment l'énergie consommée pour la communication. Quant à la dernière partie, elle représente l'énergie consommée en mode oisif (IDLE). Elle ne dépend que du temps de mise en marche de l'interface(temps de simulation). Le protocole de routage n'influe pas sur cette dernière partie. Toutefois, elle peut être diminuée si on utilise une technique qui permet d'éteindre l'interface réseau [BND02]. Comme aucun protocole implémenté ne permet d'utiliser cette technique, cette partie sera la même quelque soit les protocoles sélectionnés. Nous nous intéressons qu'aux deux premières pour comparer les protocoles. Ainsi les métriques mesurées, représentant l'énergie sont :

a.1 Energie moyenne de communication: c'est l'énergie moyenne des nœuds, consommée pour l'envoi et la réception des paquets. Elle est donnée par la formule suivante :

$$E_{com} = [\sum_{i=1}^n [\sum_{transmitted\ packet\ i} trans_delay * RTR * p/p_{max} + \sum_{received\ packet\ i} trans_delay * RRR]] / n \dots\dots\dots (2)$$

Où n est le nombre de nœuds.

Cette énergie est formée des deux énergies suivantes:

a.2 Énergie moyenne de transmission: elle est donnée par la formule suivante :

$$E_{Tx} = \left[\sum_{i=1}^n \sum_{\text{transmitted packet}} (\text{trans_delay} * (RTR) * p/p_{\max}) \right] / n \dots \dots \dots (3)$$

a.3 Énergie moyenne de réception :

$$E_{Tx} = \left[\sum_{i=1}^n \sum_{\text{received packet}} \text{trans_delay} * (RRR) \right] / n \dots \dots \dots (4)$$

b.duré de vie des batteries: le protocole de routage le plus efficace est celui qui; D'une part, augmente la durée de vie moyenne des batteries, et d'autre part, assure que les nœuds restent en vie ensemble le plus longtemps possible. Ceci en minimisons l'écart entre les durées de vie des batteries des nœuds. Ainsi nous proposons les deux métriques suivantes:

b.1 durée de vie moyenne des batteries : c'est le temps moyen de décharge des batteries, formellement :

$$BL_{avg} = \sum_{i=1}^n BL_i / n \dots \dots \dots (5)$$

BL_i : est le temps de la décharge de la batterie du nœud i .

La minimisation de l'énergie moyenne consommée doit, en principe, maximiser cette métrique.

b.2 écart entre les durées de vie des batteries: c'est la différence entre la durée de vie maximale et minimale. Formellement :

$$BL_{dif} = \text{Max}(BL_i) - \text{Min}(BL_i) \dots \dots \dots (6)$$

Ainsi le protocole efficace doit, à la fois, maximiser BL_{avg} , et minimiser BL_{dif} .

2.3 Paramètres et modèles utilisés

Nous avons simulé un réseau ad hoc formé de 10 nœuds, dans un terrain rectangulaire $330 \times 250 \text{ m}^2$ (afin d'allonger les routes), pour un temps de 15 minutes (900S). Chaque nœud a un domaine de puissance égale à 150 m. La topologie initiale est représentée sur la figure 6.1.

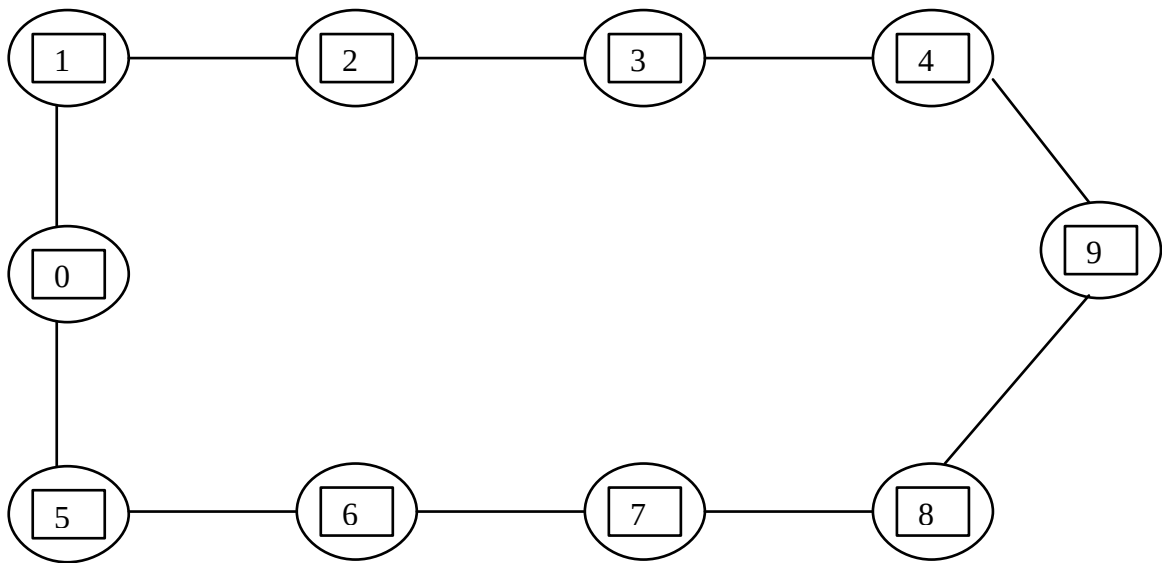


Figure 6.1 Topologie initiale du réseau

c.1 Modèle de mobilité: Parmi les modèles offerts par GloMoSim, nous avons choisi le modèle de mouvement dit WAYPOINT. C'est le modèle le plus proche du mouvement réel des nœuds: un nœud sélectionne aléatoirement une destination du terrain physique et se déplace vers cette destination, à une certaine vitesse. Lorsqu'il atteint sa destination, le nœud y reste pour une période donnée (nommée PAUSE TIME). Après cette période de repos, le nœud sélectionne une autre destination et ainsi de suite. Le pause time était fixé à 5S, et la vitesse été utilisée pour faire varier la mobilité.

c.2 Modèle de propagation : C'est le modèle FREE SPACE (propagation libre) qui suppose que le signal se propage de l'émetteur vers le récepteur sans qu'il y ait des obstacles entre eux. Le signal s'atténue en fonction de la distance entre le nœud source et le nœud destination. Ce modèle est simple car il élimine l'effet des obstacles dans la simulation. Cependant, comme on l'a déjà vu, le modèle implémenté ne prend pas en compte la puissance de transmission. Il a été modifiés en se basant sur celui présenté dans [SDTa02].

Si l'émetteur envoie les paquets à une certaine puissance P_{tx} , et si on note par P_r la puissance minimale nécessaire pour une bonne réception. Après que le signal traverse une distance d , La puissance de réception du signal sera:

$$P_{rx} = P_{tx} - (P_r * d^4 / K - P_r) \dots \dots \dots (7)$$

Tel que : $P_r = 1 \text{ mW}$ et $K = 1.4 * 10^7$

Alors pour assurer qu'un signal soit bien reçu à une distance d , la puissance de transmission minimale est:

$$P_{tx} = P_r * d^4 / K$$

Nous ajoute une certaine marge, pour surmonter un certain intervalle de mobilité : ainsi la formule devient:

$$P_{tx} = P_r * d^4 / K + P_{marg} \dots \dots \dots (8)$$

Pour une puissance de transmission donnée (P_{tx}), la distance maximale ($DistMax$) sur laquelle le signal peut être reçu est donnée par :

$$DistMax = \frac{\log(P_{tx} * K / P_r)}{\log(4)} \dots \dots \dots (9)$$

Ainsi si la distance séparant l'émetteur et le récepteur est inférieure à cette distance, le signal peut être reçu.

En fixant le domaine de puissance, la puissance maximale peut être calculée par la formule (7). Cependant, si on fixe la puissance maximale, le domaine de puissance correspondant peut être calculé par la formule (9).

c.3 Protocole de la couche MAC: Nous avons choisi comme protocole dans la couche MAC le protocole IEEE 802.11. Ce protocole est très utilisé dans les réseaux sans fil à cause des propriétés suivantes:

- Il évite les collisions en vérifiant l'état du canal avant son utilisation.
- Chaque transmission d'un paquet est suivie par un acquittement de réception. L'absence d'un acquittement après plusieurs retransmissions indique la défaillance du lien entre les deux nœuds.
- Il utilise un buffer pour sauvegarder les paquets même si le système est oisif. Cependant, les modifications décrites auparavant (paragraphe a), ont été ajoutées à l'implémentation de ce protocole.

c.4 La couche Application: parmi les applications implémentées dans GloMoSim, nous avons choisie l'application CBR. Elle fonctionne en dessus du protocole UDP, qui est très souple par rapport à TCP. Et elle est très utilisée dans les réseaux sans fil. Initialement les nœuds 0 et 9 sont éloignées (figure 1), donc nous avons choisi de lancer une session continue entre ces deux nœuds, à partir de l'instant 0 jusqu'à 5 minute. Après, nous avons remarqué que les nœuds 6 et 7 s'éloigne (pour les scénarios mobiles que nous décrirons par la suite), nous avons choisi alors une autre session entre ces deux nœuds, pendant l'intervalle 5 minutes à 15 minutes (fin de la simulation).

La taille du paquet été fixé à 1KO. Le débit été fait varié en faisant varier le temps séparent 2 envois consécutifs.

c.5 constantes du protocole de routage : Le tableau suivant synthétise les constantes de notre protocole de routage.

Temps de conservation d'une entrée dans la liste des requêtes reçues	30S
Répondre à partir de la cache	non
Période minimale entre deux envoies pour éviter les collisions	10mS
Temps entre deux transmissions d'une même requête (si aucune réponse n'est obtenue)	500 ms
Fraction d'envoi du message E_state	0.1
Temps de conservation d'une entrée dans la liste des liens rompus	1S
Energy_Threshold	0.2mWhr

Les quatre premières constantes sont utilisées par DSR, ils ont les mêmes valeurs. Les autres sont propres à DSRPA .

2.4 démarche de la simulation : nous avons évalué les performances du protocole proposé (ses quatre versions) dans différentes charges du réseau et différentes mobilités, en le comparant avec le DSR benchmark.

durant toutes les simulations, nous avons utilisé trois valeurs de mobilité. La première est la mobilité nulle (cas statique), soit Mob0. La deuxième est une mobilité moyenne, la vitesse des nœuds pour cette mobilité est 0.5 m/s, causant un changement de lien moyen de 8.00 [DDB02]. Nous la notons Mob1. La dernière est la mobilité élevée, la vitesse est 1m/s, causant un changement de lien moyen de 14.00. soit cette mobilité Mob2.

La charge du réseau est représentée par le débit de l'application CBR. Comme on l'a déjà dit, le débit est changé en changeant l'intervalle entre deux transmissions. Nous avons utilisé cinq débits différents : 1KO/S, 2KO/S, 3KO/S, 4KO/S, et 5KO/S.

Les métriques de comparaison présentées précédemment ont été mesurées dans toutes les situations des débits et de mobilités (et même de charges de batteries qu'on va voir par la suite). Donnant lieu à 225 scénarios (exécutions) différents. Les métriques mesurées ont été divisées en deux classes. Ainsi la simulation est composée des deux phases principales suivantes :

1 mesure de l'énergie : pour cette phase nous avons mesuré l'énergie consommée dans les différents cas. Nous avons fixé la charge des batteries à une valeurs supérieur à la consommation maximale, soit 300mWhr. Nous avons aussi fixé la valeur de P_{marg} (la puissance marge ajoutée au puissance de transmission) à 0.1mW pour Mob0, 6.1 pour Mob1, et 5.2 pour Mob2. Ces valeurs été déterminés d'une manière empirique.

2 mesure des durées de vie: elle se compose de deux phases :

2.1 par rapport au débit : la charge des batteries été fixée à 160mWhr, et nous avons mesuré les deux métriques concernant les durées de vie, pour les différents débits et mobilités.

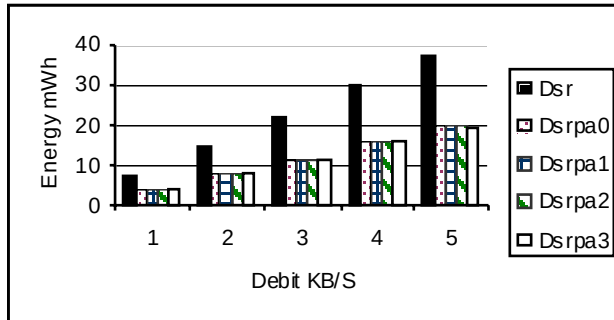
2.2 par rapport à la charge des batteries : pendant cette phase, le débit été fixé à un débit moyen, soit 3KO/S. Les deux métriques ont été mesurés dans les trois types de mobilité. En faisant varier la charge des batteries, de 40mWhr à 200mWhr, en Passant par 80mWhr, 120 mWhr, et 160mWhr. L'objectif de cette partie est d'étudier l'effet de

la charge des batteries sur les performances du protocole. Surtout sur la différence entre les durées de vie.

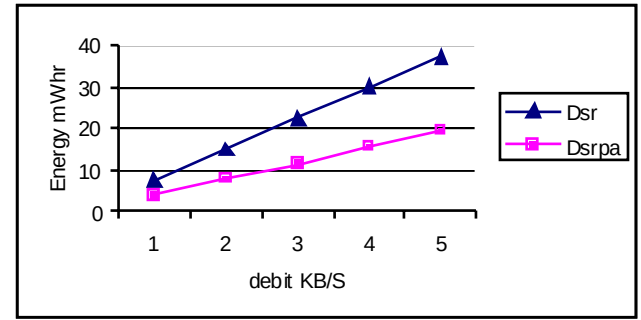
III. Résultats

1 Energie consommée

a) Energie de communication :

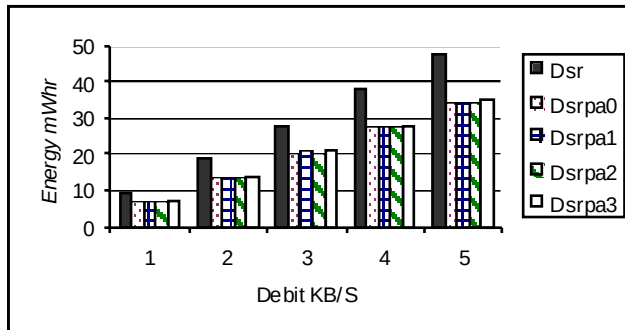


A

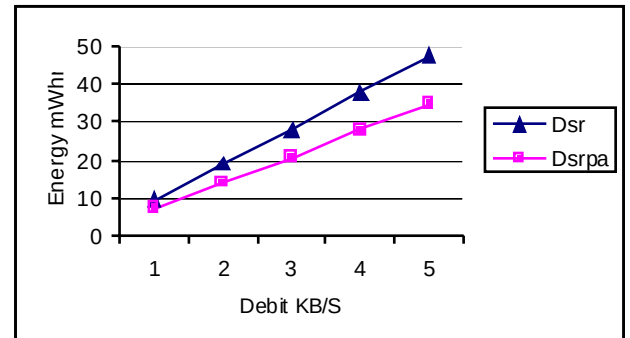


B

Mob0

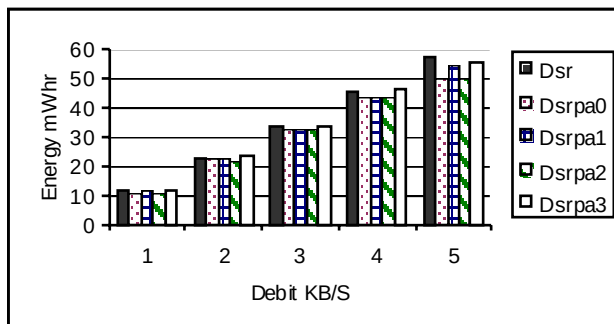


C

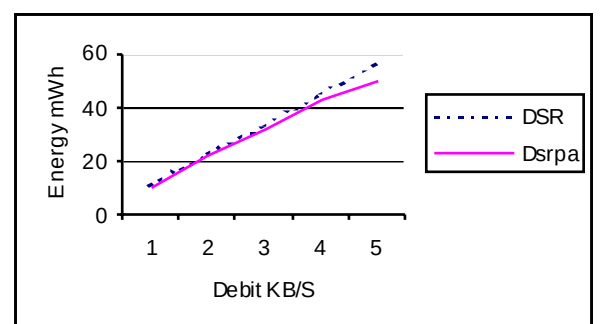


D

Mob1



E



F

Mob 2

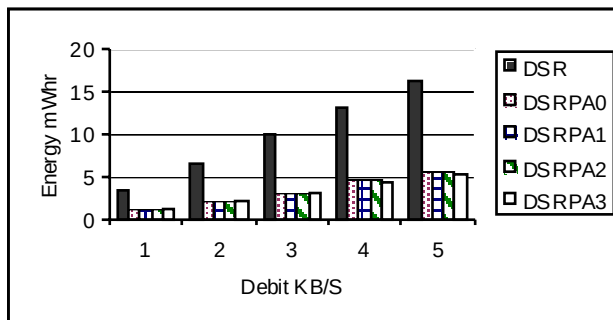
Figure 6.2 Energie de communication

Les histogrammes des figures A, C et E montrent l'énergie consommée (en mWhr) des quatre versions proposées et celle de DSR standard. A partir des trois autres, on peut analyser l'écart de la consommation entre DSR et notre protocole. En fait, les différentes versions ont des valeurs très proches les unes des autres. nous avons pris la version qui a les meilleurs performances, pour les courbes de ces dernières figures. On remarque dans la figure A, que DSRPA consomme toujours moins d'énergie que DSR. Et que tous les versions ont, pratiquement, la même consommation. On remarque la même chose pour la mobilité Mob1 (figure C). Mais pour Mob 2, on remarque que les versions utilisant l'option All ROUTES (DSRPA1 DSRPA3) consomment un peu plus d'énergie que les deux autres. Cependant, ils ont tous des consommation inférieurs à celle de DSR.

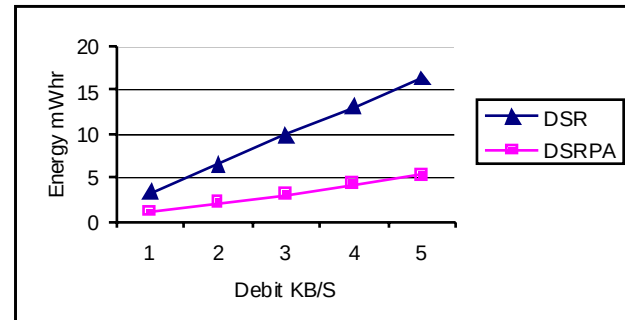
En analysant les courbes des figures B, D et F, on remarque que l'écart entre la consommation de DSR et DSRPA diminue avec l'augmentation de la mobilité. Mais avec l'augmentation du débit, la consommation de DSR augmente plus rapidement que DSRPA, et donc l'écart augmente. Ceci est valable même pour Mob2.

DSRPA donne un gain important en consommation d'énergie, surtout quand le débit augmente. Nous donnerons les causes de ce gain, en analysant les deux parties de cette énergie.

b) Energie de transmission:

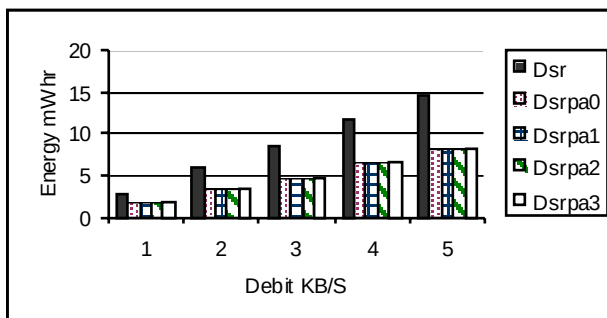


A

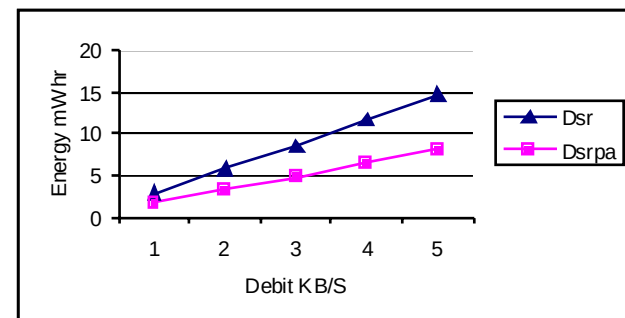


B

Mob 0



C



D

Mob 1

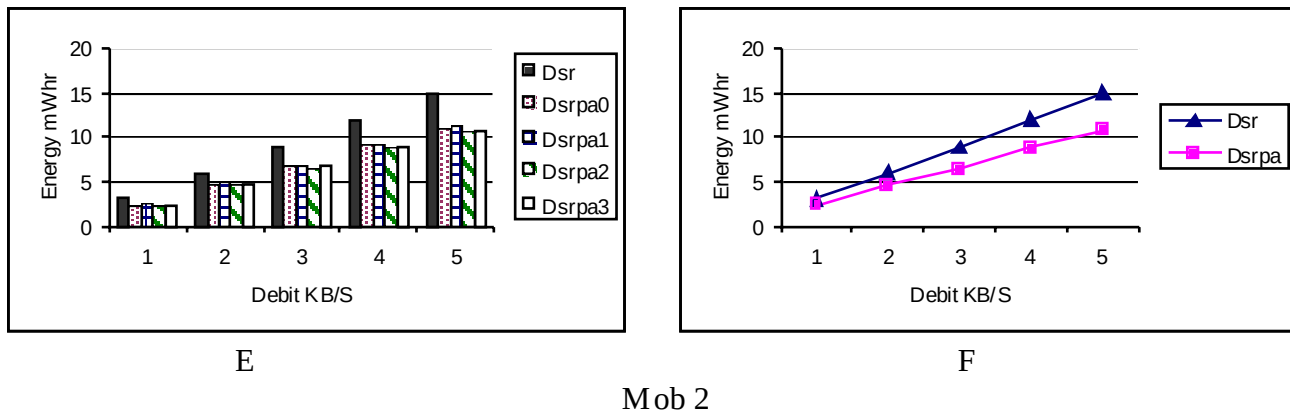


Figure 6.3 Energie de transmission

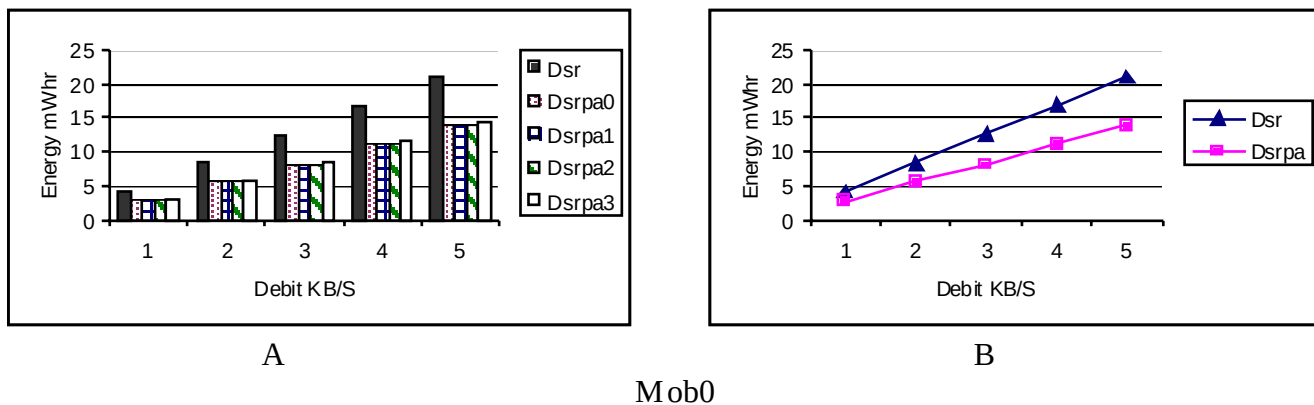
Ces courbes ont pratiquement les mêmes formes que les précédentes, les seules différences sont :

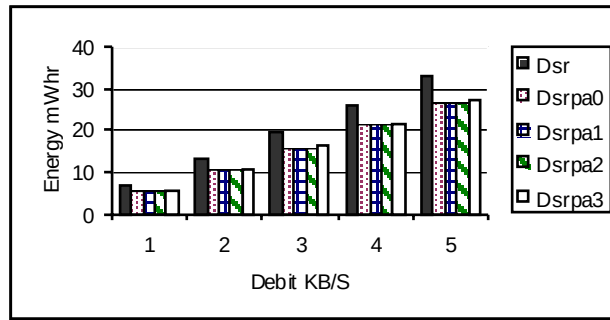
- L'écart entre DSR et DSRPA est plus important, surtout pour Mob 2.
- Les versions qui utilisent l'option ALL ROUTES, ne consomment pas plus d'énergie que les deux autres. Au contraire, la version DSRPA3 (utilisant les options : ALL ROUTES et la fonction racine pour le paramètre alpha), consomme légèrement moins d'énergie, quand le débit augmente.

Ce gain est dû principalement à l'utilisation de la technique de contrôle de puissance de transmission. Ainsi aux métriques utilisées.

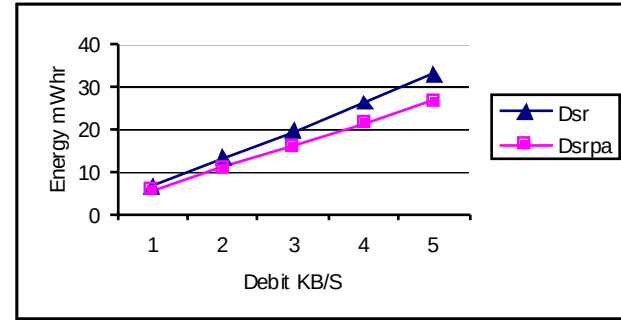
Le gain de DSRPA3 est dû à l'option All ROUTES qui lui permet d'avoir plus de routes par rapport à DSRPA0 et DSRPA2, et de choisir le plus optimal. Et à l'utilisation de la fonction racine pour le paramètre alpha, qui lui permet de choisir des routes plus optimaux par rapport à DSRPA1.

c) Energie de réception:



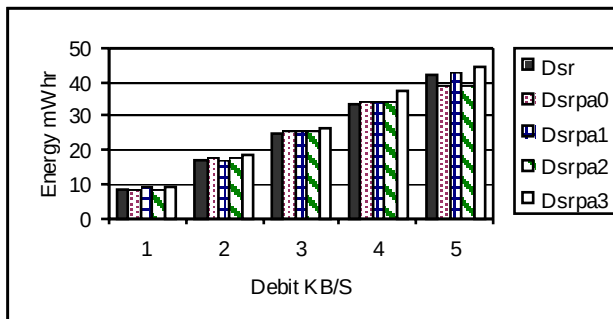


C

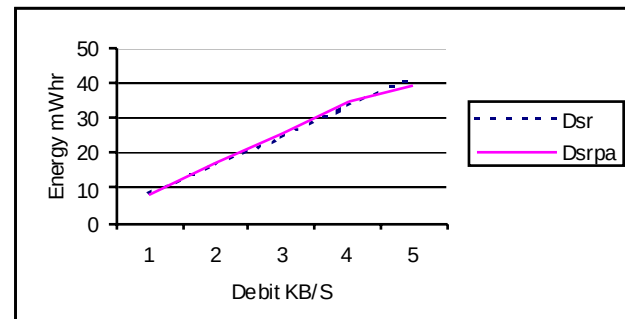


D

Mob1



E



F

Mob2

Figure 6.4 Energie de réception

On remarque que les courbes représentant l'écart entre DSRPA et DSR ont les mêmes formes que les précédentes pour les mobilités Mob0 et Mob1. Cependant, pour la mobilité Mob 2 (courbe F), on remarque que l'énergie consommée par DSR est un peu petite que celle consommée par DSRPA entre les débit 1MO/S et 4MO/S. Elle devient un peu plus grande pour le débit 5 MO/S.

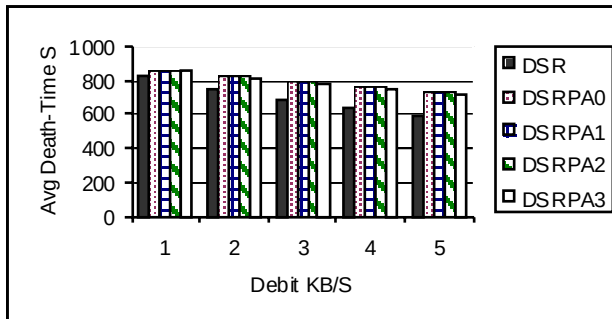
D'une part, DSR utilise le mode promiscues. Dans ce mode, l'interface réseaux reçoit tous les paquets quelle entend quand elle est en état de veille. Ceci n'est pas utilisé par DSRPA. Ce qui donne un gain à DSRPA. Et d'autre par, la mobilité nécessite plus de maintenance, surtout pour DSRPA qui utilise des puissances adaptatives. En d'autre terme, la mobilité peut rendre les puissances utilisées sur les liens insuffisantes, ce qui cause une maintenance pour DSRPA et non pas pour DSR qui utilise des puissances fixe. Cette maintenance est effectuée en envoyant plus de paquets de contrôles (paquets de requêtes et de réponse dirigés, et surtout paquets d'erreurs). Ceci causera plus de consommation pour DSRPA, ce qui explique la différence en faveur de DSR pour Mob2 quand le débit est inférieur 4MB/S. Ce phénomène n'est présent que pour cette partie, et non pas dans l'énergie de transmission. Car pour cette dernière, la légère consommation supplémentaire

causée par DSRPA pour la maintenance est largement compensée par la technique de contrôle de puissances de transmissions.

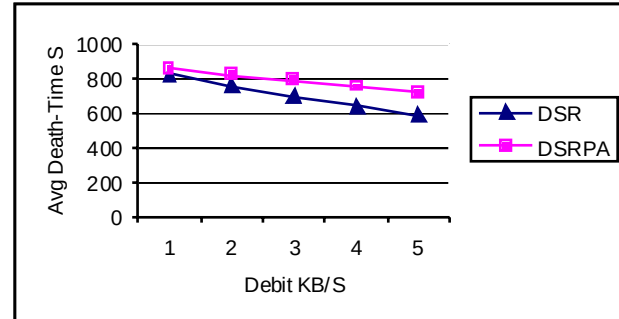
Dans les figures A, C et E, on remarque que les version de DSRPA ont des valeurs de consommation très proches. Mais, on remarque que les versions utilisant l'option ALL ROUTES (DSRPA1 et DSRPA3) Consomment un peu plus d'énergie pour la mobilité Mob2. Cette différence est dû à la propagation des requêtes, qui est plus large dans ces versions. Ceci explique aussi la différence observée pour l'énergie de communication.

2 Durée de vie des batteries par rapport au débit

a) durée de vie moyenne

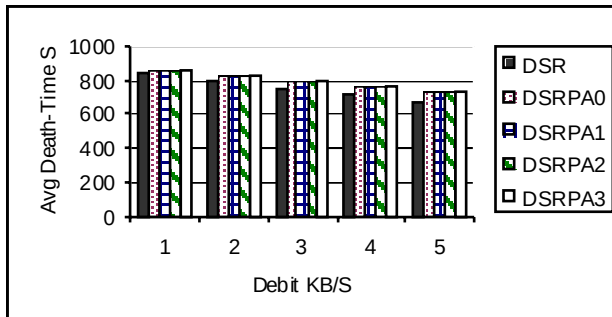


A

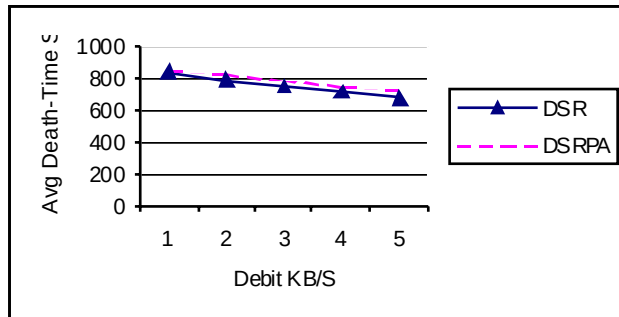


B

Mob0

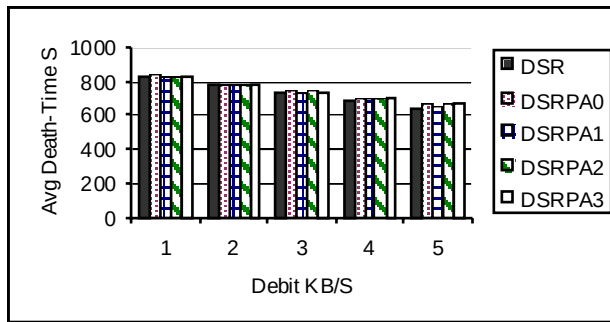


C



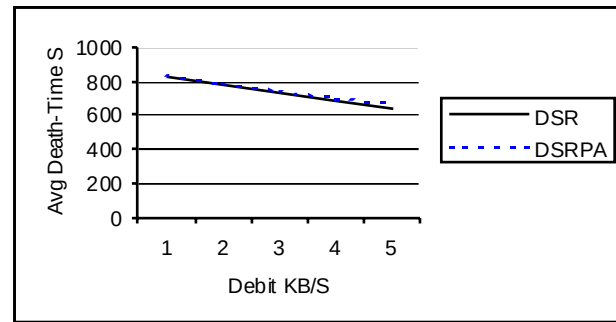
D

Mob1



E

Mob2

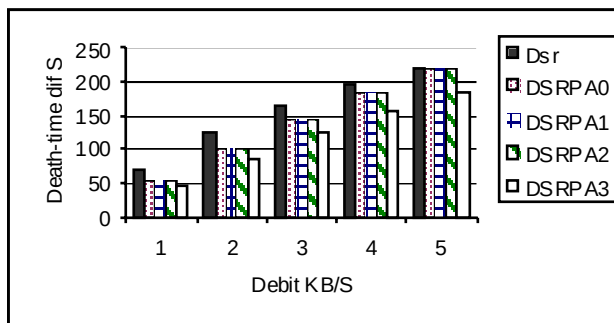


F

Figure 6.5 durée de vie moyenne

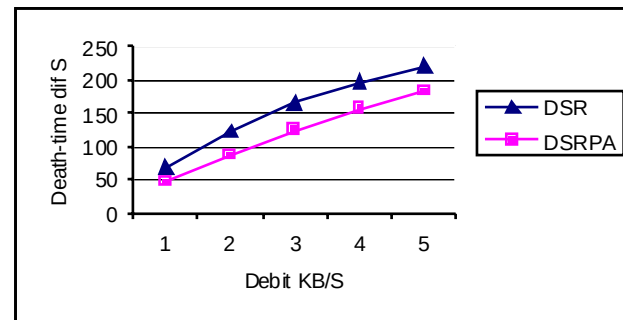
On Remarque des figures A, C et E que Les versions de DSRPA ont pratiquement les mêmes durées de vie moyennes dans tous les débits et pour toutes les mobilités. Et que leurs durées de vie sont toujours meilleurs que celles de DSR. On remarque aussi (à partir des courbes des figures B, D, et F) que la durée de vie de DSRPA, diminue moins rapidement que DSR, surtout pour le cas statique (courbes de la figure B). On Dédduit que l'écart entre DSR et DSRPA augmente avec l'augmentation du débit, et diminue avec l'augmentation de la mobilité. Cette amélioration de la durée de vie moyenne est dû au gain d'énergie, vue précédemment.

b) Ecart entre les durée de vie des batteries

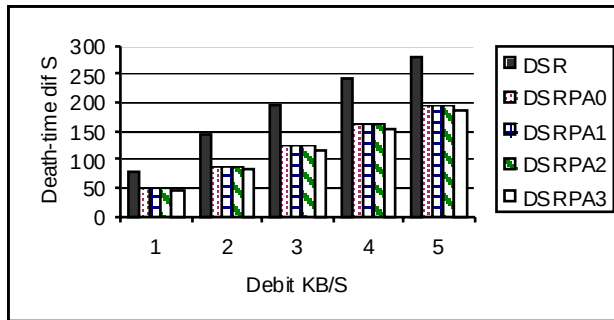


A

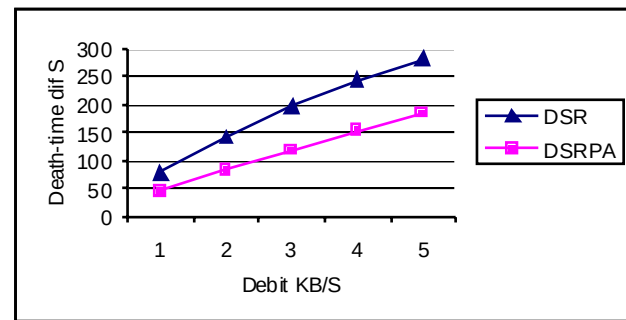
Mob0



B

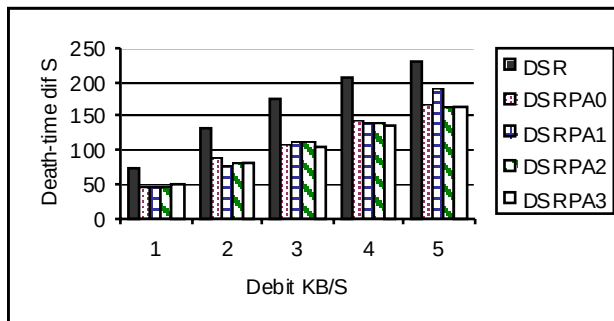


C

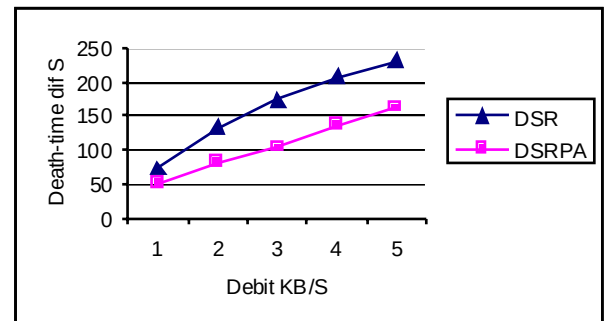


D

Mob1



E



F

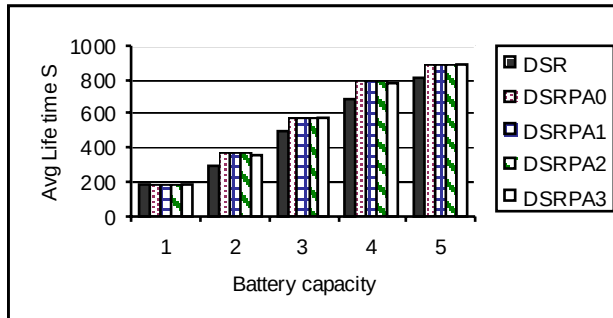
Mob2

Figure 6.6 Ecart entre les durées de vie des batteries

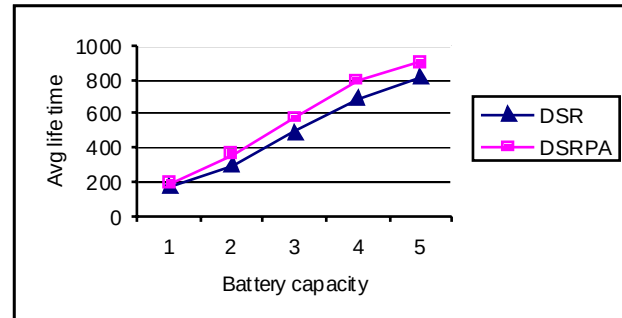
A partir des figures B, D, et F, on remarque que l'écart des durées de vie augmente avec le débit. On remarque aussi, que l'écart entre DSR et DSRPA augmente (en faveur de DSRPA) avec le débit. Il augmente aussi avec la mobilité, contrairement à la durée de vie moyenne. Cette performance est due principalement aux métriques utilisées. On remarque dans les figures A, C et E, que DSRPA3 a souvent la meilleure performance (le plus petit écart). Et que les valeurs des autres versions sont généralement proches. On déduit que la fonction racine et l'option ALL ROUTES, minimisent mieux l'écart entre les durées de vies des batteries, tout en améliorant la durée de vie moyenne .

3 Durée de vie des batteries par rapport à la charge des batteries

a) durée de vie moyenne

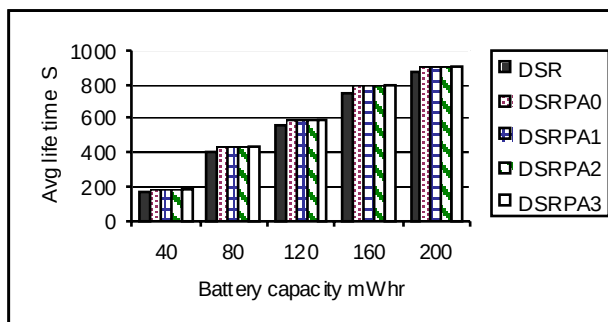


A

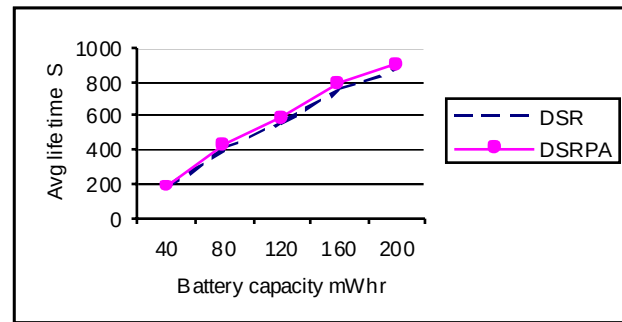


B

Mob0

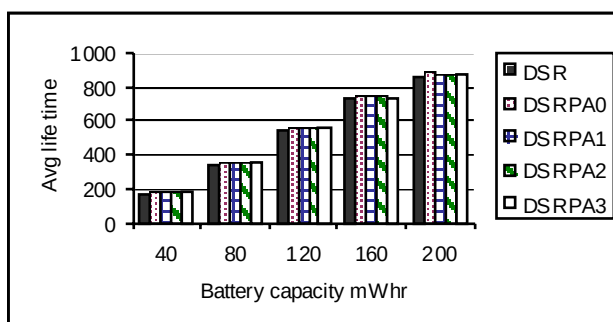


C

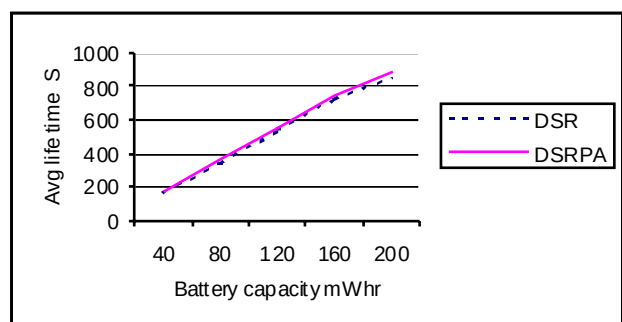


D

Mob1



E



F

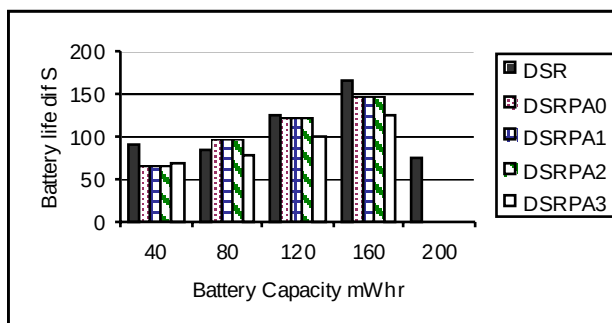
Mob2

Figure 6.7 durée de vie moyenne / capacité de batterie

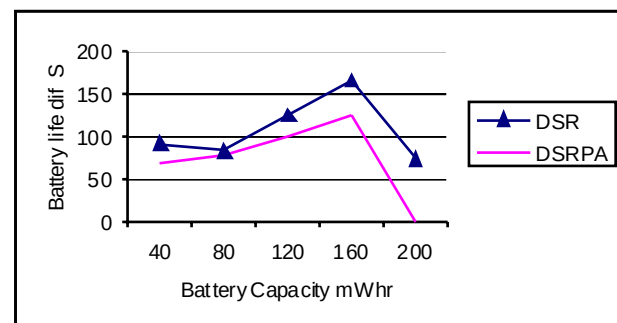
Comme on peut l'observer, à partir des figures A, C, et E, toutes les versions de DSRPA ont toujours des durées de vie meilleurs que DSR. Ces versions, ont des durées de vie presque égales dans tous les cas. On remarque aussi, que les durées de vie augmentent avec l'augmentation de la charge.

Dans les trois autres figures, on remarque que l'augmentation de la mobilité, cause une diminution de l'écart entre DSRPA et DSR. Quant à l'augmentation de la charge, elle cause une augmentation de l'écart, et donc plus de performance en faveur de DSRPA.

b) écart entre les durées de vie des batteries

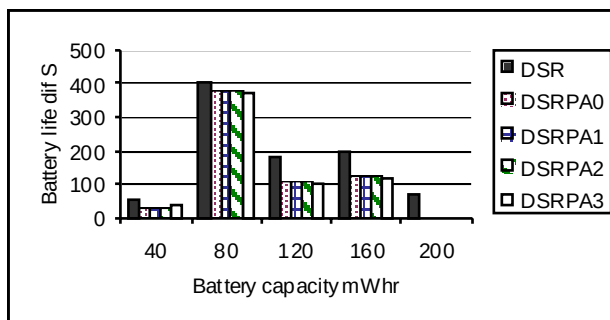


A

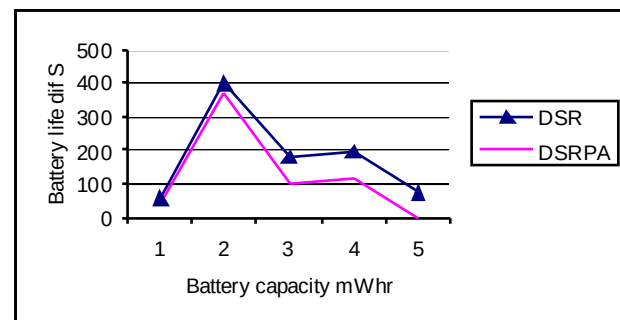


B

Mob0



C



D

Mob1

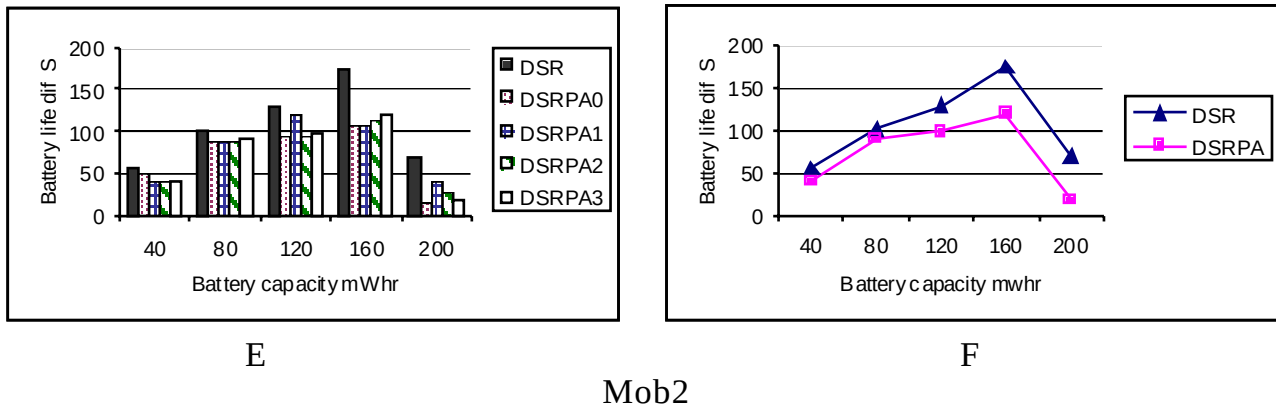


Figure 6.8 écarts entre les durées de vie des batteries

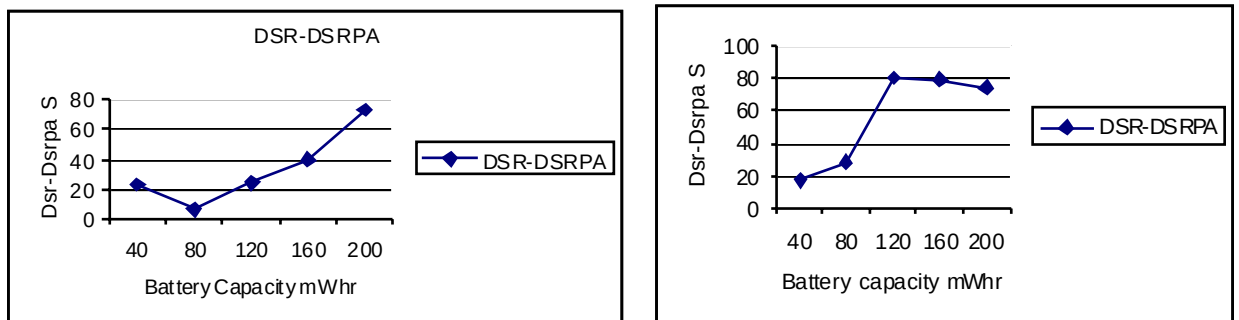
Dans les figures A et C, on remarque que DSRPA3 a toujours l'écart minimale. Et que toutes les versions de DSRPA ont souvent des écarts inférieurs à DSR. Leurs écarts est nul pour la charge 200 mWhr, ceci parce que la durée de vie de toutes les batteries est 900 S (temps totale de la simulation), c'est à dire qu'aucune batterie n'a été déchargée.

Dans la figure E, on remarque que les versions de DSRPA ont toujours des écarts inférieurs à DSR. Et que DSRPA0 et DSRPA3 ont les écarts minimaux. Ils sont presque nuls pour la charge 200mWhr.

La seule remarque à noter des autres courbes (figures B, D, F), est la diminution rapide de l'écart entre les charges 160mWhr et 200 mWhr. Cependant, ces courbes ne montrent aucune monotonie, il est donc difficile d'observer la différence entre DSR et DSRPA. Alors nous proposons les courbes de la figure 6.9, montrant cette différence.

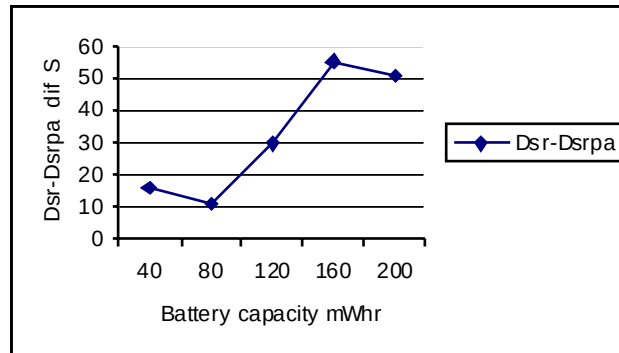
La courbe A montre une légère baisse entre 40mWhr et 80mWhr, et une croissance importante à partir de 80mWhr. La seconde montre une monotonie; une croissance entre 40 mWhr et 120mWhr, et une stabilité à partir de 120mWhr. La dernière montre une croissance considérable entre 80mWhr et 160mWhr, et une diminution négligeable ailleurs. Ces courbes sont généralement, croissantes. Et les valeurs pour les charges élevées (160mWhr et 20Mwhr) sont très importantes, contrairement aux petites charges (40mWhr et 80mWhr).

On déduit que la différence entre DSR et DSRPA est plus importante pour les charges élevées que les faibles charges.



A (Mob0)

B (Mob1)



C (Mob2)

Figure 6.9 différence entre DSR et DSRPA pour l'écart des durées de vie

IV. Conclusion

Les principales conclusions de ce travail sont:

- L'énergie moyenne consommée reflète bien la durée de vie moyenne. Mais pas l'écart entre les durées de vie des batteries.
- la durée de vie moyenne, et l'écart entre les durées de vie des batteries représentent ensemble la durée de vie du réseau.
- Le protocole que nous avons proposé (DSRPA) montre; un gain important en consommation d'énergie, une augmentation des durées de vie moyenne des batteries, et une diminution de l'écart entre les durées de vie des batteries. Il est plus performant que DSR benchmark dans toutes les conditions, et assure alors une plus longue durée de vie aux batteries des nœuds, et au réseau.
- La version utilisant l'option ALL ROUTES et la fonction racine carrée pour le paramètre alpha, donne généralement les écarts minimaux entre les durées de vie des batteries, et elle est la plus performante.
- Le protocole montre plus de performance (par rapport à DSR), dans les situations où le débit est élevé.
- Le protocole montre de meilleures performance dans les cas des charges élevées (relativement aux charges utilisées durant la dernière phase). Nous notons que les charges des batteries réelles sont plus élevées que les charges utilisées (environ 11 fois plus élevées que la charge maximale et 55 fois plus élevées que la charge minimale) [DSD02, DAW01]. Pour utiliser des charges réelles, il faut augmenter considérablement la durée de simulation.

Bibliographie

- [AGG98]: George Aggelou and Rahim Tafazolli.
"RDMAR : A bandwidth-efficient routing protocol for mobile ad hoc Networks".
Proceedings of the ACM International Workshop on Wireless Mobile Multimedia
(WoWMoM), Seattle, WA, August 1999, pp. 26-33
- [AOL03] : <http://www.viewz.com/aol/guides/pda3.shtml>
- [ASC02]: Akis Spyropoulos and C.S. Raghavendra, "Energy Efficient
Communications in Ad Hoc Networks Using Directional Antennas", Department of
Electrical Engineering-Systems University of Southern California Los Angeles, IEEE
INFOCOM, juin 2002
- [BAD98]: Nadjib Badache, "La mobilité dans les systèmes répartis",
Janvier 1998, Technique et science informatiques. Volume 17-n° 8, 1998, pp
969-997
- [BAS98]: S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward.
"A distance routing effect algorithm for mobility (DREAM)". In Proceeding of
ACM/IEEE MOBICOM'98, Dallas, Texas, pp 76-84, october 1998.
- [BND02]: Badache Nadjib, djenouri Djamel.
« Etude de la consommation d'énergie causée par les protocoles de routage dans les
réseaux ad hoc ». Rapport de mini projet de magister, département d'informatique,
université des science et de la technologie Houari Boumediene (USTHB), Alger,
Algérie.
- [BTS01]: Brown, T.X, Doshi, S., Zhang, Q.
"Optimal power aware routing in a wireless ad hoc network," , *IEEE LANMAN 2001
Workshop roceedings*, pp. 102–105.
- [CAB97]: ", C. A. Balanis
"Antenna Theory: Analysis and Design" 2nd. New York: Wiley, 1997.
- [CHE98]: Tsu-Wei Chen and Mario Gerla. "Global State Routing: A new routing
scheme for ad-hoc wireless networks". Proceedings of the IEEE International
Conference on Communications (ICC), Atlanta, GA, June 1998, pp. 171-175.
- [CHI97]: C. C. Chiang, H-K Wu, Winston Liu, and Mario Gerla.
"Routing in clustered multihop, mobile wireless networks". The IEEE
Singapore International Conference on Networks, pp 197-211, 1997.
- [COM03]: "computerword week in review newsletter",
<http://careers.computerworld.com/home/features.nsf/all/>
- [CON02]:
<http://homepage.cs.uri.edu/courses/csc101sec1/CReports/Laptops,%20features%203-02.htm>
- [COR95] : Corson, M.S.; Ephremides, A.
"A distributed routing algorithm for mobile wireless networks.", *Wireless Networks*,
vol.1, (no.1), p.61-81. 1995
- [CPP94]: Charles. E. Perkins and Pravin Bhagwat
"Highly dynamic destination-sequenced distance-vector routing (DSDV) for
mobile computer". ACM SIGCOMM'94 Conference on Communications
Architectures, Protocols and Applications, pp. 234-244, 1994.

Bibliographie

- [DDB01]: Djenouri Djamel, Derhab Abdelwahid,
«Simulation des algorithmes de routage dans les mobile ad hoc », thèse d'ingénieurs,
Université des sciences et de la technologie Hoari Boumedienne USTHB, juin 2001.
- [DDB02]: Nadjib BADACHE, Djamel DJENOURI, Abdelouahid DERHAB
“Analyse comparative des algorithmes de routage dans les réseaux mobiles ad hoc”,
département d'informatique, USTHB, Alger, Algérie, publication interne: LSI-TR-
0702
- [DDB03]: Nadjib Badache, Djamel Djenouri, Abdelouahid Derhab.
“Mobility Impact on mobile ad hoc routing protocols”, *ACS/IEEE International
Conference on Computer Systems and Applications (AICCSA'03)*, Tunis 2003.
- [DDN92]: D.Duchamp and N.F
“Measured performance of Wireless LAN”. Reynolds , In 17th Conference on
Local Computer Networks, pages 494--499, Minneapolis, Minnesota, September
1992. IEEE
- [DUB97] : R. Dube et al.
"Signal stability based adaptive routing for ad hoc mobile networks". IEEE Pers.
Comm., pp 36-45, February 1997
- [DSD02]: Daler Rakhmatov, Sarma Vrudhula, Debrah A.Wallach.
“Battery Lifetime Prediction for Energy-Aware Computing”, International
Symposium on low power electrics and design 2002 session 6, pages 154-159,
ISBN:1-58113-475-4
- [ERA 96] : Pierre-Jean Erard, Pontien Déguénon
« Simulation par évènements discrets » presses polytechnique et universitaire
Romandes, 1996 1^{ière} édition
- [ERC99]: ", Elizebeth M. Royer , C-K Toh
“A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks”,
IEEE Personal Communications, pages 46-55, Apr 1999.
- [FOR94] : G.H Forman and J.Zahrojan
“The challenge of mobile computing” IEEE Computer, 27 (4), pp38-47, April 1994
- [FRA03]: www.Streettech.com
- [GAA97]: G. Asada *et al.*, *Proc.*
“Low power wireless communication and signal processing circuits for distributed
microsensors” 1997 *IEEE Int'l. Symp. Circuits and Sys.*, Hong Kong, June 9–12,
1997, pp. 2817–20.
- [GJA96]: J. Goodman et al
“Signal Processing for an Ultra-Low-Power Wireless Video Camera,” *Proc. 3rd Int'l.
Wksp. Mobile Multimedia Commun*, Princeton, NJ, Sept. 25–27, 1996, pp. 267–74.
- [IET02]: <http://www.monarch.cs.rice.edu/ietf.html> IETF MANET Working Group,
21 February 2002
- [IWA99]: Iwata, C. C. Chiang, G. Pei, M. Gerla, and T.-W. Chen.
"Scalable routing strategies for ad hoc wireless networks". IEEE Journal on
Selected Areas in Communications, Special Issue on Ad-Hoc Networks,
pp.1369-1379, August 1999.
- [JBA02] : Jean-Pierre Ebert, Brian Burns, and Adam Wolisz, *Technical University
Berlin*
“A trace-based approach for determining the energy consumption of a WLAN
network interface”, *Telecommunications Networks Group Sekr. FT5-2 – Einsteinufer
25 – 10587 Berlin – Germany, february 2002*
- [JDM96]: Johnson, D., Maltz, D.

Bibliographie

- “Dynamic Source Routing in Ad Hoc Wireless Networks,”. *Mobile Computing*, Chapter 5, pp. 153-181, Kluwer Academic Publishers, 1996.
- [JHM98]: Jaap Haartsen Warren Allen, Jon Inouye, Olaf J. Joeressen, Mahmoud Naghshineh
“Bluetooth: Vision, Goals, and Architecture”, *Mobile Computing and Communications Review*, Volume 1, Number 2, 1998
 - [JIA99] : Mingliang Jiang, Jinyang Li, Y.C. Tay.
"Cluster based routing protocol", IETF Draft, August 1999.
<http://www.ietf.org/internet-drafts/draft-ietf-manet-cbrp-spec-01.txt>.
 - [JOA99]: M. Joa-Ng and I.-T. Lu.
"A Peer-to-Peer zone-based two-level link state routing for mobile ad hoc networks". IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks, pp 1415-1425, August 1999.
 - [KO98]: Y.-B. Ko and N.H. Vaidya.
"Location-aided routing (LAR) in mobile ad hoc networks". In Proceedings of ACM/IEEE MOBICOM'98, Dallas, Texas, pp 66-75, October 1998
 - [KOV98]: Y.-B. Ko and N.H. Vaidya.
"Location-aided routing (LAR) in mobile ad hoc networks". In Proceedings of ACM/IEEE MOBICOM'98, Dallas, Texas, pp 66-75, October 1998
 - [KWJ01] : Kui Wu, Janelle Harms
“performance study of a multipath Routing Method for Wireless Mobile Ad hoc Networks”, Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'01)
August 15 - 18, 2001
Cincinnati, Ohio.
 - [LEM00]: Tayeb Lemlouma
« Le routage dans les réseaux mobiles Ad Hoc », mini projet de magister, Université des sciences et de la technologie Houari Boumedienne USTHB, USTHB, octobre 2000.
 - [LFM01]: L. Buttyán, J-P. Hubaux
“Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks”, *Technical Report No. DSC/2001/046*, Swiss Federal Institution of Technology, Lausanne, 31 August, 2001. <http://icawww.epfl.ch/hubaux/>
 - [LWI99]: The Editors. Part 11, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY): Higher Speed Physical Layer (PHY) Extension in the 2.4 GHz band”. IEEE Standard 802.11b, Institut of Electrical and Electronics Engineers, Inc, September 1999.
 - [MFP00]: Magnus Frodigh, Per Johansson and Peter Larsson
“Wireless ad hoc networking, The art of networking without a network“, Ericsson Review No. 4, 2000, pp 248-263
 - [MUR96]: S. Murthy and J.J. Garcia-Luna-Aceves.
"An efficient routing protocol for wireless networks". ACM Mobile Networks and Application Journal, Special Issue on routing in mobile Communication Networks, pp183-197, October 1996
 - [NOK0]: <http://www.nokia.com>
 - [NOK1]: <http://www.busenesessweek.com>
 - [PAR03]: <http://pcl.cs.ucla.edu/projects/parsec/>
 - [PAR97]: V. D. Park and M. S. Corson.
"A highly adaptive distributed routing algorithm for mobile wireless networks". Proceeding INFOCOM '97, pp 1405-1413, April 1997.

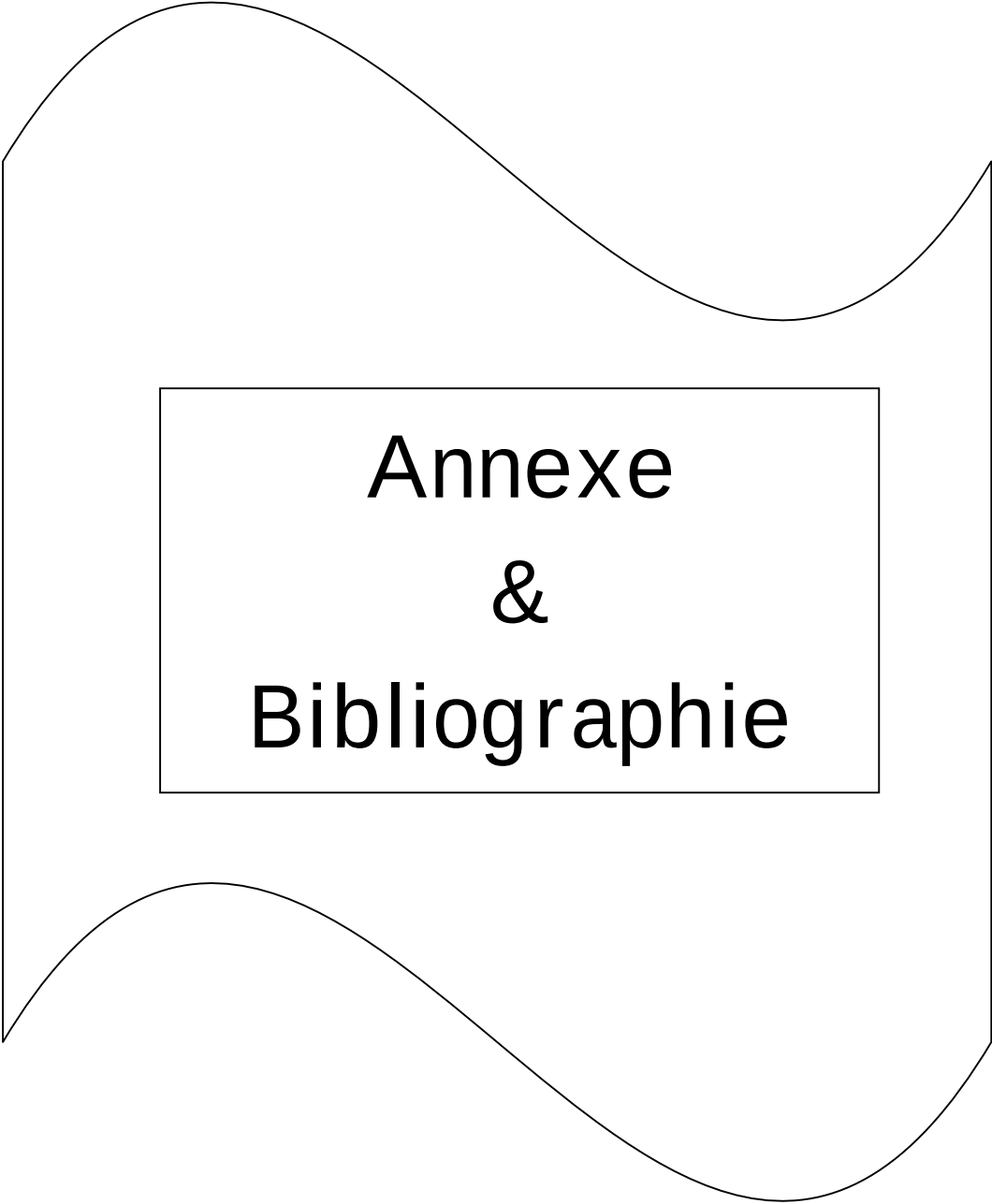
Bibliographie

- [PCW03]: <http://www.pcword.com>
- [PEI00]: Guangyu Pei Mario Gerla Tsu-Wei Chen
"Fisheye State Routing in Mobile Ad Hoc Networks". Proceedings of the IEEE International Conference on Communications (ICC), New Orleans, LA, June 2000, pp:70-74.
- [PER01]: Charles Perkins,
"ad hoc networking", edition Wesley copyright 2001
- [PER94]: Charles. E. Perkins and Pravin Bhagwat.
"Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computer". ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications, pp. 234-244, 1994.
- [PER99]: Charles. E. Perkins and Elizabeth M. Royer.
"Ad hoc on demand distance vector (AODV) algorithm". In Proceeding of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99), page 90-100, 1999.
- [PIS80]: N.Piskonov,
"calcul différentiel et intégrale", tome 2, 9^{iem} édition, édition Mir. Moscou 1980
- [PLB98]: P. Lettieri, A. Boulis, and M. B. Srivastava,
"Design of Adaptive Wireless Terminals," 1998 Int'l. Symp. Sys., Signals, and Elect., Pisa, Italy, Sept 29 to Oct. 2, 1998.
- [PLM99]: Paul Lettieri and Mani B. Srivastava University of California at Los Angeles
«Advances in Wireless Terminals», *IEEE Personal Communications*, February 1999
- [ROY99]: Elizebeth M. Royer , C-K Toh
"A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks", *IEEE Personal Communications*, pages 46-55, Apr 1999.

- [SDTa02]: Sheetal Kumar Doshi, Timothy X Brown
"Design Considerations for an On-Demand Minimum Energy Routing Protocol for a Wireless Ad Hoc Network", Department of Electrical Engg. University of Colorado at Boulder, ICC, April 2002.
- [SDTb02]: Sheetal Kumar Doshi, Timothy X Brown
"Minimum Energy Routing Schemes for a Wireless Ad Hoc Network", *IEEE INFOCOM 2002*
- [SGO97]: S. Gold
"A PSPICE Macromodel for Lithium-Ion Batteries", The 12th Annual Battery Conference on Applications and Advances, California State Univ, Long Beach, CA, Jan 14-17 1997.
- [SSM98]: Suresh Singh, Mike Woo and C. S. Raghavendra. "Power-Aware Routing in Mobile Ad Hoc Networks", Proceedings of the fourth annual ACM/IEEE international conference on Mobile computing and networking, p.181-190, October 25-30, 1998, Dallas, Texas, United States
- [TOH96]: Chai-Keong Toh.
"A novel distributed routing protocol to support ad hoc mobile computing". Proceeding 1996 IEEE 15th Annual Int'l. Phoenix Conf. Comp. and Commun, pp 480-86, March 1996.
- [WAV00]: Lucent technologies, "wavelan product specification", www.wavelan.com, 2000

Bibliographie

- [ZEN98] : X.Zeng, R.Bagrodia and M.Gerla.
“GloMoSim: A library for the parallel simulation of large-scale wireless networks”,
proceeding of the 12th Workshop on Parallel and distributed Simulation. PADS’98,
may 1998.



Annexe & Bibliographie

ANNEXE

Le langage PARSEC

Les mots clés de PARSEC :

After	finalize	self
at	in	send
clocktype	message	stacksize
driver	new	timeout
ename	or	to
entity	receive	when

La définition d'une entité :

entity-def ::= **entity** *ident* ([*parameters*] [**stacksize** (*size_expr*)] *body*
parameters ::= une liste de paramètres en langage C
size_expr ::= une expression entière différente d'un tableau
body ::= {une série d'instruction en PARSEC et en C [*finalize_st*]}
finalize_st ::= **finalize** { une série d'instruction en C }

La création d'une entité :

new-st ::= [*ename_expr* =] **new** *ident*([*arg*]...) [**at** *node-no*]
ename_expr ::= une expression de type *ename*
arg ::= expression d'argument en C
node-no ::= une expression entière de valeur positive

La déclaration d'un message :

Message-def ::= **message** *ident* { *declarations* } [*ident*]...;
Declarations ::= [*type ident* [, *ident*]...];...
Type ::= **ename** | **clocktype** | **message** *ident* | une déclaration en C

Envoi de message :

send-st ::= **send** *msg_expr* **to** *ename_expr* [**after** *time_expr*];
ename_expr ::= une expression de type *ename*
msg_expr ::= *msg-type* { [*arg*]... } | *msg-ident*
time_expr ::= une expression entière positive
arg ::= *array-param* | expression en C
array-param ::= un *pointeur* [:une expression entière positive]
msg-type ::= un type de message définie pour l'entité *ename_expr*
msg-ident ::= une variable de type *msg-type*

Réception de message :

receive-st ::= *resume-clause* [**or** *timeout-resume*]
resume-clause ::= **receive** (*msg-list*) [**when** (*guard*)] *statement*
timeout-resume ::= *receive-st* | *timeout-st*
timeout-st ::= **timeout** **after** | **in** (*delay-time*) *statement*
msg-list ::= *msg-type* *mvar* [, *msg-list*]
delay-time ::= une expression en C de type **clocktype**
guard ::= expression *sans side-effects*
mvar ::= une variable de type *msg-type*
statement ::= une instruction en C ou en PARSEC

Les fonctions de bibliothèque du langage Parsec :

Les opérations sur l'horloge de simulation :

- **Simclock** (void) : retourne la valeur de courante de l'horloge de simulation

ANNEXE

- **Setmaxclock (clocktype)** : cette fonction précise le temps de simulation maximale au valeur spécifié dans le paramètre de clocktype.
- **atoa(char*, clocktype*)** : place la valeur de clocktype représenté par une chaîne de caractère dans le paramètre clocktype.
- **ctoa(clocktype, char*)** : elle mis la valeur de clocktype dans un format de chaîne de caractère

Les fonctions de simulation conservative :

- **add_dest(ename)**: ajoute l'entité spécifiée à l'ensemble de destination de l'entité courante
- **add_source(ename)**: ajoute l'entité spécifiée à l'ensemble de source de l'entité courante.
- **del_dest(ename)**: supprime l'entité spécifiée de l'ensemble de destination de l'entité courante .
- **del_source(ename)**: supprime l'entité spécifiée de l'ensemble de source de l'entité courante.
- **setlookahead(clocktype, clocktype)**: précise la valeur de lookahead pour l'entité courante.
- **setdestlookahead(clocktype, clocktype, ename)** :précise la valeur de lookahead pour une destination spécifié.

Les générateurs de nombre aléatoire :

- **double pc_erand(unsigned short[3])**: retourne une valeur entre $[0.0, 1.0]$
- **long pc_jrand(unsigned short[3])**: retourne une valeur entre $[-2^{31}, 2^{31}]$
- **long pc_nrand(unsigned short[3])**: retourne une valeur entre $[0, 2^{31}]$

Compilation du programme Parsec :

Syntaxe : **pcc** [options] noms de fichiers

Le compilateur du langage Parsec (pcc) accepte toutes les options de compilation du langage C, Ainsi les options de compilation suivantes :

- protocol Spécifie un des algorithmes de synchronisation:
 - mpc Message-passing C: ignore les messages timestamps
 - cons Conservative
 - opt Optimistic
- c Génère des fichiers ".o" et ".pi".
- E Génère des fichiers ".c" et ".pi" files.
- P Inhibe les numéros de ligne de translation. (Normalement, le compilateur PARSEC insère les numéros de ligne dans le fichier C intermédiaire, de sorte que le compilateur affiche les numéros de ligne des erreurs correctement).
- env affiche les variables d'environnement de parsec.
 - PCC_DIRECTORY : répertoire d'installation de PARSEC.
 - PCC_CC compilateur C utilisé par PARSEC.
 - PCC_LINKER : linker utilisé par PARSEC.
 - PCC_CC_OPTIONS : options pour passer au compilateur C.
 - PCC_PP_OPTIONS : options par défaut du compilateur PARSEC.
 - PCC_LINKER_OPTIONS : options par défaut du linker C.
- pcc_directory : utilise PCC_DIRECTORY spécifié.
- pcc_cc : utilise un compilateur C spécifie.
- pcc_linker : utilise un linker spécifie.
- pcc_pp_options : utilise ces options pour la précompilation.
- pcc_cc_options : utilise ces options pour la compilation.
- pcc_linker_options : utilise ces options pour l'édition de lien.
- ini Sauvegarde les fichiers d'initialisation auto-generés.
- ff active la compilation des fonctions amies.
- stack Change la taille par défaut de la pile pour les entités.
- help affiche les options de compilation.

ANNEXE

-V affiche le numéro de version du compilateur.
-user_main renomme la fonction main par parsec_main.
-shared_lib renomme la fonction main par parsec_main et crée une bibliothèque partagée
-clock précise la représentation par défaut pour clocktype. Les options valides sont:
unsigned unsigned long, un type entier de 32 bit (par défaut)
longlong long long, un type entier de 64 bit

Les exemples suivants montrent comment PARSEC compile des programmes dans une architecture séquentielle.

% pcc -o example example.pc

Cela génère un fichier exécutable *example* dans le répertoire de travail courant.

% pcc -o example example.pc xxx.c yyy.o

Cela génère un fichier exécutable nommé *example* dans le répertoire de travail courant. Le fichier *example* est compilé et lié avec xxx.c et yyy.o.