Centre d'Innovations en Télécommunications et Intégration de services

Mémoire de Master Recherche Informatique
Spécialité Réseaux, Télécommunications et Services
2004/2005

Proposition et validation formelle d'un protocole MAC temps réel pour réseaux de capteurs linéaires sans fils

Thomas Watteyne
thomas.watteyne@insa-lyon.fr

Laboratoire CITI

Tutrice : Isabelle Augé-Blum,
Maître de Conférences à l'INSA de Lyon

Résumé

De nombreuses applications de réseaux de capteurs sans fils voient actuellement le jour, dans des domaines aussi variés que la Défense, la sécurité, la santé ou les maisons intelligentes. Leur but est souvent de surveiller une zone géographique et de remonter une alarme en cas de détection d'un évènement redouté. Lorsqu'un tel événement se produit, l'application doit réagir au plus tard après un temps connu et borné ; il s'agit de contraintes dites temps-réel dur. Or dans la littérature peu de protocoles de communication pour les réseaux de capteurs permettent de respecter ce type de contraintes, et à notre connaissance, le seul existant fait des suppositions très contraignantes sur le réseau de capteurs. Dans ce travail, nous avons choisi de prendre comme exemple une application de surveillance d'accidents sur autoroute, le réseau étant linéaire, le routage devient inutile. Ce travail propose donc un nouveau protocole MAC temps-réel en faisant des hypothèses réalistes sur les réseaux de capteurs. De plus, nous présentons une validation formelle de ce protocole, et explicitons les bornes temporelles dans le cas pire pour les différents services offerts par le protocole (initialisation, remontée d'alarmes dans les différents modes). Ces travaux ont donné lieu à la soumission d'un article de recherche au « 13th Annual Meeting of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'05) », qui aura lieu à Atlanta aux Etats-Unis, du 26 au 29 septembre 2005.

Mots clés :
Réseaux de capteurs sans fil, protocole MAC, temps-réel dur, modélisation et validation formelle.

Abstract

Several wireless sensor network applications are currently popping up, in areas ranging from Defense, security, to health or smart homes. Their goal is often to monitor a geographic area and when a sensor detects a feared event, it informs the sink using an alarm message. When such an event happens, the application needs to react with a finite bounded and known delay; these are real-time constraints. What's more, only few works on communication protocols deal with such constraints, and to our knowledge, the only real-time proposal makes very restrictive assumptions on the sensor network. In this work, we chose to use a highway car accident system as an application example, the network is then linear and routing becomes unnecessary. This work proposes a new real-time MAC protocol with realistic assumptions on sensor networks. What's more, we present a formal validation of this protocol, and explicit the worst case times for the services offered by the protocol (initialization, alarm transmission using the different modes). This work lead to a paper submission for the 13th Annual Meeting of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'05), which will be held in Atlanta, U.S, September 26-29 2005.

Key Words:
Wireless sensor networks, MAC protocol, hard real-time, formal modeling and validation.

Remerciements

Je tiens tout particulièrement à remercier le département « Télécommunication, Services et Usages » de l'INSA de Lyon pour m'avoir donné l'opportunité d'effectuer un Master au cours de ma cinquième année d'études, et de m'avoir soutenu tout au long de celui-ci.

Je souhaite aussi vivement remercier l'ensemble de l'équipe du laboratoire CITI de l'INSA de Lyon, et son directeur Stéphane Ubéda, pour m'avoir ouvert ses portes et m'avoir permis de participer à la 9ème école d'été Internet Nouvelle Génération (ING'2005), école d'été de la communauté française de recherche en réseaux, qui s'est tenue du 13 au 17 juin 2005 à Montreuil-sur-Mer.

Je tiens à remercier Jean-Marie Gorce et Stéphane Ubéda, ainsi que toute l'équipe des systèmes embarqués communicants pour leur apport scientifique, leurs commentaires et tout le temps qu'ils m'ont offert.

Mais surtout, du fond du cœur, je tiens à remercier Isabelle Augé-Blum, ma tutrice de stage, pour son engagement, sa patience, ses conseils, et pour tous les bons moments qu'on a passés ensemble.

Sommaire

Chapitre I

Introduction

On constate actuellement un essor de la recherche dans le domaine des réseaux ad-hoc (réseaux sans infrastructures, dont les nœuds communiquent par lien radio, et qui doivent s'auto-organiser, chaque nœud pouvant servir de routeur à ses voisins, et donc retransmettre un message reçu), et en particulier dans les réseaux de capteurs. Alors que peu de personnes peuvent aujourd'hui prédire quelles seront dans le futur les applications des réseaux ad-hoc, de nombreuses applications sont prévisibles pour les réseaux de capteurs, que ce soit dans le domaine militaire (surveillance des troupes déployées) ou civil (détection de feux de forêts). Les réseaux de capteurs peuvent ainsi être vus comme un domaine d'application des réseaux ad-hoc.

On a donc aujourd'hui une bonne vision des contraintes que devront respecter les systèmes de communication pour pouvoir implanter les différentes classes d'applications. Avec de nombreux domaines d'application liés à la sécurité (comme la surveillance), le respect de contraintes temps-réel par ces applications jouera un rôle de plus en plus important. En effet, dans une application de surveillance, lorsqu'un évènement se produit et nécessite une réaction, celle-ci doit intervenir au plus tard après un temps connu et borné. Ce temps de réaction à respecter est appelé contrainte temps-réel et dépend de la dynamique de l'environnement.

Pour respecter les contraintes temps-réel de l'application implantée, il faut non seulement que l'architecture matérielle et logicielle du capteur offre de telles garanties, mais également que le réseau de communication sous-jacent soit déterministe et qu'il permette des temps de transmission connus et bornés. Nous nous concentrons dans ce travail sur la partie réseau du problème (et non sur la conception du capteur).

Pour assurer le bon fonctionnement d'applications critiques, il est nécessaire de valider formellement ces garanties. S'il exite quelques propositions de protocoles de communication permettant de respecter des contraintes temporelles, ces protocoles nécessitent des capteurs ayant des caractéristiques peu réalistes. De plus, aucune validation n'est à notre connaissance proposée dans la littérature.

En prenant comme exemple une classe d'application des réseaux de capteurs qui ne nécessite pas d'algorithme de routage (comme le réseau est linéaire, le chemin que doit emprunter un message est connu) on peut travailler directement au niveau MAC. Le but de ce travail est donc de répondre à ce manque en proposant un protocole MAC temps-réel pour réseaux de capteurs, et en effectuant sa validation formelle, tant au niveau comportemental (*est-ce que notre protocole a bien le comportement voulu dans tous les cas possibles ?*) que temporel (*est-ce que notre protocole respecte bien les contraintes de temps voulues ?*).

La suite de ce document est organisée comme suit. Dans le chapitre II, nous présentons le contexte de ce travail et nous faisons un état de l'art sur les contraintes des réseaux de capteurs, et en particulier sur les communications temps-réel dans ces réseaux. Nous présentons le protocole proposé en détail dans le chapitre III, et sa validation formelle comportementale et temporelle dans le chapitre IV. Nous concluons ce travail et donnons des perspectives possibles dans le Chapitre V.

Chapitre II

Etat de l'art

1. Les réseaux de capteurs

Grâce aux avancées conjointes des systèmes microélectroniques et mécaniques (ou *Micro-Electro Mechanical Systems*, MEMS), des technologies sans fils, et de la microélectronique embarquée, les réseaux de capteurs sans fil ont récemment pu voir le jour [5].

1.1. Définition

Un capteur est un composant physique, capable d'accomplir trois tâches complémentaires: le relevé d'une grandeur physique, le traitement éventuel de cette information, et la communication avec d'autres capteurs. L'ensemble de ces capteurs déployés pour une application forme un réseau de capteurs. Le but de celui-ci est de surveiller une zone géographique, et parfois d'agir sur celle-ci (il s'agit alors de réseaux de capteurs-actionneurs). On peut citer comme exemples un réseau détecteur de feu de forêt, ou un réseau de surveillance de solidité d'un pont après un tremblement de terre. Le réseau peut comporter un grand nombre de nœuds (plusieurs milliers).

Les capteurs sont placés de manière plus ou moins aléatoire (par exemple par largage depuis un hélicoptère) dans des environnements pouvant être dangereux. Toute intervention humaine après déploiement sur les capteurs est la plupart du temps exclue, le réseau doit donc s'autogérer.

Afin de travailler de manière coopérative, les informations recueillies sont partagées entre les capteurs par voie hertzienne. Le choix du lien radio plutôt que du lien filaire permet un déploiement facile et rapide dans un environnement pouvant être inaccessible pour l'être humain.

Les réseaux de capteurs sans fils forment un domaine d'application des réseaux ad-hoc. Ces derniers sont un type de réseau sans-fil, sans infrastructure fixe et avec une topologie changeante (à cause par exemple de la disparition de nœuds suite à l'épuisement de leur batterie). La zone à couvrir peut être très étendue et l'interface radio embarquée dans chaque nœud n'est généralement pas assez puissante pour établir une communication directe avec tous les nœuds. Une communication de type ad-hoc multi-sauts est donc nécessaire : chaque nœud sert de relais à une communication entre deux nœuds trop éloignés l'un de l'autre. Des protocoles sont nécessaires afin d'assurer l'auto-organisation du réseau, mais également pour assurer une bonne communication (accès au médium, routage).

1.2. Deux types de réseaux

Il existe deux grands types de réseaux de capteurs sans fil :

- soit le réseau est constitué d'un ensemble de capteurs mobiles évoluant dans un environnement statique. Le but de tels réseaux est la plupart du temps l'exploration de zones inaccessibles ou dangereuses. Les travaux de recherche sont souvent orientés robotique [43, 50], les nœuds jouant à la fois le rôle de capteur et d'actionneur.
- soit le réseau est constitué de capteurs fixes servant à la surveillance d'occurrence d'évènements sur une zone géographique [40, 52, 55]. Ici, le réseau n'effectue que la surveillance, les données mesurées sont transmises en mode multi-sauts à un nœud

spécifique appelé « puits » qui est chargé, après réception, de mettre en œuvre les actions nécessaires. Ce puits peut être connecté, de manière filaire par exemple, à un autre réseau [46, 38, 15].

La disparité entre les nœuds peut être très grande d'un point de vue performances. Les capteurs mobiles [43, 50] sont en règles générale plus performants que les capteurs statiques, et les travaux de recherche posent ainsi des hypothèses plus importantes, comme la présence de systèmes dynamiques de localisation tel que le GPS. Au vu du grand nombre de nœuds que peut contenir un réseau statique, les capteurs de ces réseaux sont généralement plus limités [15, 46].

1.3. Domaines d'application

L'intérêt des réseaux de capteurs est réellement vu à travers l'éventail très large des domaines d'application. [5] classifie les applications en cinq familles : les applications militaires (surveillance de la position des soldats), de surveillance environnementales (détection de feux de forêt, surveillance d'une centrale nucléaire [33]), de santé (surveillance des médicaments administrés dans un hôpital), de type « maison intelligente » (réglage de l'éclairage en fonction de la position des habitants), et les autres applications commerciales (musées interactifs). Dû à la miniaturisation des capteurs, combinée avec la convergence des moyens de communication (téléphonie mobile, Internet haut débit ...) et l'apparition de réseaux ubiquitaires, les réseaux de capteurs auront un impact majeur sur la vie courante dans un avenir proche selon [15].

1.4. Contraintes

Les protocoles des réseaux ad-hoc sont fort complexes [18] ; ils doivent par exemple gérer l'auto-organisation du réseau, c'est-à-dire convertir un ensemble de nœuds qui ne savent rien les uns des autres en un réseau fonctionnel de nœuds coopératifs. Les contraintes auxquelles doivent répondre les réseaux ad-hoc (« dynamicité », absence d'infrastructure, auto-organisation) se retrouvent donc logiquement dans les réseaux de capteurs.

Des contraintes supplémentaires, héritées de l'aspect « système embarqué » propre aux capteurs, font également leur apparition. Deux séries de contraintes s'opposent : celles du capteur et celles liées à l'application de surveillance. En effet, comme un réseau peut être formé d'un grand nombre de capteurs (des milliers), il est nécessaire d'avoir un coût unitaire réduit, ce qui implique une baisse de performances (en puissance de calcul, mémoire disponible, énergie embarquée et bande passante, essentiellement). D'un autre côté, les applications posent des contraintes fortes, telles qu'une durée de vie minimale du système (aucune intervention humaine n'est possible, donc la batterie des capteurs ne peut être changée), une forte tolérance aux fautes (la disparition d'un nœud par exemple), et une possibilité de passage à l'échelle (les protocoles de communication doivent rester utilisables, même avec plusieurs milliers de nœuds).

2. Communication entre les noeuds

Méthodes d'accès au médium. Le rôle des protocoles MAC est de réguler l'accès au médium entre les nœuds qui désirent parler. Cette régulation peut se faire de manière centralisée ou distribuée. La communication entre les nœuds peut se faire selon deux logiques : l'élection d'un nœud leader à un niveau local ou pour l'ensemble du réseau [12, 34] ou l'utilisation de solutions entièrement distribuées [19]. Les inconvénients de la première approche sont que la présence d'un leader diminue la robustesse (la perte de ce nœud rend inutilisables les nœuds qui lui sont rattachés) et la durée de vie d'un système (le leader pourra épuiser son énergie plus rapidement). Les solutions distribuées seront donc privilégiées pour les réseaux de capteurs.

Comme illustré par la figure 1, l'accès au médium en soit peut être fait de deux manières : un accès garanti (avec par exemple l'assignation d'un intervalle de temps distinct à chaque nœud) ou un accès aléatoire (chaque nœud émet quand il veut et réémet si le message n'a pas été reçu correctement). L'accès garanti est plus facilement implémenté à partir d'un protocole centralisé [13].

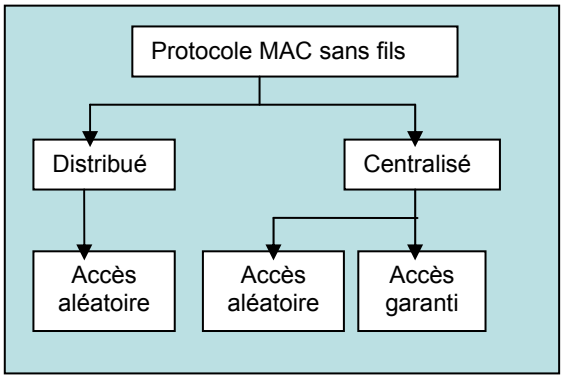


Figure 1. Une classification des protocoles MAC sans fils existants

[13] rajoute une autre famille de protocoles MAC (qui peuvent être distribués ou centralisés) : la non-collision, en utilisant plusieurs canaux non-interférents [37]. Alors que les méthodes basées sur l'accès garanti ou la non-collision supposent un coût matériel plus coûteux (interfaces radio à plusieurs fréquences, méthodes de synchronisation par GPS, …), ils offrent le déterminisme nécessaire à la garantie de contraintes temps-réel.

Routage. Le routage dans un réseau de capteurs [28] doit satisfaire à la fois aux contraintes des réseaux ad-hoc (fiabilité des liens radio, changements de topologie après perte de noeuds) et les contraintes propres des réseaux de capteurs, comme la consommation d'énergie [20]. Comme l'identité propre d'un nœud n'a que peu d'importance par rapport aux alarmes qu'il génère, les nœuds sont le plus souvent identifiés par leur position, ce qui induit l'utilisation de protocoles de routage géographiques [35], tel que GPSR [29, 14].

Synchronisation. Avec l'idée de coordonner les actions des nœuds apparaît le problème de synchronisation. Même si des méthodes de synchronisation sur horloge globale comme le GPS sont possibles, elles sont coûteuses, et une synchronisation des horloges internes est parfois préférable. Ces horloges ne sont toutefois pas parfaites et les nœuds subissent forcément une dérive [45]. La re-synchronisation simple par message de synchronisation est peu précise car il ne prend pas en compte les temps de propagation. Une idée est de diffuser un tel message de synchronisation puis de comparer les instants de réception entre récepteurs [45]. D'autre part, d'autres nœuds peuvent écouter les échanges de tels messages entre deux autres nœuds pour se synchroniser sans avoir à émettre de messages de synchronisation propres [54].

Energie. Une intervention humaine n'étant pas possible pour changer une batterie, la durée de vie du réseau dépend de l'utilisation de l'énergie disponible. Or la consommation est faite essentiellement par l'interface radio (qui est presque aussi gourmande en réception qu'en émission). L'optimisation de l'utilisation de l'énergie peut donc être faite à tous les niveaux. Une première idée est d'endormir un certain nombre de nœuds (leur interface radio étant alors éteinte), cet endormissement pouvant être fait de manière complètement aléatoire [16] ou coordonnée, du moins localement [53]. Les algorithmes MAC jouent aussi leur rôle, puisqu'en évitant les collisions (par exemple en allouant un intervalle de temps à chaque nœud), donc les réémissions des messages, on réduit la consommation d'énergie [25]. Le routage se base sur le choix du prochain nœud relais d'un message, il est intéressant de le choisir de manière aléatoire et équiprobable parmi un ensemble de nœuds potentiels afin de ne pas surcharger un nœud en particulier, donc de réduire sa durée de vie [36]. D'un point de

vue applicatif, les nœuds peuvent être vus comme un ensemble de ressources en temps de calcul et bande passante, et il faut répartir équitablement la charge totale de l'application sur les ressources. Une répartition homogène allonge la durée de vie du réseau [21].

Cross-layering. L'exemple de l'énergie est parlant, puisqu'on se rend compte que la prise en compte de certaines contraintes propres aux réseaux de capteurs ne se fait pas uniquement sur une couche, mais est distribuée sur toutes. On parle de caractéristiques trans-couches, ou de *cross-layering.* Le modèle ISO-OSI [49] utilisé pour décrire les réseaux et composé de 7 couches superposées coopérantes ne suffit donc plus. Une proposition (illustrée par la figure 2) est de généraliser le modèle plan en un modèle en trois dimensions, en rajoutant outre le plan de communication, un plan de management et un plan de coordination [4].

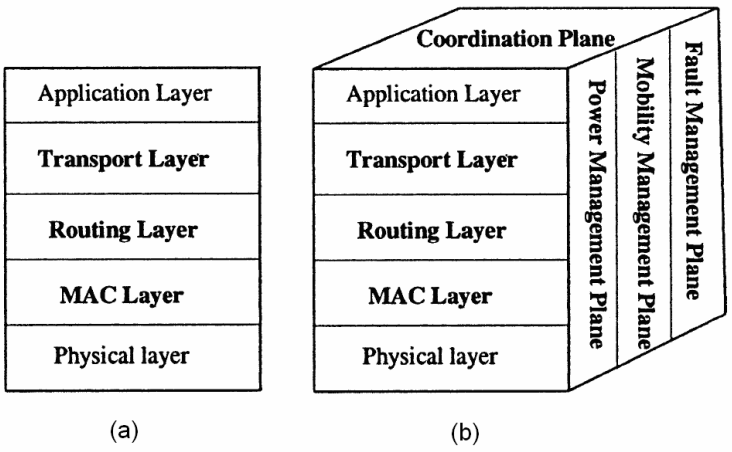


Figure 2. Le modèle ISO-OSI original (a) et spécifié pour réseaux de capteurs (b)

Le réseau ne peut donc plus être vu comme un ensemble hétérogène de nœuds indépendants voulant communiquer ; il est fortement coopératif, orienté application et données. Comme ce qui importe sont les données récoltées par les capteurs, les détails du réseau peuvent être rendus abstraits à l'application, celle-ci étant alors distribuée sur un ensemble de nœuds coopératifs dépendants. Cette coopération s'exprime par exemple par des mécanismes d'agrégation de données, lorsque les informations relevées par différents capteurs sont regroupées voire prétraitées dans un autre capteur, au cours de leur transmission en mode multi-sauts vers le puits.

3. Temps-réel dans les réseaux de capteurs

De par la nature même des applications (e.g. réseau de détection de feux de forêt avec remontée d'alarmes), le temps entre l'apparition d'une alarme et la réaction de l'application à celle-ci doit être connu et borné [48]. En effet, un message arrivant trop tard est considéré comme un message perdu, ou dont l'utilité est amoindrie (dans le cas d'un feu de forêt, une alarme arrivant lorsque tout a déjà brûlé ne présente pas beaucoup d'intérêt). On parle de contraintes temps-réel.

Bien sûr, pour qu'une application respecte des contraintes temps-réel, les capteurs doivent être capables de les respecter eux aussi, que ce soit au niveau de l'acquisition, du traitement ou de la communication. Nous nous focalisons sur la partie communication. La prise en compte de contraintes temps-réel par le réseau apparaît comme une priorité, et est d'ailleurs citée comme telle dans [47], présentant les défis de la Recherche concernant les réseaux de capteurs. Cette prise en compte est un exemple type de *cross-layering*, puisqu'elle doit être faite à tous les niveaux.

Il existe deux types de contraintes temps-réel : le temps-réel relâché (*soft real-time*) et le temps-réel dur (*hard real-time*). Dans une même application et un même réseau, plusieurs flux peuvent cohabiter, chacun ayant des contraintes temporelles plus ou moins fortes. Un flux temps-réel dur doit toujours arriver avant son *deadline*, c'est-à-dire avant le dernier instant avant la perte de validité de l'information contenue. Pour un flux temps-réel relâché, un pourcentage de messages arrivés après leur deadline est acceptable (on appelle *miss ratio* ce pourcentage). De même, un système temps-réel dur garantit un temps d'exécution pire ou de transmission pire (*Worst Case Execution Time* WCET, et *Worst Case Transmission Time* WCTT).

Il est intéressant de noter que tout réseau a une capacité limitée de trafic temps-réel qu'il peut acheminer avec un *miss ratio* nul (temps-réel dur). Une fois ce seuil dépassé, le *miss ratio* n'est plus nul, et on tombe sur un cas de temps-réel relâché. Ce seuil peut être quantifié et exprimé en fonction du nombre de nœuds et de leur densité, de la longueur du plus long chemin, et du taux d'apparition de messages [1].

3.1. Réseaux de capteurs et temps-réel relâché : état de l'art

Dans les réseaux supportant des contraintes temps-réel relâché, deux types de flux cohabitent : les flux traditionnels (sans contraintes temporelles) et les flux dits temps-réel. L'idée est de favoriser la transmission de ces derniers, afin d'augmenter leur probabilité d'arriver à destination à temps (avant leur *deadline*). Le *miss ratio* sert de critère pour comparer les différentes solutions proposées.

Couche MAC. Pour favoriser les flux dits temps-réel, il faut différentier les flux, et ce dès la couche MAC. Un certain nombre de protocoles MAC distribués non temps-réel existent déjà, et comme aucune réelle garantie n'est à donner, le plus simple est de partir de l'un d'eux, et de le modifier pour y intégrer des priorités. En partant par exemple de 802.11 DCF [26], et en réduisant les temps d'attente d'accès au medium (*backoff* et *DIFS*) en fonction de la priorité du flux, on arrive à la différentiation voulue [41]. Toujours avec 802.11, on peut réguler l'accès au médium en demandant à tous les nœuds voulant y accéder d'émettre un signal d'une longueur proportionnelle à leur priorité. Si après cette émission, un nœud détecte que le médium est toujours occupé, il se tait ; ainsi seul le nœud de plus haute priorité pourra émettre [41].

Outre les propositions distribuées, l'accès au médium peut également être régulé par un ordonnanceur central. Celui-ci doit nécessairement connaître la position de tous les nœuds ainsi que les caractéristiques de tous les messages, et prendre en compte le nombre de sauts qu'un message doit effectuer pour atteindre la destination. Ce type de solution est pour cela particulièrement adaptée aux équipes de robots coopératifs [31, 32].

RAP. Même avec une couche MAC supportant des flux avec des priorités différentes, il faut savoir comment attribuer une priorité à un flux. La proximité temporelle du *deadline* est une solution intuitive, mais celle-ci ne prend pas en compte la distance géographique à parcourir. [35] présente une architecture de communication appelée RAP qui définit pour un message une vitesse requise, qui sera ensuite traduite en une priorité de couche MAC. En respectant cette vitesse au cours de la transmission multi-sauts, le message arrivera à temps à destination. Comme l'évaluation de la distance physique que parcourra le message dépend du protocole de routage choisi, le concept de vitesse requise doit être implémenté dans une architecture regroupant les couches 1, 2 et 3 du modèle ISO-OSI [49]. Afin de réaffecter le temps accumulé par un message ayant jusque là voyagé à une vitesse supérieure à la vitesse requise, celle-ci peut être recalculée à chaque saut.

SPEED. Un protocole de routage temps-réel est intéressant car il prend en compte l'urgence des messages pour prendre ses décisions. Ainsi, il suffit de rajouter dans les traditionnelles tables de routages échangées entre nœuds routeurs un champ rendant compte du temps nécessaire pour aller à tel ou tel nœud (renseigné à partir de messages test). Ainsi, un message ne sera aiguillé sur cette interface que si elle permet d'arriver à destination à temps. En cas de congestion d'un chemin, donc de l'augmentation du temps de parcours de celui-ci, des paquets de rétro-information doivent avertir les nœuds en aval de choisir un autre chemin [24]. Bien entendu, ce protocole de routage peut être intégré dans l'architecture RAP [2]. Aucune garantie n'est néanmoins donnée puisque la congestion ne peut être prédite, et la perte de paquets par dépassement de *deadline* est toujours possible.

Remarque : Plus anecdotique, [3] propose de positionner de puits de manière optimale, c'est-à-dire près des zones congestionnées du réseau, afin de diminuer le *miss ratio* d'un système quelconque.

3.2. Réseaux de capteurs et temps-réel dur : état de l'art

Même si l'approche temps-réel relâché propose des solutions visant à réduire le *miss ratio*, pour des systèmes critiques ou ayant des contraintes de qualité de service fortes, il est indispensable que celui-ci soit nul, et que les temps soient bornés et garantis. La conception de systèmes temps-réel dur est quasiment impossible à faire à partir de systèmes non temps-réel déjà existants. Afin qu'une solution de communication soit temps-réel dur, cette caractéristique doit nécessairement être présente à toutes les couches.

I-EDF. A notre connaissance, seul [12] présente une solution réellement temps-réel : *Implicit Earliest Deadline First* (I-EDF). Il s'agit d'un protocole MAC, basé sur un ordonnancement de type EDF (*Earliest Deadline First*, [12]). Cet ordonnanceur attribue le médium au flux avec le *deadline* le plus proche temporellement de l'instant présent. Comme illustré par la figure 3, on suppose le réseau découpé en cellules de nœuds hexagonales et de même taille. Chaque cellule a donc 6 voisines, et au centre de chacune se trouve un nœud routeur qui a deux interfaces radio. On suppose que les rayons de communication et d'interférence sont égaux à la distance entre deux nœuds routeurs de cellules voisines. On différencie la communication intra- en intercellulaire. On attribue un canal de fréquence différent de ses voisines à chaque cellule ; 7 canaux suffisent pour éviter toute interférence entre voisines. A l'intérieur d'une cellule, chaque nœud sait quels sont ses voisins et quelles sont les caractéristiques des messages que chacun compte envoyer (période, *deadline*, durée). L'algorithme EDF [12] est lancé à chaque nœud, ce qui permet à chaque nœud d'obtenir la même table d'ordonnancement. En utilisant celle-ci, aucune collision n'est donc possible durant la communication intracellulaire. Pour ce qui est de la communication intercellulaire, les 6 directions des hexagones sont notées A,B,C, … et les intervalles de temps réservés aux communications intracellulaires sont entrecoupés d'intervalles de communication intercellulaire. Le routeur d'une cellule émet avec la fréquence de la cellule destination, ce qui rend une communication intercellulaire directionnelle possible. Ainsi, dans un même intervalle réservé à la communication intercellulaire, tous les nœuds vont émettre dans la même direction (i.e. C), cette direction étant changée après chaque émission. L'utilisation à la fois d'ordonnancement temporel (une découpe en intervalles de temps, combinée avec l'algorithme d'ordonnancement EDF) et de multiples fréquences garantit un fonctionnement sans collisions. Cette solution est bien temps-réel dur, une validation par théorèmes est présentée [12]. [11] présente une évaluation quantitative de la capacité atteignable en trafic temps-réel en utilisant ce protocole. [17], [19] et [34] présentent une application de I-EDF sur une équipe de robots coopératifs.

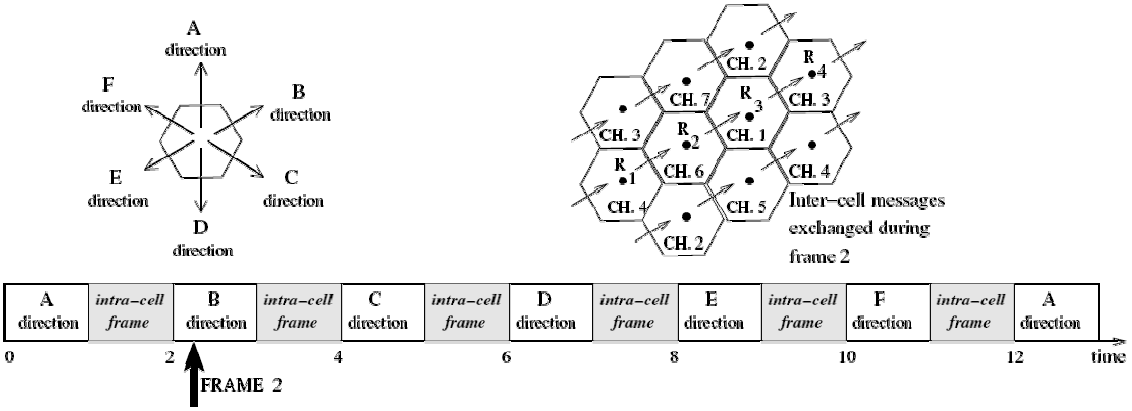


Figure 3. Communication intercellulaire avec I-EDF

Même s'il s'agit d'une proposition temps-réel dur, les suppositions faites sont difficiles à satisfaire. Premièrement, l'organisation en hexagones de même taille est très rigide, et semble peu compatible avec les milieux inhospitaliers de certaines applications (surveillance sismique, feux de forêts, …) et un déploiement aléatoire. De plus, l'égalité entre les rayons de communication et d'interférence est physiquement impossible à réaliser. De par l'importance des ressources embarquées dans les nœuds (multiples fréquences, double interface radio pour les nœuds routeurs), le prix unitaire du nœud ne semble pas assez bas pour permettre un déploiement à grande échelle. Enfin, la parfaite synchronisation des nœuds suppose un système de synchronisation sur une horloge globale tel que le GPS (*Global Positioning System*), augmentant encore le prix du nœud.

4. Validation formelle

Avant tout déploiement, le protocole de communication doit être validé pour vérifier, d'une part, qu'il a bien le comportement voulu par les spécifications, et d'autre part qu'il respecte bien les contraintes temps-réel imposées par l'application.

4.1. Validation comportementale et temporelle

Pour valider un protocole, on le modélise à l'aide d'un formalisme, choisi en fonction de trois critères majeurs : le pouvoir d'expression (capacité du modèle à exprimer les caractéristiques du système à étudier), le pouvoir de modélisation (pouvoir les exprimer simplement, critère assez subjectif, qui dépend beaucoup de l'expertise de la personne qui modélise) et le pouvoir d'analyse ou de vérification (capacité à exprimer et à vérifier des propriétés sur le système).

Une fois le modèle formel construit, une exploration exhaustive de tous les chemins peut être menée par *model-checking*. La méthodologie que nous allons mettre en œuvre est inspirée de la méthodologie présentée dans [22] pour la validation temporelle de réseaux embarqués critiques pour l'automobile. Dans un premier temps, les contraintes que devra satisfaire le système doivent être définies formellement et seront utilisées comme propriétés au cours du *model-checking*. Il s'agit à la fois de contraintes de comportement et de contraintes temporelles. Parallèlement, un jeu de scénarios est créé, sur lequel les propriétés seront vérifiées. Un moteur de *model-checking* effectue la validation formelle de l'ensemble des propriétés sur le modèle, en s'appuyant sur le jeu de scénarios. La méthodologie est synthétisée dans la figure 4.

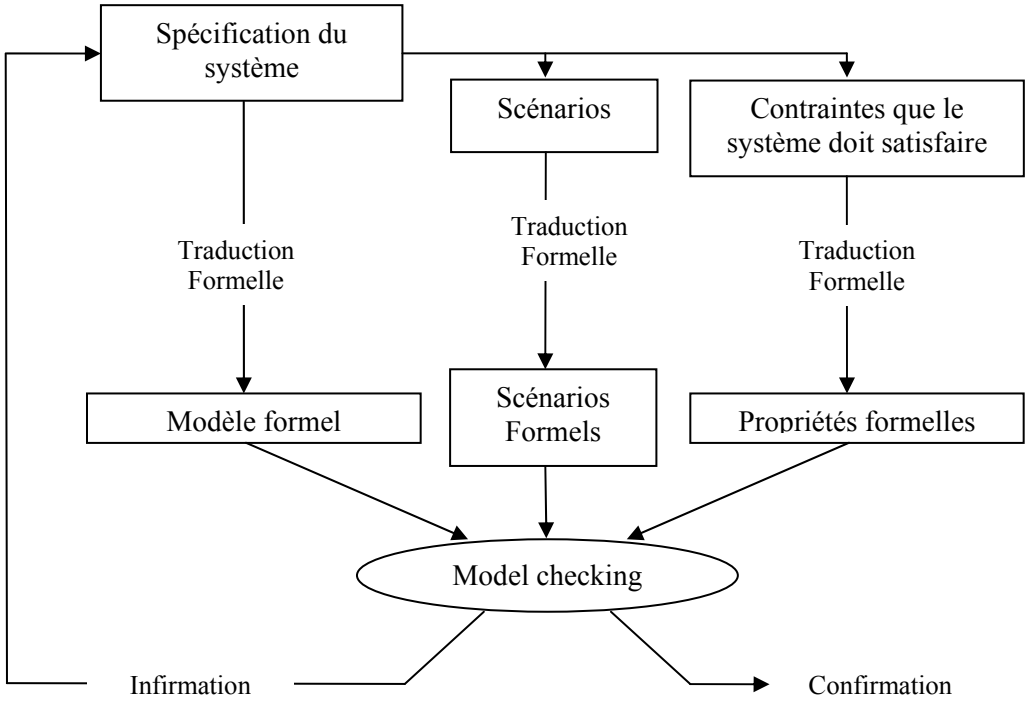


Figure 4. Méthodologie adoptée

Justification du choix d'UPPAAL. Pour ce qui est du pouvoir d'expression, nos besoins sont ceux de tout protocole de communication. Il doit être possible d'exprimer le parallélisme et la concurrence (plusieurs noeuds fonctionnent en parallèle), une communication entre ceux-ci doit être possible avec un mécanisme d'envoi de messages. Il doit de plus être possible d'exprimer des choix, ainsi que des contraintes temporelles. Pour ce qui est de l'analyse, nous voulons pouvoir vérifier un comportement et obtenir des bornes temporelles.

Nos besoins sont très proches de ceux de [22, 23] dont le but est de valider temporellement une architecture embarquée temps-réel dans l'automobile. Ces travaux ont comparés quatre formalismes basés sur les systèmes à états-transitions (TSA [8], IF [10], les réseaux de Petri temporels [39] et SDL [27]), en particulier en termes de pouvoir d'expression de contraintes temporelles et de la complexité de l'analyse. Ces travaux concluent que pour nos besoins, TSA (et l'outil UPPAAL associé) est le formalisme le plus adapté.

Présentation d'UPPAAL. UPPAAL [51, 30] est un environnement complet de validation formelle. Le formalisme de modélisation TSA (*Timed Safety Automata*) utilisé est une extension des automates temporisés [7] où un système est représenté sous la forme d'un ensemble concurrent de systèmes à états-transitions. Des variables globales/locales peuvent être définies ; les variables numériques et les horloges sont déclarées de la même manière. La synchronisation entre les différents automates se fait par signaux. Un exemple de système composé d'une horloge envoyant de manière périodique un signal à un utilisateur est représenté dans la figure 5. Les deux automates *Horloge* et *Utilisateur* communiquent par signaux.

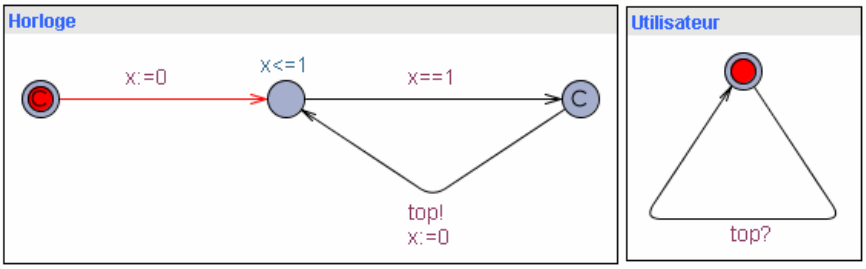


Figure 5. Modélisation d'une horloge en TSA

Un état peut avoir un *invariant*, une propriété qui doit être vérifiée lorsque le système se trouve dans cet état (exemple *x<=1* dans l'état central du système *Horloge*). Dans un état *committed* (marqué *C*) le temps ne peut pas s'écouler, il doit donc être quitté immédiatement. Une transition est franchie instantanément et on peut lui affecter un *guard*, une propriété à vérifier avant de pouvoir la franchir. Le franchissement d'une transition peut également engendrer la mise à jour d'une variable (l'horloge *x:=0*), et/ou l'envoi d'un signal (*top!*). Un autre automate doit pouvoir recevoir ce signal (*top?*) sous peine de bloquer le passage de cette transition.

Les propriétés à vérifier sont exprimées dans le langage de logique temporelle TCTL (*Timed Computation Tree Logic*, [6]). La vérification se fait en parcourant de manière exhaustive tous les chemins possibles lors de l'exécution du graphe ; les propriétés sont donc exprimées sur ces chemins d'exécution. Les formules logiques suivantes sont utilisées :

E	Il existe un chemin tel que …
A	Pour tous les chemins …
[]	… dans tous les états de ce chemin.
<>	… dans au moins un état de ce chemin.

Ainsi, en se reportant à l'exemple de la figure 5, la propriété (vérifiée) « *A[ ] x<=1* » vérifie que l'horloge *x* est toujours plus petite ou égale à 1.

5. Conclusion

A notre connaissance, il n'existe donc pas d'architecture permettant de fournir des garanties temps-réel dur et de contraintes réalistes pour les réseaux de capteurs. Dans la suite de ce travail, nous allons donc essayer de proposer un protocole MAC temps-réel et de le valider, c'est-à-dire prouver formellement qu'il permet de respecter des contraintes temps-réel.

Chapitre III

Proposition

Le but de cette partie est de proposer un protocole temps-réel, avec des hypothèses les plus réalistes possibles sur les nœuds et le lien radio. Dans un premier temps cependant, nous ne considérons pas de fautes sur les transmissions, et nous considérons une classe d'application permettant de nous affranchir du routage.

1. Classe d'application considérée

Dans le but de nous affranchir du routage, nous considérons une classe d'application de réseau linéaire. Il existe de nombreuses applications de réseau linéaire comme la surveillance d'une chaîne de production ou le suivi de trains sur une voie ferrée. Nous avons choisi d'adopter une application tout au long de ce document : la détection d'accident sur autoroute. Un ensemble de capteurs capables de détecter un accident sont placés le long d'une autoroute. Tout accident génère une alarme qui est transmise et qui sera reçue par un puits (qui devra alors mettre en œuvre les actions nécessaires). Le temps séparant l'occurrence de l'alarme et sa réception par le puits doit être connu et borné.

Le choix de cette classe d'application nous permet bien de nous affranchir du routage, car dans ce type de réseau, les messages ne peuvent emprunter qu'un seul chemin. Ainsi une couche protocolaire de routage (la couche 3 du modèle ISO-OSI) devient inutile, la couche MAC est suffisante.

Remarque: Un certain nombre de systèmes de surveillance du trafic sur autoroute sont commercialisés. Alors que la majorité se base sur des capteurs à induction magnétique avec un réseau filaire sous-jacent, SmarTek Systems™ [42] propose un système alliant l'utilisation de capteurs acoustiques directifs et de technologie sans fil. Une borne sans fil positionnée au centre des capteurs joue le rôle de maître et récupère les enregistrements des différents capteurs par méthode de requête/réponse périodique de chaque capteur (*polling*). Outre le fait que cette méthode n'est pas optimale en terme de consommation d'énergie, la longueur de la zone couverte est limitée à deux fois la portée du système de communication, et le déploiement des différents éléments du système (capteurs, maître) telle que les capteurs puisse communiquer avec le maître et inversement est effectué de manière empirique. Comme les caractéristiques du lien radio peuvent changer avec la météo par exemple, aucune garantie n'est offert sur la réception par le maître des enregistrements.

2. Hypothèses

Sur les nœuds. Le prix unitaire d'un nœud doit être aussi bas que possible, d'une part parce qu'un réseau peut être constitué d'un nombre important de nœuds, et d'autre part parce que la destruction de nœuds (déploiement par largage à partir d'un hélicoptère par exemple) est possible. Afin de minimiser les coûts de production, et de faciliter le déploiement, tous les nœuds devront être identiques (pas de nœuds « routeurs » spécifiques). Leur interface radio ne peut fonctionner qu'à une seule fréquence, à une puissance fixe, ce qui interdit l'utilisation de multiples canaux de fréquences. Parallèlement, la puissance de calcul sera limitée ne sera pas considérée assez importante pour permettre l'utilisation de CDMA (*Code Division Multiple*

Access), une technique qui consiste à créer plusieurs canaux logiques dans un même canal fréquentiel en utilisant des codes orthogonaux [37]. Comme les nœuds sont fixes, un système de positionnement dynamique tel que le GPS est superflu, rendant en même temps impossible toute synchronisation sur une horloge globale.

Chaque nœud devra connaître un certain nombre de paramètres. Il doit connaître sa position absolue A (on considérera qu'il l'aura apprise au cours d'une phase de programmation de la position par exemple), ainsi que les différentes variables globales :

- max_{range} [1], la portée maximale d'un signal émis,
- $T_{retournement}$, temps nécessaire à l'interface radio pour basculer entre modes émission et réception,
- BW, la bande passante de son interface radio, en bits par seconde (bps).

Sur la zone à surveiller. Le réseau doit être linéaire, c'est-à-dire que pour tout capteur le signal émis atteint au minimum les deux bords du réseau (la portée est supérieure ou égale à la largeur du réseau). La position des nœuds peut ainsi être représentée par leur projection sur l'axe médian. Deux nœuds voisins devront être éloignés au plus d'une portée radio (pour qu'ils puissent communiquer). La distance minimale $dist_{min}$ séparant les deux nœuds voisins les plus proches dans le réseau devra être connue par tous. Enfin, un puits devra être présent à une extrémité de la zone à surveiller.

Sur le lien radio. On suppose que le lien radio est un lien bidirectionnel, c'est-à-dire que si A entend B, alors B entendra A. De plus, si B se trouve entre A et C et que ces derniers peuvent communiquer, alors B pourra communiquer et avec A et avec C. La réalité du lien radio est prise en compte, c'est-à-dire que l'on considère une équation de propagation avec affaiblissement (en $1/d^2$ par exemple, d étant la distance). Néanmoins, les fautes de transmission ne sont pas prises ne compte.

Sur les alarmes. Les alarmes peuvent être émises par n'importe quel nœud, elles ont toutes la même priorité. Ces alarmes atteignent le puits en mode multi sauts. Entre l'occurrence de l'évènement qui déclenche l'alarme, et l'instant de réception de l'alarme par le puits, il y a un *deadline* à respecter (qui dépend de l'application). Notre but est de proposer un protocole qui assure un temps connu et borné. Il faudra ensuite vérifier que ce temps est compatible avec le *deadline* de l'application.

Le réseau après déploiement est illustré par la figure 6. Le carré représente le puits, les cercles les nœuds.

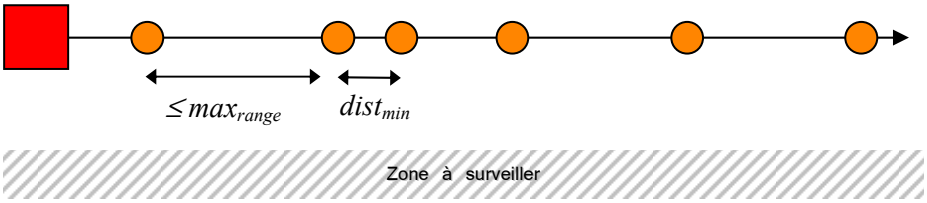


Figure 6. Le réseau après déploiement

[1] Une liste des variables utilisées tout au long de ce document, ainsi que leur signification est consultable dans l'annexe 2.

3. Idée générale de la proposition

Le réseau est découpé en cellules au cours d'une phase d'initialisation. Les cellules sont créées de telle sorte que tout nœud d'une cellule peut communiquer avec tout nœud d'une cellule voisine. Grâce à la façon de former les cellules, une fiabilité est de plus garantie pour cette communication. La remontée d'alarmes durant la phase de fonctionnement se fait selon deux modes : un mode non-protégé et un mode protégé. Le fonctionnement débute en mode non-protégé où la remontée se fait à une vitesse quasi-optimale, mais où plusieurs messages peuvent entrer en collision. Lorsqu'une collision se produit, le réseau bascule en mode protégé, mode plus lent mais dans lequel les collisions sont impossibles. C'est le puits qui décide de rebasculer le réseau en mode non-protégé.

4. Initialisation

L'objectif de la phase d'initialisation est de découper le réseau en cellules. Chaque nœud aura donc deux identifiants possibles : sa position absolue et son identification par rapport à la découpe en cellules. Deux types de messages de signalisation sont utilisés : *CREATION* (C) et *END_INIT* (E). Chaque message contient la position absolue $A_{émetteur}$ de l'émetteur. Un message n'est considéré reçu que lorsqu'il est reçu avec un rapport signal à bruit SNR (*Signal to Noise Ratio*, [44]) supérieur à un seuil, que nous expliciterons à la fin de cette partie. Ceci garantira pendant la phase de fonctionnement une fiabilité dans la communication entre cellules voisines.

Accès au médium. Pour éviter que deux messages n'entrent en collision, chaque nœud détermine un temps d'attente de début d'émission proportionnel à sa distance au dernier nœud ayant émis. Lorsqu'un nœud émet, tout nœud ayant entendu son message détermine son temps d'attente, le *backoff*, à partir de la connaissance de sa position absolue A et de la position absolue de l'émetteur $A_{émetteur}$.

$$backoff = \frac{A - A_{émetteur}}{W_{initialisation}}$$

Ce n'est que lorsque le backoff est écoulé que le nœud peut éventuellement émettre. $W_{initialisation}$ est appelé « vitesse de vague d'initialisation », car la concaténation des instants de début d'émission forme comme une vague balayant l'ensemble des nœuds ayant reçus le message, à partir de la position de l'émetteur. $W_{initialisation}$, exprimé en mètres par seconde, est borné. En effet, si x et y sont deux nœuds voisins, avec $A_x < A_y$, il faut que y ait fini de recevoir le message de x avant que le front de vague ne l'atteigne, c'est-à-dire après la fin de l'émission du message (durée de l'émission et temps de propagation) et le temps de retournement de l'interface radio (passage du mode réception au mode émission). Cette borne sera minimale pour x et y séparés d'une distance géographique $dist_{min}$.

$$W_{initialisation} \leq \frac{dist_{\min}}{T_{propagation} + \frac{longueur_{creation}}{BW} + T_{retournement}}$$

Algorithme d'initialisation.

(1) Le puits émet *CREATION(1)*, créant ainsi la cellule *1*. Tout nœud ayant entendu un message *CREATION* détermine son *backoff*. Durant l'écoulement du *backoff*, tout nœud enregistre la date de réception du dernier message de *CREATION*, le nombre de ces messages reçus, et le numéro de la dernière cellule créée.

(2) Lorsque le *backoff* expire et que le nœud n'a reçu qu'un message *CREATION(i)*, il émet *CREATION(i+1)*. Il fait alors partie de la cellule *i+1*.

(3) Lorsque le *backoff* expire et que le nœud a reçu deux messages *CREATION(i)* et *CREATION(i+1)*, il sait qu'il fait partie de la cellule *i+1*. Il déclenche une temporisation de durée $timer_{faille}$ (voir plus loin) à partir de l'instant de réception de *CREATION(i+1)* et si celle-ci expire sans avoir reçu un autre message, il émet *CREATION(i+2)*. Il fait alors partie de la cellule *i+2*.

(4) Toute émission d'un message *CREATION* s'accompagne du déclenchement d'une temporisation initialisée à $timer_{dernier}$. Si celle-ci expire sans avoir entendu de message, le nœud sait qu'il est dernier du réseau et émet un message *END_INIT(1)*.

(5) Tout nœud qui a émis un message *CREATION* et qui reçoit un message *END_INIT(i)* émet *END_INIT(i+1)*. Le puits sera ainsi averti de la fin de l'initialisation du réseau, mais également du nombre de cellules qui le compose (utilisé plus loin).

$Timer_{faille}$ sert à détecter la situation où un message *CREATION(i+1)* n'a pas été entendu par plus de nœuds que le message *CREATION(i)*. Dans ces cas là, le dernier nœud qui aura entendu les deux messages de *CREATION* considérera qu'il ne doit pas envoyer de nouveau message. Il bloque ainsi sans le savoir l'initialisation du réseau à sa position, tous les nœuds situés en aval n'étant jamais initialisés. C'est pourquoi à l'instant de réception du deuxième message de *CREATION*, il démarre une temporisation de durée $timer_{faille}$. Si après un temps correspondant à $max_{range}/W_{initialisation}$ il n'a pas reçu de message, il sait que l'initialisation s'est arrêtée à cause de lui, et il émet un message *CREATION(i+1)*. S'ils sont plusieurs nœuds dans le même cas, il faut éviter qu'ils n'émettent tous ce message de manière simultanée (engendrant une collision), il faut donc un mécanisme d'élection du nœud le plus loin du nœud émetteur d'entre eux. Cette élection est faite à l'aide du même $timer_{faille}$, à partir de la différence de position entre chaque nœud et la portée maximale du message initial.

$$timer_{faille} = \frac{2 \times \max_{range} - \left(A - A_{CREATION\ (i+1)}\right)}{W_{initialisation}}$$

$Timer_{dernier}$ est égal au temps que met la vague d'initialisation pour arriver à la position limite de réception du message *CREATION* (à l'aide de max_{range}). Si durant l'intervalle de temps aucun nœud n'a émis, c'est qu'aucun nœud ne se trouve entre le nœud qui a émis le message et cette position limite.

$$timer_{dernier} = \frac{\max_{range}}{W_{initialisation}}$$

Exemple.

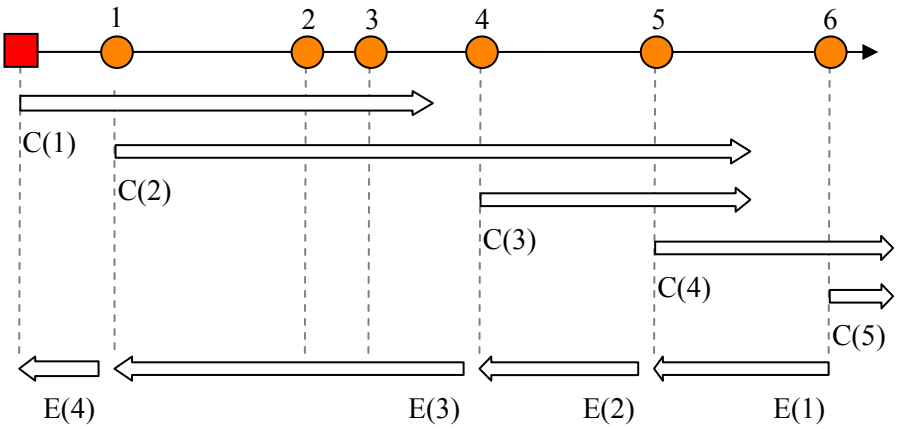


Figure 7. Initialisation

L'exemple représenté sur la figure 7 nous permet d'expliquer le fonctionnement de l'algorithme. Le carré représente le puits, les ronds les nœuds (identifiés par un numéro de nœud) et les flèches la portée des messages. Le puits émet *CREATION(1)*, entendu par les nœuds 1, 2 et 3. Le *backoff* du nœud 1 est le premier à expirer, il émet *CREATION(2)* puisqu'il n'a reçu qu'un seul message *CREATION*. L'expiration des *backoffs* des nœuds 2 et 3 n'entraîne pas d'émission puisqu'ils ont entendu deux messages, ils font partie de la cellule 2. Le nœud 4 est le prochain nœud à émettre *CREATION(3)*, puisqu'il n'a reçu que *CREATION(2)*. On arrive à un cas critique puisque ce message n'est pas entendu par plus de nœuds que *CREATION(2)* : lorsque le backoff du nœud 5 expire, il a reçu deux messages donc ne fait rien. Il faudra attendre l'expiration de son $timer_{faille}$ pour que le nœud 5 se rende compte qu'il a coupé le processus d'initialisation, et qu'il émette *CREATION(4)*. Enfin, le nœud 6 émet *CREATION(5)*, mais après n'avoir rien entendu durant $timer_{dernier}$, il se rend compte qu'il est le dernier nœud du réseau et émet *END_INIT(1)*, relayé par tous les nœuds ayant émis un message *CREATION*. Lorsque le puits recevra le message *END_INIT*, il saura que l'initialisation est terminée, et connaîtra le nombre de cellules (utilisé plus tard).

Identification. Chaque nœud met en mémoire les positions absolues des nœuds qui ont émis le message de *CREATION* de sa cellule *i*, et de la prochaine *i+1*. Connaissant sa position absolue *A*, il calcule sa position relative *R* dans la cellule, exprimée ici en pourcentage. Ainsi, à la fin de l'initialisation, un nœud pourra être identifié de deux manières : sa position absolue *A* ou le doublet *[I,R]*, *I* étant le numéro de la cellule à laquelle il appartient, *R* sa position relative dans cette cellule.

Fiabilité. Nous voulons qu'un message émis par un nœud de la cellule *i* soit reçu par un nœud d'une cellule voisine avec une fiabilité donnée. La fiabilité, définie physiquement par un BER (*Bit Error Rate*), est reliée directement au rapport signal à bruit SNR (*Signal to Noise Ratio*) [44] par la courbe présentée dans la Figure 8(a). Le SNR d'un signal reçu est mesurable par l'interface radio du capteur. Le cas de communication pire est lorsque le premier nœud de la cellule *i* veut communiquer avec le dernier nœud de la cellule *i+1* (figure 8(b)). Or cette

communication a déjà eu lieu lors de la formation des cellules, lorsque le premier nœud de i a émis le message $CREATION(i)$; ce message avait été entendu par le nœud qui allait ensuite être identifié comme le dernier nœud de la cellule $i+1$. Ce nœud avait reçu le message car il présentait un SNR supérieur à un seuil. Ainsi, il suffit de fixer ce seuil en fonction du BER, donc de la fiabilité voulue. Toute communication entre cellules voisines est donc ainsi rendue fiable.

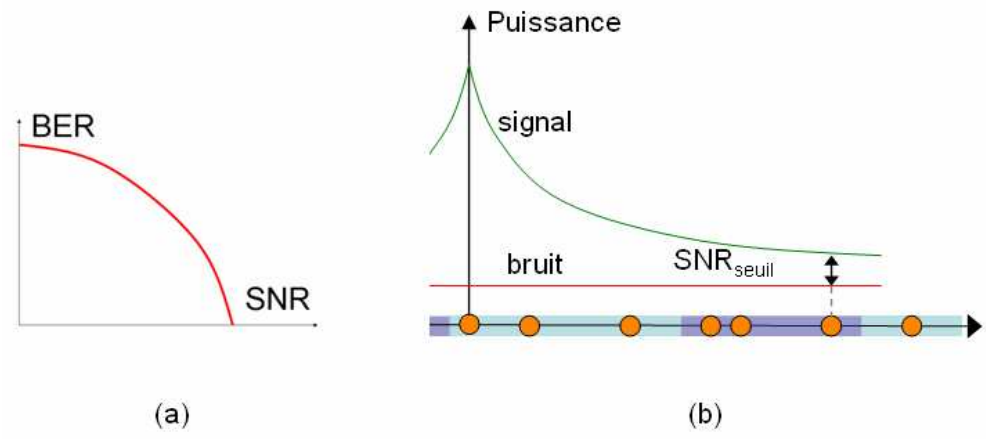


Figure 8. Relation entre BER et SNR (a) détermination de la taille des cellules (b)

5. Fonctionnement en mode non-protégé

Ce mode est utilisé tant qu'il n'y a pas de collision. S'il y a peu d'alarmes, la probabilité de collision est faible. La vitesse de transmission sur le réseau est alors la caractéristique à optimiser.

Comme illustré par la figure 9, lorsqu'un nœud envoie un message d'alarme, un ensemble de nœuds plus proches du puits que l'émetteur l'entendent, et démarrent un *backoff*. Celui-ci est déterminé pour chaque nœud à partir de la position absolue du nœud A, de la position absolue du nœud émetteur $A_{émetteur}$, et de la portée maximale max_{range}, relativement à une vitesse $W_{émission}$. Dès qu'un nœud est élu (son $backoff_{non\text{-}protégé}$ expire), il émet le message. Les autres nœuds entendent ce message (puisqu'ils sont situés entre le nœud élu et le nœud ayant émis le message précédent) et arrêtent leurs temporisations. Lors de l'initialisation, le mécanisme de *backoff* est utilisé pour donner la parole aux nœuds dans l'ordre de position absolue croissante ; ici, il s'agit d'élire le nœud ayant reçu le message le plus lointain du nœud émetteur.

$$backoff_{non-protégé} = \frac{A - \left(A_{émetteur} - \max_{range}\right)}{W_{émission}}$$

La vitesse $W_{émission}$ doit être bornée. En effet, il faut qu'un nœud ait le temps de détecter le début d'émission de son voisin le plus proche avant que $backoff_{non\text{-}protégé}$ n'arrive à expiration. Il est intéressant de remarquer que cette vitesse peut être grande, $T_{propagation}$ et $T_{détection}$ étant petits.

$$W_{émission} \leq \frac{dist_{\min}}{T_{propagation} + T_{détection}}$$

Remarque : Le cas théorique optimal est celui où un message d'alarme émis est relayé par le nœud le plus proche du puits, parmi les nœuds qui ont entendu le message. Seul le temps du processus d'élection du nœud le plus proche du puits ayant reçu le message sépare notre solution de l'optimal.

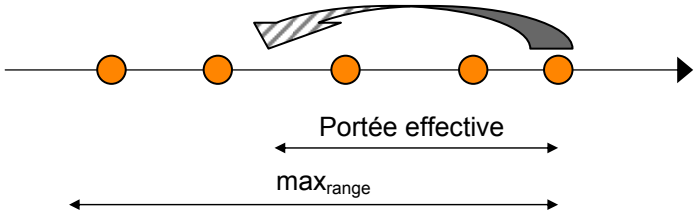


Figure 9. Fonctionnement du mode non-protégé

6. Basculement entre mode non-protégé et mode protégé

Le réseau débute en mode non-protégé. Lorsqu'un nœud émetteur n'entend pas son alarme réémise après une durée correspondant au parcours de max_{range} à la vitesse $W_{émission}$, ou lorsqu'il entend un message incompréhensible, il détecte une collision. Il émet alors un message de bourrage JAM. Dès qu'un nœud entend ce message, il le réémet une seule fois. Il est important de noter qu'une collision avec un message JAM génère un nouveau message de ce type, le processus n'est donc pas stoppé. Tous les nœuds attendent ensuite le temps pire d'exécution du processus de bourrage (explicité dans le chapitre IV) pour reprendre les envois avortés par la collision, cette fois-ci en mode protégé.

D'un côté, le mode non protégé offre une vitesse de transmission quasi-optimale (différence due au temps d'élection du nœud relais) ; d'un autre côté, un mode protégé, déterministe et sans collisions, est nécessaire pour prendre en compte des contraintes temps-réel dur. Lorsqu'une collision est détectée, le réseau bascule entièrement du mode non-protégé au mode protégé.

Remarque: on est obligé de basculer le réseau entier en mode protégé. Si on avait seulement une partie basculée, les alarmes remontant en mode non-protégé « rattraperaient » le tronçon protégé, créant des collisions.

Synchronisation. Nous n'avons pas de système de synchronisation global tel que le GPS, mais afin d'éviter toute collision dans le mode protégé, une synchronisation du réseau est nécessaire. Nous présentons ici une synchronisation basée sur les horloges internes des nœuds. A la fin du processus de bourrage, le puits émet un message $SYNC$, entendu par un certain nombre de nœuds. Ceux-ci démarrent à cet instant une temporisation de durée $backoff_{synchronisation}$, égale à la différence de leur position relatives et celle de l'émetteur $A_{émetteur}$, relativement à une vitesse $W_{synchronisation}$ que nous expliciterons.

$$backoff_{synchronisation} = \frac{R - R_{émetteur} + (A - A_{émetteur}) \times 100}{W_{synchronisation}}$$

Sur chaque nœud, l'expiration de $backoff_{synchronisation}$ lance le démarrage d'une horloge périodique de période T.

$$T = \frac{600}{W_{synchronis\ ation}}$$

Cette période correspond au temps de parcours de 6 cellules à une vitesse $W_{synchronisation}$. En effet, comme $W_{synchronisation}$ est exprimé en pourcentage de cellule par seconde, le numérateur

correspond bien à 600% d'une cellule, soit 6 cellules. En outre, les nœuds à $R=0$ émettent à cet instant un message $SYNC$, afin de propager la synchronisation sur tout le réseau.

L'effet global, en observant les différents instants d'expiration d'abord de $backoff_{synchronisation}$ puis périodiquement des horloges T, est un train de vagues virtuelles balayant le réseau à partir du puits, et qui détermine les instants de début d'émission des nœuds en passant par-dessus. Ces vagues sont espacées de 6 cellules.

En mode protégé, un nœud ne pourra émettre que lorsqu'une vague passera au dessus de lui (donc lorsque $backoff_{synchronisation}$ ou T expire), deux émissions simultanées seront donc espacées de 6 cellules. Comme nous considérons que lorsqu'une cellule i émet, le message est entendu par les cellules $i-2$, i-1, i, i+1 et $i+2$, et brouille $i-3$ et $i+3$, l'espacement de 6 cellules suffit pour qu'un message ne vienne pas brouiller la réception de l'autre. Si le rayon de brouillage est plus grand, il suffit d'augmenter l'espacement entre les vagues.

De même qu'avec la phase d'initialisation, la vitesse de la vague de synchronisation $W_{synchronisation}$ est bornée. Il s'agit de la même vitesse que celle exprimée pour la vague d'initialisation mais le dénominateur est changé pour l'exprimer en pourcentage de cellules par seconde.

$$W_{synchronis\ ation} \leq \frac{\frac{dist_{\min}}{\max_{range}}}{T_{propagatio\ n} + \frac{taille_{sync}}{BW} + T_{retourneme\ nt}}$$

7. Fonctionnement en mode protégé

Le réseau bascule en mode protégé dès qu'une collision a lieu dans le mode non-protégé. Ce nouveau mode doit donc garantir un envoi sans collision, déterministe et en temps connu et borné, quitte à être plus lent que le mode non-protégé. L'idée directrice est d'éviter toute collision entre alarmes en réservant, avant chaque émission d'une alarme, à chaque saut, un certain nombre de cellules en amont. Deux alarmes émises simultanément sont ainsi séparées d'une distance suffisamment grande pour ne pas entrer en collision. Cette réservation est faite à l'aide de messages de signalisation. Pour éviter que ces messages n'entrent en collision à leur tour, la synchronisation à l'aide des vagues est utilisée. Enfin, pour garantir un envoi fiable et donc éviter la perte de messages de signalisation, ceux-ci ne sont échangés qu'entre cellules voisines. Trois types de messages sont utilisés ($SILENCE$, ACK_EXP et $DATA$), chacun contenant l'identifiant du nœud émetteur, les deux premiers sous la forme $[I,R]$, le message $DATA$ par sa position absolue A. Il est important de remarquer qu'une fois la zone protégée établie, il est possible d'envoyer un (et un seul) message en mode non-protégé.

Algorithme de protection.
Cet algorithme est illustré par la figure 10.

(1) Un nœud de la cellule i qui veut faire remonter une alarme attend le passage d'une vague et émet un message $SILENCE(1)$. Tous les nœuds de la cellule i entendent ce message et se mettent dans l'état $RESERVE(0)$ et ceux de la cellule $i-1$ (voisine plus proche du puits) dans l'état $RESERVE(1)$. Des nœuds réservés ne peuvent pas émettre de nouvelle alarme. Un message $SILENCE(2)$ doit être envoyé pour réserver l'ensemble des nœuds de la cellule $i-2$.

(2) Un nœud relais est pour cela élu, en utilisant l'algorithme d'élection à charge répartie (*load-balanced*) que nous présentons dans le paragraphe suivant.
(3) Ce nœud émet un message *SILENCE(2)*.
(4) Ce processus de réservation multi-sauts continue jusqu'à ce que la cellule *i-5* se mette dans l'état *RESERVE(4)*. La cellule *i-5* renvoie un message *ACK_EXP* d'acquittement explicite de fin de protection, à la fin du processus d'élection du nœud relais.
(5) Ce message *ACK_EXP*, comme il est transmis à l'intérieur de la zone protégée, peut être transmis de la même manière qu'en mode non-protégé.
(6) Le nœud émetteur reçoit ce message. Ceci marque la fin de la phase de protection.

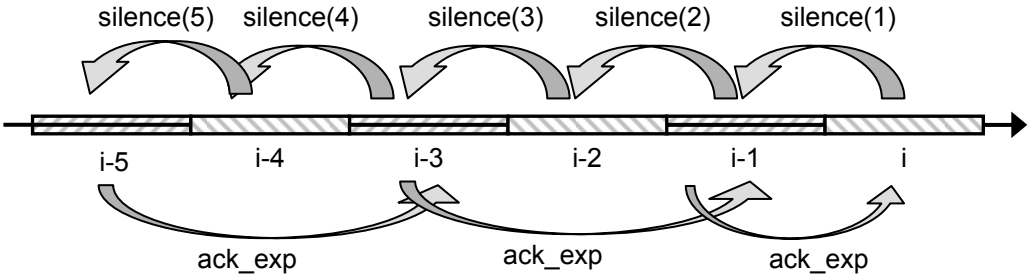


Figure 10. Fonctionnement du mode protégé

Election du nœud relais. Seul un nœud relais peut être élu. L'algorithme suivant est utilisé (illustré par la figure 11), et dure exactement T (période de passage de vagues). L'idée derrière cet algorithme est que le nœud relais reste approximativement à la même position relative que le nœud émetteur par souci de répartition de charge entre nœuds d'une même cellule (*load-balancing*). Cet algorithme est robuste, vu qu'un nœud est élu dès que la cellule en compte un.

(1) Un nœud entend un message *SILENCE* et enregistre la position relative de l'émetteur.
(2) Sachant sa propre position relative, l'instant où le front de la vague l'atteint, et la vitesse de vague $W_{synchronisation}$, il détermine l'instant où le front atteint de premier nœud de sa cellule, et attend cet instant.
(3) Si sa position relative est plus petite que celle de l'émetteur du signal *CREATION* (i.e. il est plus proche du puits, relativement), il attend un temps proportionnel à la différence de positions relatives, relativement à $W_{synchronisation}$.
(4) Si sa position relative est plus élevée, le temps d'attente est proportionnel à sa position relative.
(5) Le premier nœud qui voit son temps d'attente écoulé est élu nœud relais et émet immédiatement le message approprié (*SILENCE* ou *ACK_EXP*). Cette émission annule les temps d'attente des autres nœuds.

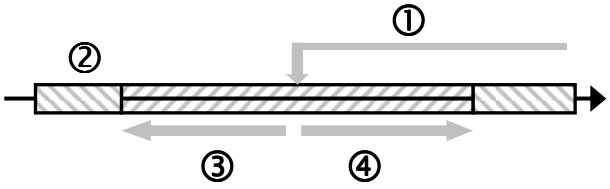


Figure 11. Election du nœud relais

Remarque : En favorisant en premier les nœuds plus loin de l'émetteur, on compense le fait que le signal est mieux entendu près de l'émetteur. Cette solution évite donc de sur-utiliser les nœuds situés aux extrémités des cellules, ce qui réduirait leur temps de vie.

Transmission. Après la phase de protection, le message d'alarme peut être transmis. Comme la protection a été mise en place (aucune collision n'est possible dans la zone protégée), et comme la limite de la zone protégée se trouve au-delà de la portée maximale (max_{range}), l'alarme peut être envoyée en mode non-protégé. Le nœud qui sera élu relais ne réémettra pas directement l'alarme mais lancera une nouvelle phase de protection. L'alternance entre phases de protection et de transmission génère la transmission en mode multi-sauts de l'alarme.

Comportement du puits. Nous ne nous étions jusque là pas intéressés au comportement du puits. Pour que les nœuds proches n'aient pas à être différenciés, le puits doit agir comme la continuité du réseau (cellules 0, -1, …). Il doit pour cela combiner le comportement de la cellule 0 (i.e. émission de message $CREATION$) et de la cellule 1 (e.g. réponse par message $SILENCE$ ou ACK_EXP à un message $SILENCE$). Le puits absorbe ainsi de manière transparente pour le réseau le flux des alarmes.

Remarque : le nombre de cellules à réserver dépend du rayon d'interférence maximal des nœuds. Nous avons fait l'hypothèse qu'une émission de la cellule *i-6* interférerait jusqu'à la cellule *i-3*. Les cellules *i-2* et *i-1* sont donc capables d'entendre un message venant de la cellule *i*. Néanmoins, si la zone d'interférence est plus grande, il suffit d'augmenter le nombre de cellule à réserver, ainsi que la distance entre deux vagues consécutives.

8. Basculement entre mode protégé et mode non-protégé

Alors que le basculement vers un mode protégé est effectué après une collision, aucun évènement ne conditionne le re-basculement en mode non-protégé. Il doit se faire lorsque le réseau est suffisamment peu chargé pour ne pas qu'il y ait immédiatement une nouvelle collision. Il faut donc éviter à la fois un re-basculement trop hâtif (qui engendrerait immédiatement une collision), et un re-basculement trop tardif (c'est-à-dire un ralentissement inutile de la remontée des alarmes). Afin de ne pas avoir de spécialisation à l'intérieur des capteurs, le puits se charge de la décision de basculer du mode protégé au mode non-protégé.

Après une durée de non réception de messages correspondant au temps qu'aurait mis un message à remonter la longueur totale du réseau (temps explicité dans le chapitre IV), le puits considère que le réseau est assez peu chargé en alarmes et émet un message de type JAM, traité par le réseau de la même manière que pour le basculement vers le mode protégé. Après avoir relayé le message JAM, les capteurs attendent le temps pire d'exécution de ce processus de bourrage $WCET_{JAM}$ (voir chapitre IV) et réémettent leurs alarmes en mode non protégé lorsque le médium est libre.

9. Conclusion

Notre proposition de protocole MAC pour réseaux de capteurs se base sur un accès au médium régulé en fonction de la position géographique du nœud. Un fonctionnement selon deux modes (protégé et non-protégé) assure à la fois le déterminisme du protocole (afin d'assurer des caractéristiques temps-réel dur), et un temps de remontée d'alarmes quasi-optimal lorsque peu d'alarmes sont présentes sur le réseau. L'initialisation a une durée connue et bornée, elle peut donc être relancée en cours de fonctionnement pour faire face à des changements de conditions (par exemple, portée radio diminuée par la pluie) sans affecter sa caractéristique temps-réel. Les temps pire d'exécution de l'initialisation et du basculement, ainsi que les temps de remontée d'alarmes sont connus et bornés, et explicités dans la partie suivante.

Chapitre IV

Modélisation et Validation

Le but de ce chapitre est de valider formellement notre proposition de protocole MAC temps-réel, présentée dans le chapitre précédent. Nous utilisons pour cela la méthodologie de validation formelle présentée au chapitre II, qui est basée sur le formalisme TSA (*Timed Safety Automata*) [8] de l'outil UPPAAL [51, 30]. Cette validation est faite à la fois sur le comportement du protocole et sur ses caractéristiques temporelles.

1. Modélisation

1.1. Méthodologie

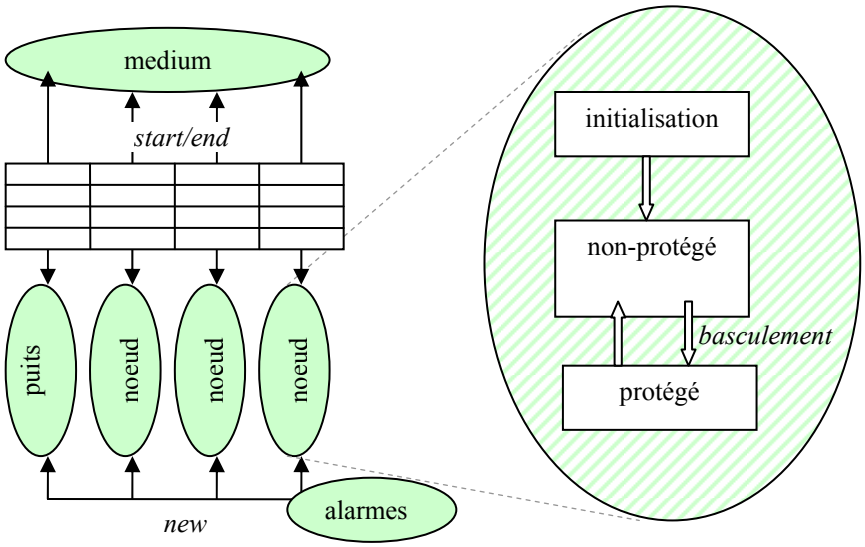


Figure 12. Méthodologie de modélisation

Le système à modéliser peut être vu comme un ensemble de composants (les différents nœuds, le puits, le medium, …). La modélisation d'un composant se fait selon une logique proche du paradigme orienté objet: les automates permettent la création de différentes instances. Ainsi, la modélisation du système entier s'axe autour de 4 automates: le puits, le nœud, l'automate représentant le médium radio et celui modélisant la génération d'alarmes. Pour générer une topologie, il suffit d'instancier correctement les objets modélisant le comportement du nœud et de paramétrer l'automate *medium* en fonction des portées radio de chaque nœud. Ce découpage est présenté dans la figure 12 avec un exemple de réseaux à trois capteurs.

La communication entre les instances se fait à l'aide de signaux. Deux problèmes se posent : l'échange de données entre nœuds, et la modélisation d'une transmission avec une durée non nulle.

Echange de données. Il est impossible de paramétrer un signal, c'est-à-dire de lui affecter une valeur afin de transmettre une donnée. Pour contourner ce problème, nous avons défini une variable globale. La communication avec transmission de données consiste alors à copier la

valeur que l'on veut émettre dans la variable globale et d'avertir par signal le destinataire qui ira lire la variable. C'est le mécanisme de base de l'échange de messages en nœuds. Sur la figure 12, cette variable globale est représentée par un tableau entre les nœuds et le medium.

Durée de transmission. La transmission des messages doit avoir une durée non nulle. Or l'émission et la réception d'un signal entre deux automates est instantanée. Pour contourner ce problème, on utilise deux signaux : un pour avertir du début de l'émission et un autre pour la fin. Ainsi, lorsqu'un nœud veut envoyer un message à un autre nœud, il envoie un signal pour avertir le destinataire du début de l'émission et même temps qu'il copie les données dans la variable globale, et après le temps voulu envoie le message de fin d'émission. Le destinataire peut alors aller lire la variable.

Portée radio des noeuds Nous avons choisi de créer un automate spécial représentant les différentes portées radio de chaque noeud: *medium*. Lorsque le nœud x émet en diffusion un message, il l'envoie en réalité à l'automate medium qui renvoie le message à l'ensemble des nœuds se situant à portée radio de x. Il faut donc installer un canal de communication indépendant entre chaque nœud et l'automate medium, constitué d'une paire de signaux (*start* et *end*) et d'une variable globale propre pouvant accueillir un message. Un message sera constitué au plus de quatre champs, la variable globale sera donc un vecteur de longueur 4. Afin d'organiser de manière efficace l'ensemble de ces canaux, nous avons choisi de créer les vecteurs $start[nombre_{noeuds}]$ et $end[nombre_{noeuds}]$ et la matrice *mat_medium[nombrenoeuds][4]*, et de numéroter les nœuds en commençant par 0 pour le puits. Ainsi, lorsque le nœud 3 veut émettre un message, il émet le signal *start[3]*, copie sont message dans *mat_medium[3][0]*, *mat_medium[3][0]*, *mat_medium[3][0]* et *mat_medium[3][0]*, et enfin il émet le signal *end[3]*.

Découpage en étapes. Pour chaque nœud, nous découpons le protocole complet en quatre étapes: l'initialisation, le fonctionnement en mode non-protégé, le basculement entre modes, et le fonctionnement en mode protégé. Ainsi, on retrouve dans l'automate du nœud ces quatre étapes.

1.2. Exemple d'automate.

Nous ne pouvons par manque de place décrire tous les automates nécessaires pour modéliser complètement le protocole. Nous décrivons donc uniquement le modèle UPPAAL du comportement du nœud dans le mode non-protégé (présenté dans la figure 13). Un certain nombre de variables sont utilisées : A la position absolue du nœud, N son numéro dans le réseau (utilisé seulement pour faciliter la modélisation), x une horloge interne.

Nous allons décrire la figure 13. L'automate représenté démarre dans l'état initial (en haut, le double cercle). Le nœud commence par instancier ses variables A et N *(respectivement à 200 et 12)*, et attend la réception d'un message (*start[N]?*). Lorsque ce signal arrive, il stocke le message reçu dans son *buffer* de message interne (*message[N]:=medium[N][0]*…). Il attend ensuite la réception du signal de fin de transmission (*end[N]?*) et utilise cet instant pour initialiser à zéro son horloge interne. Il vérifie que le message est bien issu d'un message en aval (*message[1]*>A), sinon, il se replace en attente. Si c'est bien le cas, il attend l'expiration du $backoff_{non\text{-}protégé}$ (*message[1]-max_range)*). A cet instant, il émet le message en émettant le signal de début d'émission (*start[N]!*), initialise son horloge à zéro et copie son message dans la variable globale. Après le temps d'émission (x==*duration_DATA*), il envoie le signal de fin d'émission *end[N]!* et se met en attente. Si au cours de l'attente de l'expiration de $backoff_{non\text{-}protégé}$, il reçoit un message, il attend la fin de celui-ci, relance son $backoff_{non\text{-}protégé}$ et se place en attente. C'est le cas où un autre nœud est élu avant lui. Enfin, il peut recevoir un

signal *new* venant de l'automate de génération des alarmes. Cela signifie qu'une nouvelle alarme a été détectée par ce nœud et doit être transmise au puits.

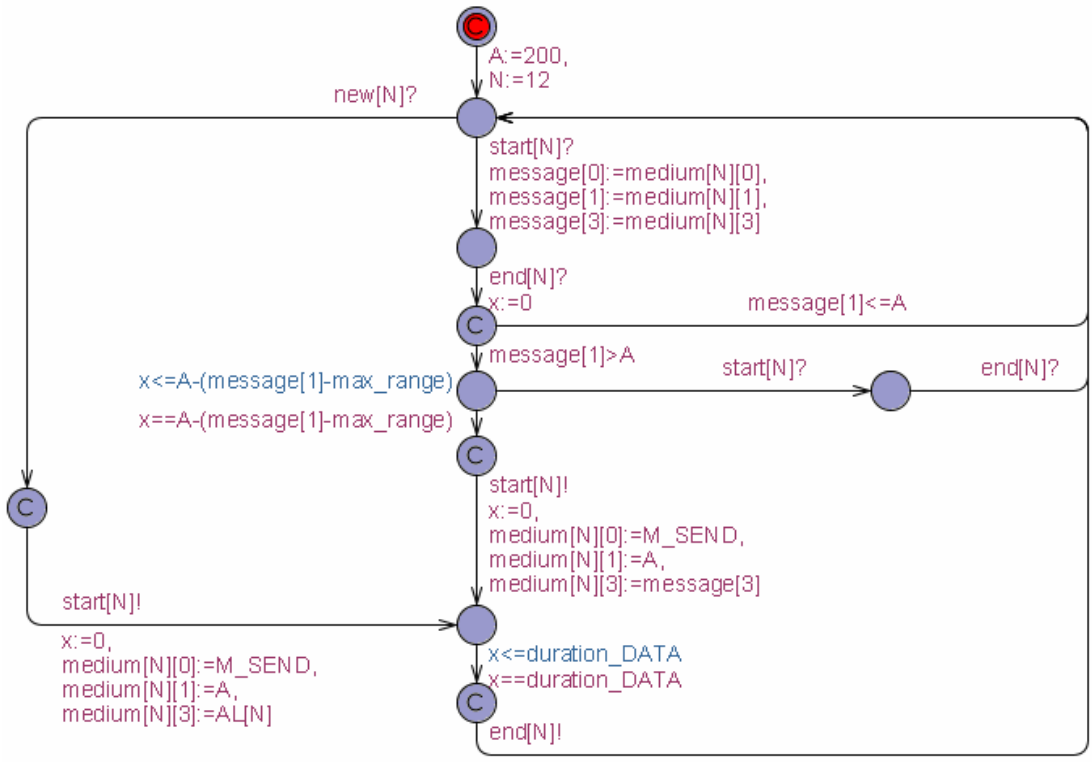


Figure 13. Modélisation du comportement du nœud dans le mode non-protégé

2. Validation temporelle

2.1. Méthodologie

Dans un premier temps, par analyse du protocole proposé, on identifie les pire cas en termes de durée d'exécution ou de transmission sur les quatre parties, et on exprime ces durées analytiquement. Il s'agit des WCET (initialisation, basculement) et WCTT (fonctionnement en mode protégé ou non-protégé). Une fois cette analyse faite, on modélise le système à l'aide d'UPPAAL et on effectue sa validation comportementale et temporelle avec la méthodologie de *model-checking* présentée au chapitre II. Les deux méthodes sont complémentaires. Le *model-checking* permet de vérifier formellement que le comportement du protocole modélisé est conforme aux spécifications, et permet de vérifier les valeurs de WCET et WCTT en prenant les scénarios pires. La méthode analytique permet d'avoir une formule paramétrée (par les différentes caractéristiques du réseau de capteurs) de la borne dans le cas pire.

2.2. WCET analytiques

Afin d'extraire les temps pire d'exécution ou de transmission des différentes étapes du protocole (initialisation, basculement, modes de fonctionnement), on identifie les cas pires et on exprime leur durée. Par manque de place, nous allons effectuer cette analyse uniquement pour le mode non-protégé, et exprimer $WCTT_{non\text{-}protégé}$.

Le pire cas en termes de durée est lorsque l'alarme doit aller le plus loin, donc venir du dernier nœud du réseau. De plus, le temps de transmission sera le plus grand lorsque l'alarme aura à effectuer un saut par nœud. Pour chaque saut, deux durées rentrent en compte : le temps de transmission et le temps d'élection du voisin le plus lointain. Le temps de transmission est toujours constant et calculé avec $taille_{data}$ et BW. Le mécanisme d'élection

(qui ne survient que lorsque le nœud émetteur fait partie d'une cellule de numéro supérieur ou égal à 3, donc qu'il ne peut pas envoyer directement l'alarme au puits) se fait en vague de vitesse $W_{émission}$. Cette vague devra parcourir en moyenne max_{range}-$(longueur_{réseau}/ nombre_{noeuds})$, donc au total $nombre_{noeuds}$ fois cette valeur. Ces différentes observations sur le pire cas conduisent à la formulation suivante de la pire durée.

$$WCTT_{non-protégé} = nombre_{noeuds} \times \left[\frac{taille_{data}}{BW} + \left(\frac{\max_{range} - \frac{longueur_{réseau}}{nombre_{noeuds}}}{W_{émission}} \right)_{si\ puits\ non\ connecté} \right]$$

$WCET_{initialisation}$, $WCET_{jam}$ et $WCTT_{protégé}$ sont obtenus de manière similaire[2].

$$WCET_{initialisation} = \frac{longueur_{réseau}}{W_{initialisation}} + \left\lceil \frac{nombre_{noeuds} - 1}{2} \right\rceil \times \frac{2 \times \max_{range}}{W_{initialisation}} + \frac{2 \times \max_{range}}{W_{initialisation}} + (nombre_{cellules} - 1) \times \frac{taille_{end\_init}}{BW}$$

$$WCET_{jam} = (nombre_{noeuds} + 1) \times \frac{taille_{jam}}{BW} + \frac{(number_{cellules} - 1) \times 100}{W_{synchronisation}} + \frac{taille_{sync}}{BW} + 1$$

$$WCTT_{protégé} = (nombre_{cellules} - 1) \times \frac{taille_{data}}{BW} + \left(3.T + \frac{taille_{ack\_exp}}{BW}\right) + \left(3.T + 2.\frac{taille_{ack\_exp}}{BW}\right)_{si\ nombre_{cellules} \geq 3} + \left(4.T + 3.\frac{taille_{ack\_exp}}{BW}\right)_{si\ nombre_{cellules} \geq 4} + \left(5.T + 4.\frac{taille_{ack\_exp}}{BW}\right)_{si\ nombre_{cellules} \geq 5} + \left(6.T + 5.\frac{taille_{ack\_exp}}{BW}\right)_{si\ nombre_{cellules} \geq 6} + \left(6.T + 5.\frac{taille_{ack\_exp}}{BW}\right)_{si\ nombre_{cellules} \geq 7} \times (nombre_{cellules} - 5)$$

2.3. Scénarios détaillés

Afin de valider les différents pires temps proposés, nous utilisons des scénarios avec différents déploiements des nœuds, différentes portées radio et différentes occurrences d'alarme. Nous vérifions sur ces scénarios que le comportement du système est bien celui prévu, puis que les durées (d'exécution ou de transmission) soient bornées en utilisant une valeur très grande pour ensuite retrouver la borne exacte par dichotomie. Cette borne doit être inférieure ou égale au WCET. Une analyse fine du scénario permet de paramétrer la borne et de montrer pourquoi celle-ci est plus ou moins éloignée du cas pire. L'éloignement entre le WCET et la durée effective quantifie l'éloignement entre le cas pire et le cas effectif.

Nous nous proposons d'analyser en détail de déroulement d'une phase du protocole sur un exemple afin de vérifier le bon comportement du protocole (comportemental et temporel)

[2] Une liste des variables utilisées tout au long de ce document, ainsi que leur signification est consultable dans l'annexe 2.

dans des situations « critiques ». C'est ainsi que dans un premier temps, nous illustrerons le cas appelé faille dans l'initialisation, ensuite nous déterminerons le temps de remontée d'une alarme en mode non-protégé sur le même cas d'étude. Nous travaillons sur le cas représenté dans la figure 14. Le carré représente le puits, les ronds les nœuds (identifiés leur position absolue A). Les barres horizontales représentent la portée des noeuds. $max_{range}=100$, $W_{initialisation}=1$, $BW=1$, et $taille_{CREATION}=taille_{END\_INIT}=3$, $taille_{data}=10$.

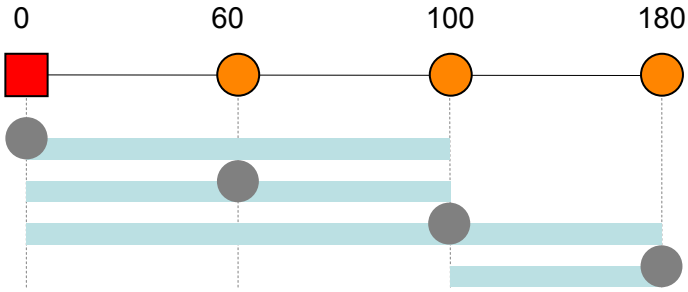


Figure 14. Le cas d'étude

Cas de faille. Dans un premier temps, nous allons dérouler l'algorithme d'initialisation sur cet exemple, qui a été choisi parce qu'il déclenche le cas de *faille,* afin d'effectuer une analyse comportementale du système (*le système va t'il bien réagir au cas de faille ? l'initialisation ne va-t-elle pas être stoppée ?*). Le cas de faille se produit lorsqu'au cours de l'initialisation le message CREATION(i+1) n'est pas entendu par plus de nœuds que le message CREATION(i). Il en résulte que l'initialisation est stoppée à un nœud, tous les nœuds situés en aval n'étant pas initialisés. L'utilisation de $timer_{faille}$ permet de résoudre ce problème.

Le puits émet $CREATION(1)$ à $t=0$ (t étant le temps global), le nœud à 60 émet $CREATION(2)$ à $t=60$, et la vague atteint le nœud à 100 à $t=100$. On est dans un cas de *faille*, et le nœud 100 attend $timer_{faille}=120$ donc $t=220$ avant d'émettre $CREATION(3)$. A $t=300$, le backoff du nœud à 180 expire et il émet $CREATION(4)$. Après $timer_{dernier}=200$, donc à $t=500$, il détecte qu'il est le dernier nœud. Il émet alors le message END_INIT, relayé par les nœuds de position 100 et 60, soit trois émissions pour une durée totale de 9. Le réseau est initialisé à $t=509$. Cette valeur est confirmée par l'outil UPPAAL, dans lequel nous avons modélisé le même cas.

D'après la formule $WCET_{initailisation}=180+200+200+9=589$. La différence entre cette formule et le cas effectif réside dans le fait que $timer_{faille}$ pire est de 200. Dans le cas étudié, il est de 120, le nœud de position 100 étant éloigné de 40 du nœud de position 60.

Mode non-protégé. Nous analysons maintenant le cas d'une remontée d'alarme depuis le nœud de position absolue 180 sur le même cas d'étude représenté par la figure 14. Nous déroulons donc le protocole, plus particulièrement la phase non-protégée.

Le nœud à 180 émet une alarme de $t=0$ à $t=10$. Elle est reçue par le nœud à 100 qui évalue $backoff_{émission}=20$. Entre $t=30$ et $t=40$, il réémet cette alarme qui est reçue par le puits. La durée de remontée de l'alarme a donc duré 40. Cette valeur est confirmée par l'outil UPPAAL, dans lequel nous avons modélisé le même cas.

L'application numérique du WCET pour 3 nœuds donne $WCET=3*(10+(100-60))=150$. La différence entre ce cas et le cas pire est qu'ici, le nœud à 60 ne doit pas réémettre une

alarme ni effectuer l'élection du nœud relais, et que le nœud à 100 ne doit pas non plus effectuer cette élection.

2.4. Ensemble des cas étudiés par les scénarios

Dans la partie précédente, nous avons détaillé deux scénarios en vérifiant en parallèle le comportement et les caractéristiques temporelles, avec l'outil UPPAAL et « à la main ». Il est bien sûr fastidieux de continuer la validation manuelle, l'outil UPPAAL est utilisé pour systématiser la validation.

Cette validation s'appuie sur un ensemble de scénarios et de propriétés à valider. Elle a permis de valider l'ensemble des caractéristiques comportementales ou temporelles suivantes.

Phase d'initialisation.

- Détermination correcte de $backoff_{initialisation}$,
- cas appelé faille dans l'initialisation,
- détection correcte par un nœud qu'il est le dernier du réseau
- initialisation correcte du réseau, avec identification *[I,R]* et respect de $WCET_{initialisation}$,

Mode de fonctionnement non-protégé.

- remontée d'alarmes correcte en mode non-protégé, avec respect de $WCET_{non\text{-}protégé}$,

Basculement entre modes.

- Basculement correcte du réseau entre les modes de fonctionnement, avec respect de $WCET_{jam}$,

Mode de fonctionnement protégé.

- élection d'un seul nœud relais dans le mode protégé,
- basculement entre phases de protection et de transmission dans le mode protégé,
- décalage de la zone protégée dans une transmission multi-sauts des alarmes dans le mode protége,
- remontée d'alarmes correcte en mode protégé, avec respect de $WCET_{protégé}$.

3. Conclusion

Dans ce chapitre, nous avons modélisé notre protocole MAC temps-réel à l'aide d'UPPAAL. Grâce au modèle obtenu et par *model-checking*, nous avons validé formellement que le comportement de notre protocole est bien celui que nous souhaitions, et que les bornes temporelles de ses différents services (initialisation, bascuelement de mode, remontée d'alarme en mode protégé ou non-protégé) sont bien égales à celles que nous avions trouvées de manière analytique. Donc, le fait d'avoir obtenu des bornes pour l'initialisation, le basculement et pour la remontée des alarmes permet effectivement d'implanter une application ayant des contraintes temps réel fortes.

Chapitre V

Conclusion et Perspectives

Ce travail est parti du constat que la plupart des travaux de la littérature présentent des solutions temps-réel relâché, où une part des messages peut arriver après leur deadline, sans affecter l'application de manière significative. Ici, nous nous sommes intéressés au temps réel dur, qui doit garantir pour chaque message un temps de transmission connu et borné.

Nous avons proposé un protocole MAC temps-réel dur en posant des hypothèses aussi réalistes que possible. Notre solution, basée sur l'alternance entre deux modes, permet à la fois un temps de remontée des alarmes quasi-optimal en mode non-protégé, et un mode déterministe sans collisions en mode protégé. La réalité du lien radio est prise en compte, avec une découpe du réseau en cellules, garantissant une transmission fiable.

Une validation comportementale et temporelle a été effectuée à l'aide d'une méthodologie de type *model-checking* avec l'outil UPPAAL. Les WCET (*Worst Case Execution Time*) et WCTT (*Worst Case Transmission Time*) sont exprimés et validés.

Ce travail a donné lieu à la soumission d'un article de recherche au « 13th Annual Meeting of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'05) », qui aura lieu à Atlanta aux Etats-Unis, du 26 au 29 septembre 2005.

Actuellement, dans l'étude effectuée dans ce rapport le réseau permet de remonter une alarme à la fois par cellule. Une étude intéressante à mener consisterait à regarder combien d'alarmes par cellule et par unité de temps le réseau est capable d'acheminer en continuant d'assurer le respect des contraintes temps-réel.

De plus, au niveau de la couche physique, nous avons considéré l'affaiblissement du signal avec l'éloignement du récepteur mais nous n'avons pas tenu compte des fautes sur le lien radio qui entraîneraient la perte de paquets entre deux nœuds à portée radio. Ce type d'étude permettrait d'évaluer la performance du protocole en présence de fautes, et de l'améliorer en lui rajoutant des mécanismes de tolérance aux fautes.

Une autre perspective de ce travail serait de considérer les applications à deux dimensions et donc de rajouter une couche protocolaire de routage temps-réel au dessus de notre protocole.

Enfin, il serait intéressant d'évaluer les performances de notre proposition avec des protocoles MAC déjà existants. Ainsi, une étude de ceux-ci pourrait apporter la preuve que ces derniers n'offrent pas de garanties temps-réel. Une comparaison de performance sur temps moyen pourrait néanmoins être faite pour situer notre protocole.

Bibliographie

[1] T. F. Abdelzaher, S. Prabh, et R. Kiran. *On real-time capacity limits of multi-hop wireless sensor networks.* IEEE Real-Time Systems Symposium (RTSS'04), Lisbonne, Portugal, décembre 2004.

[2] T. F. Abdelzaher, J. A. Stankovic, S. Son, B. Blum, T. He, et A. Wood. *A communication architecture and programming abstractions for real-time embedded sensor networks.* 23rd International Conference on Distributed Computing Systems Workshops (ICDCS 2003 Workshops), mai 2003.

[3] K. Akkaya et M. Younis. *Relocation of gateway for enhanced timeliness in wireless sensor networks.* IEEE Workshop on Energy-Efficient Wireless Communications and Networks (EWCN'04), avril 2004.

[4] I.F. Akyildiz et I.H. Kasimoglu. *Wireless sensor and actor networks: research challenges.* 1st IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS'04), décembre 2004.

[5] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, et E. Cayirci. *Wireless sensor networks: a survey.* Computer Networks (Elsevier) Journal, vol. 38, no. 4, pp. 393-422, mars 2002.

[6] R. Alur, C. Courcoubetis et D.L. Dill. *Model-checking in dense real-time.* Journal of Information and Computation, 104(1):2-34, 1993.

[7] R. Alur et D. L. Dill. *Automata for modeling real-time systems.* Proceedings of the 7th International Colloquium on Aotomata, Lamguages and Programming, pp. 321-335, 1990.

[8] J. Bergson et Wang Yi. *Timed Automata: Semantics, Algorithms and Tools.* Dans W. Reisig et G. Rozenberg, editors, In Lecture Notes on Concurrency and Petri Nets, Lecture Notes in Computer Science vol. 3098, Springer-Verlag, 2004.

[9] P. Boone. *Real-time communication and coordination in wireless embedded sensor networks.* Avril 2004.

[10] M. Bozga, J.C. Fernandez, L. Chirvu, S. Graf, J.P. Krimm, L. Mounier et J. Sifakis. *IF : An Intermediate Representation for SDL an dits Applications.* Proceedings of SDL-FORUM'99, 1999.

[11] M. Caccamo et L. Y. Zhang. *The capacity of implicit EDF in wireless sensor networks.* IEEE 15th Euromicro Conference on Real-Time Systems (ECRTS'03), 2003.

[12] M. Caccamo, L. Y. Zhang, L. Sha, et G. Buttazzo. *An implicit prioritized access protocol for wireless sensor networks.* IEEE Real-Time System Symposium (RTSS'02), décembre 2002.

[13] A. Chandra, V. Gummalla, et J. O. Limb. *Wireless medium access control protocols.* IEEE Communications Surveys, 2000.

[14] J. Chen, Y. Guan, et U. Pooch. *Customizing GPSR for wireless sensor networks.* 1st IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS'04), oct. 2004.

[15] http://www.xbow.com/Products/Wireless_Sensor_Networks.htm Crossbow, Inc., Berkeley .Mote. Dernière consultation le 06/06/2005.

[16] O. Dousse, P. Mannersalo, et P. Thiran. *Latency of wireless sensor networks with uncoordinated power saving mechanisms.* Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2004), pp. 109-120, 2004.

[17] T. Facchinetti, L. Almeida, G. Buttazzo, et C. Marchini. *Real-time resource reservation protocol for wireless mobile ad-hoc networks.* Proceedings of the 25th IEEE Real-Time Systems Symposium (RTSS 2004), pp. 382-391, décembre 2004.

[18] T. Facchinetti et G. Buttazzo. *Integrated wireless communication protocol for ad-hoc mobile networks.* Proceedings of the 2004 International Workshop on Real-Time Networks (RTN 2004), pp. 43-46, juin 2004.

[19] T. Facchinetti, G. Buttazo, M. Caccamo, et L. Almeida. *Wireless real-time communication protocol for cooperating mobile units.* 15th Euromicro Conference on Real-Time Systems, juillet 2003.

[20] E. Felemban, G. Lee, E. Akici, R. Boder, et S. Vural. *Probabilistic QoS guarantee in reliability and timeliness domains in wireless sensor networks.* Proceedings of IEEE Infocom 2005, mars 2005.

[21] S. Giannechini, M. Caccamo, et C. Shih. *Collaborative resource allocation in wireless sensor networks.* Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS '04), pp. 35-44, juin 2004.

[22] K. Godary. *Validation temporelle de réseaux embarqués critiques et fiables pour l'automobile.* Thèse de l'INSA de Lyon, France, novembre 2004.

[23] K. Godary, I. Augé-Blum, et A. Mignotte. *SDL and Timed Petri Nets Versus UPPAAL for the validation of embedded architecture in automotive.* Forum on Specification and Design Languages (FDL'04), septembre 2004.

[24] T. He, J. A. Stankovic, C. Lu , et T. F. Abdelzaher. *SPEED: a stateless protocol for real-time communication in sensor networks.* International Conference on Distribued Computing Systems (ICDCF'03), mai 2003.

[25] T. Herman et S. Tixeuil. *Un algorithme TDMA réparti pour les réseaux de capteurs.* 6ème Rencontres Francophones sur les aspects Algorithmiques des Télécommunications (AlgoTel 2004), mai 2004.

[26] IEEE 802.11, 1999 Edition (ISO/IEC 802-11:1999)

[27] ITU (International Telecommunication Union). *Specification and Description Language – SDL.* Recommendation z. 100 edition, 2000

[28] Q. Jiang et D. Manivannan. *Routing protocols for sensor networks.* Proceedings of the 1st IEEE Consumer Communications and Networking Conference (CCNC 2004), pp. 93-98, janvier [2004].

[29] B. Karp et H.T. Kung. *GPSR: Greedy Perimeter Stateless Routing for wireless networks.* Proceedings of the 6th Annual International ACM Conference on Mobile Computing and Networking (Mobicom 2000), pp. 243-254, août 2000.

[30] G. Larsen, P. Petterson, et W. Yi. *UPPAAL in a nutshell.* Int. Journal on Software Tools for Technology Transfer, 1(1):134-152, décembre 1997.

[31] H. Li, P. Shenoy, et K. Ramamritham. *Scheduling messages with deadlines in multi-hop real-time sensor networks.* IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2005), mars 2005.

[32] H. Li, P. Shenoy, et K. Ramamritham. *Scheduling communication in real-time sensor applications.* Proceedings of 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2004), pp. 10-18, mai 2004.

[33] R. Lin, Z. Wang, et Y. Sun. *Wireless sensor networks solutions for real time monitoring of nuclear power plant.* Proceegins of the 5th World Congress on Intelligent Control and Automation, juin 2004.

[34] X. Liu, Q. Wang, L. Sha, et W. He. *Optimal QoS sampling frequency assignment for real-time wireless sensor networks.* Proceedings of the 24th IEEE International Real-Time Systems Symposium (RTSS 2003), pp. 308-320, décembre 2003.

[35] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, et T. He. *RAP: A real-time communication architecture for large-scale wireless sensor networks.* IEEE Real-Time Technology and Application Symposium (RTAS'02), septembre 2002.

[36] A. Mahapatra, K. Anand, et D. P. Agrawal. *QoS and energy aware routing for real-time traffic in wireless sensor networks.* Special issue of the journal of Computer Communications on Sensor Networks, décembre 2004.

[37] G. Orfanos, J. Habetha, et L. Liu. *MC-CDMA based IEEE 802.11 wireless LAN.* Proceedings of the 12th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2004), pp. 400-405, octobre 2004.
[38] V. Rajavavivarme, Y. Yang, et T. Yang. *An overview of wireless sensor network and applications.* Proceedings of eth 35th Southeastern Symposium on System Theory, 2003.
[39] C. Ramchandani. *Analysis of asynchronous concurrent systems by Timed Petri Nets.* PhD thesis, MIT Cambridge, MA, United Knogdom, 1974.
[40] M. Ringwald et R. Römer. *BitMAC: a deterministic, collision-free, and robust MAC protocol for sensor networks.* 2nd European Workshop on Wireless Sensor Networks, janvier 2005.
[41] J.P. Sheu, C.H. Liu, S.L. Wu, et Y.C. Tseng. *A priority MAC protocol to support real-time traffic in ad-hoc networks.* ACM Wireless Networks, vol. 10, pp. 61-69, janvier 2004.
[42] http://www.smarteksys.com/ SmarTek Systems Homepage. Dernière consultation le 06/06/2005.
[43] http://www-robotics.usc.edu/~behar/SKIT.html Sub-Kilogram Intelligent Tele-Robots (SKIT) Home Page. Dernière consultation le 06/06/2005.
[44] S. R. Saunders. *Antennas and propagation for wireless communication systems.* Wiley, 1999.
[45] F. Sivrikaya et B. Yener. *Time synchronization in sensor networks: a survey.* IEE Network, juillet-août 2004.
[46] http://robotics.eecs.berkeley.edu/~pister/SmartDust/ Smart Dust Research Project Homepage. Dernière consultation le 06/06/2005.
[47] J. A. Stankovic. *Research challenges for wireless sensor networks.* SIGBED Review: Special Issue on Embedded Sensor Networks and Wireless Computing, 1(2), juillet 2004.
[48] J. A. Stankovic, T. F. Abdelzaher, C. Lu, L. Sha, et J. C. Hou. *Real-time communication and coordination in embedded sensor networks.* Proceedings of the IEEE, juillet 2003.
[49] A. Tanenbaum. *Computer Networks.* Fourth edition, Prentice Hall, 2003.
[50] http://robots.stanford.edu/ Sebastian Thrun's Homepage. Dernière consultation le 06/06/2005.
[51] http://www.uppaal.com/ UPPAAL Home Page.
[52] Z. Yang, M. Dong, L. Tong, et B. M. Sadler. *On the MAC for optimal information retrieval pattern in sensor networks with mobile access.* MILCOM 2004, octobre 2004.
[53] W. Ye, J. Heidemann et D. Estrin. *An energy-efficient MAC protocol for wireless sensor networks.* Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM 2002), pp. 1567-1576, vol. 3, juin 2002.
[54] Q. Ye, Y. Zhang, et L. Cheng. *A study on the optimal time synchronization accuracy in wireless sensor networks.* A paraître dans Computer Networks, vol. 48, no. 4, pp 549-566, juillet 2005.
[55] Q. Zhao et L. Tong. *Distributed opportunistic transmission for wireless sensor networks.* Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP 2004), mai 2004.

Annexes

1. Soumission à MASCOTS'05

Le travail présenté dans ce rapport a fait l'objet d'une soumission au « *13th annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'05)* » qui se tiendra à Atlanta aux Etats-Unis, du 26 au 29 septembre 2005. L'article de 8 pages peut être trouvé à l'adresse http://fex.insa-lyon.fr/get?k=xVXW6zRXAdSzvH2ijbX.

Title:
« Proposition of a Hard Real-Time MAC Protocol for Wireless Sensor Networks ».
Thomas Watteyne and Isabelle Augé-Blum

Abstract. *Many wireless sensor network applications are emerging nowadays. As an example throughout this paper, we use a highway car accident monitoring system. When a sensor node detects an accident, it generates an alarm and reports it to a sink node in a multi-hop way. For the sink to receive the alarms and take the appropriate actions on time, the network needs to provide bounded transmission delays. As a consequence, hard real-time guarantees need to be given by wireless sensor network communication protocols. In this paper, for our study case, we propose a new hard real-time MAC protocol, and we give the time constraints that can be reached.*

2. Lexique

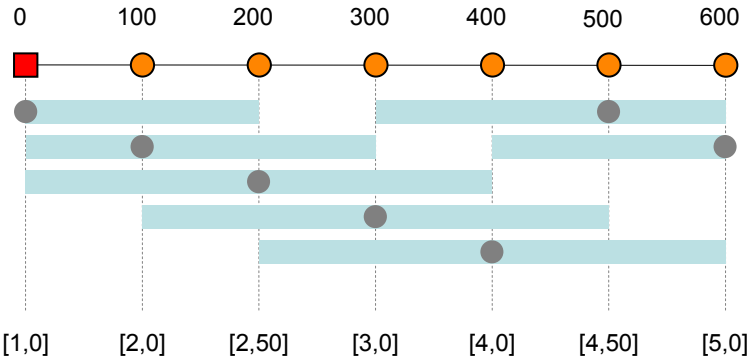
Nous présentons ici les différentes variables utilisées tout au long de ce document, ainsi que leur signification.

A	position absolue du noeud
ACK_EXP	message utilisé pour signaler la fin de la phase de protection
$A_{émetteur}$	position absolue du nœud émetteur
$backoff_{émission}$	backoff utilisé pour élire le nœud relais
$backoff_{initialisation}$	backoff utilisé dans la phase d'initialisation
$backoff_{non\text{-}protégé}$	backoff utilisé en mode non protégé
$backoff_{synchronisation}$	backoff utilisé pendant la synchronisation
BW	bande passante offerte par l'interface radio, en bits par seconde
$CREATION$	message de signalisation créant une nouvelle cellule
$DATA$	contenu du message d'alarme
$dist_{min}$	la distance minimale entre deux nœuds voisins
end	signal pour signaler la fin d'une émission (modélisation UPPAAL)
END_INIT	message de signalisation signalant la fin de l'initialisation
I	le numéro de la cellule à laquelle appartient un nœud
JAM	message de bourrage pour basculer entre modes de fonctionnement
$longueur_{réseau}$	longueur du réseau, en mètres
max_{range}	portée maximale d'un nœud dans les cas les plus favorables (en mètres)
N	le numéro du nœud dans le réseau
new	signal déclenchant la génération d'une nouvelle alarme (UPPAAL)
$nombre_{cellules}$	le nombre de cellules dans le réseau
$nombre_{noeuds}$	le nombre de nœuds dans le réseau
R	la position relative du nœud dans sa cellule
$RESERVE$	état d'une cellule, aucun nœud ne peut générer de nouvelles alarmes
$SILENCE$	message pour réserver une cellule
start	signal pour signaler le début d'une émission (modélisation UPPAAL)
$SYNC$	message de synchronisation
T	la période de passage de deux vagues successives
$taille_{<x>}$	la taille du message de type <x>
$T_{détection}$	la durée mise par l'interface radio pour détecter un message
$Timer_{dernier}$	timer pour qu'un nœud détecte qu'il est le dernier
$Timer_{faille}$	timer pour qu'un nœud détecte une interruption d'initialisation
$T_{propagation}$	temps de propagation
$T_{retournement}$	temps de retournement de l'interface radio
$WCET_{initialisation}$	pire temps d'exécution de la phase d'initialisation
$WCET_{jam}$	pire temps d'exécution du basculement
$WCTT_{non\text{-}protégé}$	pire temps de transmission en mode non-protégé
$WCTT_{protégé}$	pire temps de transmission en mode protégé
$W_{émission}$	vitesse de la vague d'émission
$W_{initialisation}$	vitesse de la vague d'initialisation
$W_{synchronisation}$	vitesse de la vague de synchronisation

3. Application numérique

Afin de donner une idée plus précise de la vitesse des remontées d'alarme, mais aussi de comparer la différence entre ces vitesses selon le mode de fonctionnement (protégé ou non-protégé), nous avons réalisé une application numérique. Nous utilisons l'exemple de la surveillance d'autoroute et disposant un capteur tous les 100 mètres, tout au long des 600 mètres de la portion d'autoroute surveillée. Nous supposons une interface standard de *Berkeley Mote* offrant une bande passante de 19,2 kbps. La taille de tous les messages de signalisation est fixée à 10 bits, celui de l'alarme à 20 bits. Le temps de propagation est celui de la lumière (3.10^8 m/s). Le temps de retournement est supposé négligeable.

Le carré représente le puits, les cercles les nœuds. Les positions absolues sont indiquées au dessus. Les portées des signaux de chaque nœud ou puits est indiqué par les barres horizontales. +découpage en cellule



secondes							
	Non-protégé	0,002	0,002	0,004	0,004	0,006	0,006
	Protégé	1,403	1,403	3,203	5,603	5,603	8,603

Grâce au bon paramétrage de max_{range} (il est ici égal à la portée effective), les temps de remontée d'alarme dans le cas non-protégé sont optimaux. Les vitesses de dégrade en basculant en mode protégé où un alarme met 8,6 secondes à remonter 600 mètres. Les vitesses moyennes de remontée d'alarmes ne peuvent être obtenue qu'en connaissant la part de temps où le réseau fonctionnerait dans les deux modes. Cette part dépend de la charge du réseau, donc du modèle de génération des alarmes.