International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

http://ijpsat.ijsht-journals.org
Vol. 25 No. 2 March 2021, pp. 275-287

# *Vers Un Protocole TDMA Multi-Débit A Diversité Spatiale POUR Les Réseaux De Capteurs Et D'actionneurs Sans Fil : Modélisation Et Vérification*

ANDRIANANTENAINA Zo Ambinintsoa, RAKOTOMIRAHO Soloniaina, RANDRIAMAROSON Rivo Mahandrisoa,

SE-I-MSDE, ED-STII, Université d'Antananarivo

Antananarivo, Madagascar



**Abstract— At the core of many today's industrial applications are the wireless sensor networks (WSNs) and wireless sensor and actuator networks (WSANs). WSNs and WSANs have led to the development of industrial wireless sensor networks (IWSNs) and industrial wireless sensor and actuator networks (IWSANs). These networks play a central role of connecting machines, parts, products, and humans and create a diverse set of new applications to support intelligent and autonomous decision making. For each application, communication protocols should be tailored to comply all requirements specific to that application. Requirements could be specified in terms of end-to-end delay, throughput flexibility, etc. In order to achieve compliance with such requirement, choices should be made during the design process regarding MAC access and routing algorithm. We propose in this paper a multi-rate TDMA protocol with spatial re-use. The protocol proposed in this work is designed to fit heterogeneous networks where different types of sensor and actuators coexist. We give first a formal model of the protocol using timed automata paradigm and then we carry out a formal verification of basic properties upon the model with the UPPAAL model-checker.**

**Keywords— wireless sensor and actuator networks, wireless sensor networks, spatial TDMA, multi-rate TDMA, formal modeling, communication protocol.**

## I. INTRODUCTION

Un réseau de capteurs et d'actionneurs (WSAN : Wireless Sensor Actuator Network) sans fil est une évolution des réseaux de capteurs sans fil (WSN : Wireless Sensor Network). Un réseau WSAN consiste en un ensemble de capteurs et d'actionneurs communiquant par onde radio. Les WSANs se retrouvent actuellement dans plusieurs domaines de l'industrie [1]. Ils ont été intégrés avec succès dans divers systèmes de contrôle-commande [2]. Les applications dans lesquelles un WSAN est mis en œuvre peuvent avoir différentes exigences en termes de délai, de débit, etc. Chaque nœud et chaque actionneur peut requérir par exemple un débit différent selon les dynamiques des types de données ou de commandes à un instant donné. Un protocole de communication adapté doit être mis en place afin de tenir compte de ces spécificités. Nous proposons dans cet article un protocole de communication multi-débit basé sur un contrôle d'accès au medium TDMA. Nous entendons par « multi-débit » le fait que pour un nœud le débit qui lui est alloué n'est pas figé dans le temps. Ce débit lui sera alloué à la demande selon la quantité de données à transmettre et le délai de bout en bout requis pour l'acheminement de ces données jusqu'à destination. Aussi, le protocole proposé minimise le délai d'acheminement des données en exploitant la diversité spatiale, c'est-à-dire que deux nœuds peuvent émettre simultanément tant que leurs transmissions ne sont pas en conflits.

Cet article est organisé en sept sections. Dans la section suivante, nous donnerons d'abord quelques définitions sur les notions

de graphes et quelques terminologies utilisées dans cet article. Ensuite aux sections 3 et 4, nous donnerons une représentation mathématique des modèles d'un réseau WSAN et des transmissions et, par la suite, nous poserons le problème de l'ordonnancement TDMA. Nous développons notre contribution à la section 5. A la section 6, nous procèderons à une modélisation formelle et une vérification du protocole proposé. Nous conclurons à la section 7.

## II. QUELQUES DEFINITIONS

**Définition 1 (Graphe)** Un graphe $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ est constitué de deux entités : V un ensemble fini non vide d'éléments, les éléments de V sont appelés sommets ou nœuds, et E un ensemble tel que $E \subset V \times V$, les éléments de E sont appelés arêtes ou liens. Pour un graphe G, on note $n = |V|$ le nombre de sommets du graphe et $m = |E|$ le nombre d'arêtes du graphe.

**Définition 2 (Chemin)** Soit une paire de sommets $u, v \in V$. Un chemin de *u* à *v*, noté *P(u,v)*, est une séquence de sommets $< u, v_0, \dots, v_k, v >$ tel que $\forall i, 0 \leq i < k$, il existe une arête entre les sommets $v_i$ et $v_{i+1}$ dans G.

**Définition 3 (Cycle)** Un cycle *C* est un chemin $P(u, v)$ tel que $u = v$

**Définition 4 (Voisin d'un nœud)** Soit un nœud $v \in V$. L'ensemble des nœuds *u* tel que $\{u, v\} \in E$ sont appelés voisins de *v*.

**Définition 5 (Degré d'interférence d'un nœud)** Soit un nœud $v \in V$. Le degré d'interférence de *v*, noté *d(v)* est le nombre de ses voisins *u*.

**Définition 6 (Arbre)** Soit $T = (V, E)$ un graphe. Les définitions suivantes sont équivalentes à la définition d'un arbre :

- T est connexe et sans cycle ;
- Il existe un seul et unique chemin entre toutes paires de sommets $u, v \in V$ ;
- T est connexe, et T n'est plus connexe si une arête de T est supprimée ;
- T est sans cycle, et on crée un cycle et un seul cycle en ajoutant une arête entre deux sommets non- adjacents dans T ;
- T est connexe et contient *n - 1* arêtes.

**Définition 7 (Parent)** Soit $T = (V_T, E_T)$ un arbre enraciné au sommet $r \in V_T$, le sommet $u \in V_T$ est le parent du sommet $v \in V_T$ si *u* est l'unique voisin de *v* sur le chemin entre *v* et la racine *r* dans l'arbre *T*.

**Définition 8 (Enfants)** Soit $T = (V_T, E_T)$ un arbre enraciné au sommet $r \in V_T$, tous les sommets voisins de $v \in V_T$ qui ne sont pas parent de *v* sont des enfants dans T.

**Définition 9 (Slot)** En TDMA, le temps est subdivisé en slot. Chaque nœud émet seulement pendant le slot de temps qui lui est alloué.

**Définition 10 (Trame)** En TDMA, nous appellerons trame un ensemble de slots qui se répète périodiquement au fil des transmissions.

## III. REPRESENTATION DU RESEAU ET DES TRANSMISSIONS

La topologie d'un réseau TDMA peut être modélisée par un graphe. Deux nœuds de ce graphe sont connectés s'ils sont à portée radio l'un de l'autre. Dans ce qui suit, nous supposons que les liens sont symétriques. Le réseau TDMA peut alors être représenté par le graphe de connectivité $G(V, E, f_t)$, où :

- $V = \{v_1, \dots, v_n\}$ est l'ensemble des nœuds du graphe,
- $E = \{e_1, \dots, e_m\}$ constitue un ensemble dont les éléments sont des liens directionnels entre nœuds voisins et
- $f_t : E \to V \times V$ est une relation assignant des liens à des pairs de nœuds.

L'expression $e_k \in E, f_t(e_k) = (v_i, v_j)$ signifie qu'il y un lien entre les nœuds $v_i$ et $v_j$ et que la transmission se fait de $v_i$ vers $v_j$.

Nous associons un débit binaire à chaque lien. Ce débit binaire dépend du type de modulation utilisé, laquelle est choisie en fonction du rapport signal à bruit du lien. Nous définissons le débit binaire d'un lien comme le nombre de bits transmis pendant un slot TDMA. Nous pouvons alors écrire la relation $b : E \to \{B_1, B_2, \dots, B_{max}\}$ où $B_1$ est le nombre de bits dans un slot pour une

modulation minimale et $B_{max}$ correspond au nombre de bits par slot avec une modulation maximale.

Pour un nœud source $v_i$, nous noterons $g_i^{\phi} > 0$ le nombre de bits entrant via la connexion $\phi$ durant une trame TDMA. Et pour un nœud de destination $v_j$, $g_i^{\phi} < 0$ correspond au nombre de bits sortant via la connexion $\phi$ durant une trame TDMA.

La bande passante d'un lien est définie comme étant le nombre de bits transmis pendant une trame. Etant partagée par les différentes connexions du lien, et tenant compte du principe de la conservation des flux, nous avons :

$$\sum_{e_j \in \{v_i\}^+} f_j^{\phi} = g_i^{\phi} + \sum_{e_k \in \{v_i\}^-} f_k^{\phi} \quad (1)$$

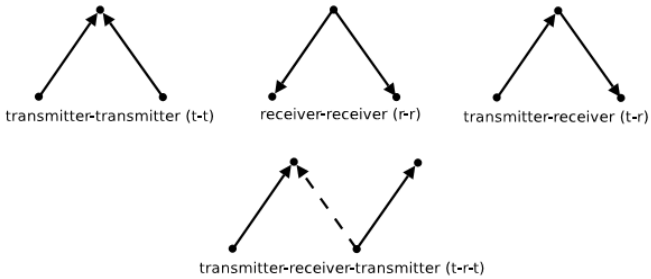
Dans cette équation, $f_j^{\phi}$ est la bande passante allouée à la connexion $\phi$ sur le lien $e_j$, $\{v_i\}^+$ et $\{v_i\}^-$ représentent respectivement l'ensemble des liens sortant et entrant du nœud $v_i$.

En supposant qu'à chaque connexion d'un lien a été allouée une bande passante, le nombre de slot du lien dans une trame peut s'écrire :

$$d_j = \left\lceil \frac{\sum_{\phi} f_j^{\phi}}{b_j} \right\rceil = \left\lceil \frac{f_j}{b_j} \right\rceil \quad (2)$$

où $f_j = \sum_{\phi} f_j^{\phi}$ est le nombre total des bits transmis par toutes les connexions utilisant le lien, $b_j$ représente la modulation utilisée et $d_j$ est le nombre de slots occupés par le lien $e_j$ dans une trame.

Un lien peut interférer avec d'autres liens. On dira aussi dans la suite de cet article que les transmissions sur ces liens sont en




**Légende** :

- *t-t* : Deux transmissions de deux émetteurs à destination d'un même récepteur se brouilleront
- *r-r* : Un seul récepteur ne peut pas séparer dans une seule transmission les trafics à destination de deux récepteurs différents
- *t-r* : Un nœud ne peut pas émettre et recevoir en même temps
- *t-r-t* : Deux transmissions de deux émetteurs à destination de deux récepteurs se brouilleront si un des récepteurs est connecté aux deux émetteurs

Fig. 1. Les conflits dans un réseau TDMA

conflits. Deux transmissions sont en conflits [3] si l'un des quatre cas illustrés par la figure **Fig.1** se présente.

Les conflits peuvent être modélisés par un graphe de conflit. Ce graphe peut être défini comme un triplet $G_c(E, C, f_c)$ où

- $E$ désigne l'ensemble des liens,
- $C = \{c_1, \ldots, c_r\}$ est l'ensemble des conflits TDMA dont chaque élément représente un des $r$ conflits entre un pair de liens,
- $f_c : C \rightarrow \{\{e_i, e_j\}, e_i, e_j \in E\}$ est une relation associant un conflit à un pair de lien.

## IV. FORMULATION DU PROBLEME DE L'ORDONNACEMENT TDMA

Un ordonnancement TDMA assigne un lien à chaque slot d'une trame. Etant donné que cet ordonnancement se répète à chaque trame, il est suffisant de représenter cette assignation par la relation $I : E \times M \rightarrow \{0,1\}$ dans laquelle $E$ est l'ensemble des liens et $M = \{0, \dots, T-1\}$ représente l'index des slots de la trame [4]. De cette formulation, nous disons qu'un lien $e_i$ est programmer à transmettre dans le slot $t$ si $I(e_i, t) = 1$. Et tenant compte de la périodicité de l'ordonnancement, nous pouvons encore dire que $e_j$ est actif dans tous les slots éléments de l'ensemble $\{t + z_i T,\ z_i \in \mathbb{Z}\}$.

L'ordonnancement est dit valide si à chaque lien a été alloué le nombre slots nécessaire pour la transmission de ses données :

$$\sum_{t=0}^{T-1} I(e_i, t) = d_i\ , \forall e_i\ \in E \tag{3}$$

où $d_i$ est le nombre de slot défini dans l'équation (2).

L'ordonnancement est dit sans conflit si les liens en conflit ne sont pas programmés simultanément pour transmettre :

$$I(e_i, t) + I(e_j, t) \leq 1, \forall t \in M, \forall c_k \in C \tag{4}$$

où $f_c(c_k) = \{e_i, e_j\}$. En général, le problème de l'ordonnancement TDMA est de trouver un ordonnancement **valide** et **sans conflit**. Nous pouvons représenter un ordonnancement TDMA par le pair de relations $S(\pi, d)$ où $\pi = [\pi_1, \dots, \pi_m]^T$ est un vecteur représentant l'instant d'activation de chaque lien et $d = [d_1, \dots, d_m]^T$ un vecteur dont les éléments sont les durées de transmissions des liens formulées par l'équation (2). Le vecteur $\pi$ est positif et est inférieur à $T$, c'est-à-dire :

$$\pi\ \in \mathbb{Z}^m_{[0,T]}$$

L'ordonnancement $S(\pi, d)$ correspond à une matrice d'ordonnancement $I_S$ dont chaque ligne correspond à un lien $e_i$. Sur cette ligne, les $\pi_i$ premières colonnes sont mises à 0, et les colonnes $\{\pi_i, \dots, \pi_i + d_i - 1\}$ sont à 1.

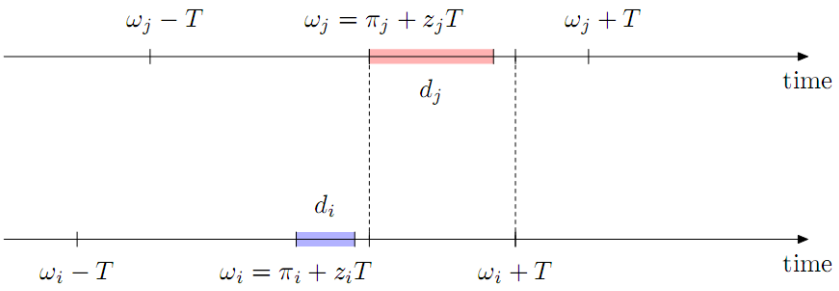



Fig. 2. Instant d'activation de deux liens $e_i$ et $e_j$

Du fait de la périodicité des trames, les instants d'activation $\pi_i$ du lien $e_i$forment une série $\Pi_i = \{\pi_i + z_i T, z_i \in \mathbb{Z}\}$ (Fig-2). Pour tout $\omega_i \in \Pi_i$, on aura :

$$\pi_i = \omega_i\ (mod\ T) \tag{5}$$

L'ordonnancement défini par $S(\pi, d)$ est valide si à chaque lien $e_i$ est alloué $d_i$colonnes dans la matrice $I_S$. $S(\pi, d)$ est sans conflit pour deux liens $e_i$ et $e_j$ conflictuels si aucune transmission simultanée n'a lieu dans une même trame pour tout $\omega_i \in \Pi_i$ et $\omega_j \in \Pi_j$.Considérant que $e_j$ ne doit pas transmettre tant que $e_i$ n'ait pas fini de transmettre, on peut écrire $\omega_j = \min\{\omega \in \Pi_j : \omega \geq \omega_i\}$ (**Fig. 2**). Aussi :

$$\omega_j \geq \omega_i + d_i \Leftrightarrow \omega_j - \omega_i \geq d_i \tag{6}$$

Considérant que $e_j$ doit arrêter de transmettre avant que $e_i$ ne commence à transmettre, on peut écrire :

$$\omega_j + d_j \leq \omega_i + T \Leftrightarrow \omega_j - \omega_i \leq T - d_j \quad (7)$$

En combinant les équations (6) et (7), nous déduisons que pour un ordonnancement sans conflit, les conditions suivantes doivent être remplies :

$$d_i \leq \omega_j - \omega_i \leq T - d_j \; et \; 0 \leq \omega_j - \omega_i \leq T \quad (8)$$

Dans [5], il a été démontré que le problème de la recherché d'un ordonnancement satisfaisant (8) est NP-complet [6]. Dans la section suivante, nous proposons une heuristique permettant une résolution du problème d'ordonnancement TDMA.

## V. CONTRIBUTION

Dans cette section, nous proposons une démarche permettant de trouver un ordonnancement TDMA valide et sans conflit pour un réseau de capteurs sans fil. La demarche proposée est subdivisée en trois étapes :

- Construction d'un arbre couvrant
- Construction d'un plan d'acheminement
- Allocation des slots de la trame

Du point de vue d'un réseau de capteurs sans fil, il est important de noter que la démarche que nous proposons prend en charge également l'acheminement d'un paquet d'un nœud source à un nœud de destination, c'est-à-dire le routage des paquets. Les algorithmes décrits dans cette section sont exécutés par le puits. Une fois un ordonnancement valide et sans conflit calculé, le puits transmet à chaque nœud la table d'allocation des slots.

Afin d'illustrer notre démarche, nous considérerons la topologie de la figure **Fig. 3**. Les éléments constitutifs du réseau représentés par cette topologie sont les suivants :

- S : le puits
- {a, e}, {b, g}, {c, d}, {f, h} : différents capteurs dont chaque ensemble peut requérir un débit différent

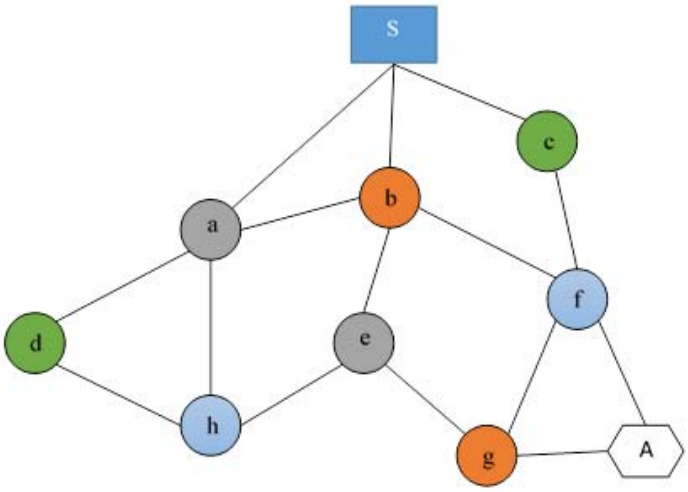



Fig. 3. Graphe de départ

- A : actionneur

### 1.1 Construction d'un arbre couvrant

Etant donné un graphe issu d'une topologie d'un réseau de capteurs et d'actionneurs sans fil donné, nous pouvons obtenir de ce graphe un arbre couvrant par un algorithme de parcours en largeur BFS (breadth-first search). Cet algorithme est composé des étapes suivantes :

1. Déterminer le puits du graphe et le sélectionner comme racine de l'arbre ;
2. Pour un nœud *i* déjà sélectionné et se trouvant à *n* saut de la racine de l'arbre, sélectionner tous les nœuds $c_i$ voisins d'un saut de *i* et tel que $c_i$ se trouvent à n+1 sauts de la racine ;
3. Associer $c_i$ au nœud *i* qui sera dès lors considéré comme parent de $c_i$. Les nœuds $c_i$ seront les enfants du nœud *i*.
4. Répéter les étapes 2 et 3 pour chaque $c_i$ et selon l'ordre de sélection des nœuds $c_i$ dans l'étape 2 jusqu'à ce que tous les nœuds du graphe soient sélectionnés.

Après exécution de l'algorithme BFS, nous obtenons l'arbre couvrant de la figure **Fig. 4**.

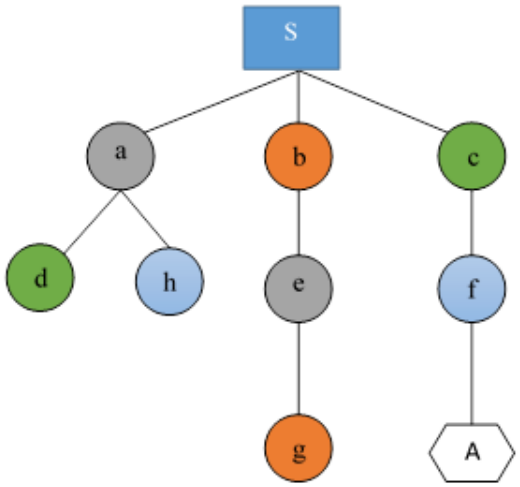



Fig. 4. Arbre couvrant obtenu

### 1.2 Construction d'un plan d'acheminement

Un plan d'acheminement est une séquence ordonnée d'étapes contenant toutes les transmissions nécessaires pour l'acheminement sans conflit des paquets présents dans le réseau. Le plan est construit de la manière suivante :

- A chaque étape, une ou plusieurs transmissions sont assignées
- L'ordre des transmissions respecte l'ordre des sauts dans le routage des paquets. Une transmission d'un nœud parent doit être programmée après les transmissions de ses nœuds fils.

Nous disons qu'une étape est compatible pour un nœud *i* si :

- L'étape n'est assignée à aucun nœud ou,
- Si les nœuds assignés dans l'étape ne sont pas des voisins à un saut ou à deux sauts du nœud *i.*

L'algorithme d'assignation d'étape est la suivante :

1. Sélectionner la racine de l'arbre et lui assigner la première étape ;
2. Sélectionner les nœuds ayant déjà un parent assigné au plan d'acheminement en gardant le même ordre de sélection des parents ;
3. Trier ces nœuds, en ordre décroissant, selon leur degré d'interférence ;
4. Suivant l'ordre du tri, pour chaque nœud, trouver une étape compatible en commençant la recherche par l'étape à laquelle le parent est assigné et en faisant une recherche circulaire. Si aucune étape n'est trouvée, ajouter une étape à la fin du plan d'acheminement et l'assigner au nœud.
5. Répéter à partir de 2. jusqu'à ce que tous les nœuds du réseau aient été assignés à une étape ;
6. Inverser la séquence des étapes.

En appliquant cet algorithme jusqu'à l'étape 5, nous obtenons la table l'assignation intermédiaire du **Tableau 1** pour le graphe de la figure **Fig. 3**.

Tableau 1. Assignation intermédiaire

| | | **Etapes** | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 |
| **Nœud** | S | **S** | b | a | c | | |
| | a | S | b | **a** | | h | d |
| | b | S | **b** | a | e | f | |
| | c | S | | | **c** | f | |
| | d | | | a | | h | **d** |
| | e | | b | | **e** | h | g |
| | f | | b | | c | **f** | g |
| | g | | | | e | f | **g** |
| | h | | | a | e | **h** | d |

Après inversion de la séquence, l'assignation finale est donnée par le **Tableau 2** ci-après.

Tableau 2. Assignation finale

| | | **Etape** | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 |
| **Nœud** | S | | | c | a | b | **S** |
| | a | d | h | | **a** | b | S |
| | b | | f | e | a | **b** | S |
| | c | | f | **c** | | | S |
| | d | **d** | h | | a | | |
| | e | g | h | **e** | | b | |
| | f | g | **f** | c | | b | |
| | g | **g** | f | e | | | |
| | h | d | **h** | e | a | | |

En assignant une étape à chaque nœud de l'arbre couvrant, nous obtenons un arbre de routage. La figure **Fig. 5** représente l'arbre de routage avec les étapes assignées par l'algorithme précédemment développé.

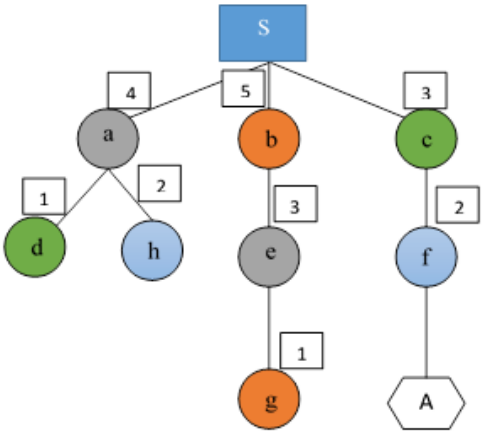



Fig. 5. Représentation de l'arbre de routage avec les étapes du plan d'acheminement

### 1.3 Allocation des slots

La structure de la trame que nous proposons dans ce travail est illustrée par la figure **Fig. 6.**

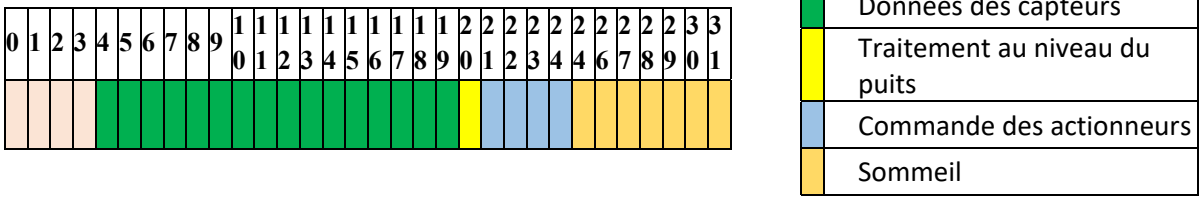



Fig. 6. Structure d'une trame

Les messages de contrôle du réseau sont envoyés dans les slots Contrôle. Ces messages sont diffusés par le puits en broadcast à destination des nœuds. Ce sont ces messages qui contiennent les tables d'allocation de slot que le puits a calculé et qui doivent être disséminés au niveau de chaque nœud. Le nombre de slots Contrôle dépend de la topologie du réseau et de l'ensemble dominant connexe de diffusion qui peut être construit à partir cette topologie. La construction d'un tel ensemble n'est pas l'objet de cet article. Pour simplifier, nous supposons dans la suite qu'à chaque fois que le puits émet un message de contrôle, tous les nœuds reçoivent le message en un slot.

A la suite des slots Contrôle viennent les slots durant lesquels les données des capteurs sont acheminées vers le puits. Un des caractéristiques du protocole que nous proposons dans cet article est l'adaptation du débit selon les besoins de chaque nœud. Dépendant du type de transmission générée par le nœud *i*, on allouera alors à chaque nœud *i* source du réseau $d_i$ slots de temps dans la trame TDMA. Pour un nœud relais *r*, il lui faudra associer

$$d_r + \sum_{j \in C_r} d_j$$

où $d_r$ est le nombre de slots associé au nœud *r* et $C_r$ désigne l'ensemble des nœuds ayant *r* comme parent. Outres les messages

de données, des messages d'alertes peuvent être aussi envoyés dans ces slots. Un message d'alerte est émis par un nœud pour faire savoir au puits :

- La perte d'un nœud voisin
- L'apparition d'une période durant laquelle le débit alloué par le puits au nœud n'arrive plus à supporter la transmission des données disponibles. Le nœud concerné envoie alors une requête d'augmentation du nombre de slots qui lui sont alloués. Le puits recalcule l'ordonnancement et renvoie à tous les nœuds par un message de contrôle la nouvelle table d'allocation de slots.

Un slot est ensuite réservé pour donner au puits le temps de traiter les données reçues des capteurs à la suite duquel les commandes à destination des actionneurs sont envoyées.

Selon le volume du trafic et la durée d'un slot, il se peut que des slots soient inutilisés. Les nœuds entrent en sommeil durant ces slots.

En supposant la configuration du **Tableau 3**, l'allocation qui sera disséminé par le puits à tous les nœuds est donnée par le **Tableau 4**.

Tableau 3. Nombre de slots à allouer à chaque nœud

| **Nœuds** | **Nombre de slots alloué** |
|---|---|
| c – d | 1 |
| f – h | 3 |
| a – e | 1 |
| b – g | 2 |

Tableau 4. Allocation des slots pour la transmission des données des capteurs

| | **Etape** | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| | **Slots** | 1 - 2 | 3 - 5 | 6 - 9 | 10 - 14 | 15 - 19 | |
| **Nœud** | S | | | | | | **S** |
| | a | | | | **1+3+1** | | S |
| | b | | | | | **2+1+2** | S |
| | c | | | **3+1** | | | S |
| | d | **1** | | | | | |
| | e | | | **2+1** | | | |
| | f | | **3** | | | | |
| | g | **2** | | | | | |
| | h | | **3** | | | | |

## VI. MODELISATION ET VERIFICATION

Dans cette section, une modélisation du protocole est faite. Cela permet de nous assurer que le protocole proposé est correct par construction. Pour ce faire, nous avons choisi le *model checker* UPPAAL.

L'outil de *model checking* UPPAAL [7] offre la possibilité d'implémenter le modèle en tant qu'automate. Dans UPPAAL, un automate est défini par ses états (appelés Place), ses transitions, ses invariants, des gardes et des mises à jour. UPPAAL permet de déclarer des variables et tableaux de variables qui sont notamment utiles pour implémenter une représentation de la topologie du réseau. Les données peuvent être locales à un nœud ou bien globales. Il est possible d'écrire des mises à jour sur les données (exécutées lors du passage des transitions) sous forme de fonctions en un langage dérivé du C. Les synchronisations entre transitions

sont réalisées sur des variables appelées canaux déclarés en tant que chan dans UPPAAL.

Dans cette modélisation, nous supposons que la durée d'un slot est de 10 unités de temps. En interprétant une unité de temps comme 1 ms, la durée d'un slot serait alors de 10 ms, ce qui est plutôt courant dans les réseaux de capteurs sans fil.

### 1.4 Modélisation du puits

Le modèle UPPAAL du puits est donné par la figure **Fig. 6**. L'état initial de l'automate est *Start* Le puits initialise le réseau en envoyant en broadcast une synchronisation sur le canal *startup*. La fonction *initialize()* appelée lors de la transition vers l'état *Idle* est utilisée pour définir des valeurs appropriées pour les variables locales et globales. C'est à partir de l'état *Idle* que le modèle effectue les différentes transitions vers les états *Send, Receive, ReceiveAlert, Control* et *Sleep*. La transition choisie dépend du message reçu par le puits et de la variable *currentSlot* et de l'horloge *x*. L'horloge *x* est active lorsque *x'==1*, elle est en pause lorsque *x'==0*.

L'automate accède à l'état Control quand le puits doit diffuser un message de contrôle. L'émission d'un message de contrôle dans le modèle est représenté par l'émission d'une synchronisation sur le canal broadcast *sendControl[change]* dans laquelle *change* est une variable booléenne indiquant à tous les nœuds s'il y a ou non un changement dans l'allocation des slots. *change* est *true* s'il y a une table d'allocation à prendre en compte.

Quand il n'y a aucune opération active (envoi ou réception d'un message) programmée pour le slot courant, l'automate accède à l'état *Sleep*. Une vérification de la valeur de la garde *shouldSleep()* est faite avant d'effectuer la transition vers cet état. Nous utilisons un canal de broadcast urgent *choice* pour forcer le choix de cette transition dès que *shouldSleep()* est vraie. La place *Sleep* a un invariant *x<=currentSlot*10* . La valeur de *currentSlot* est calculée par la fonction *calculateWakeup()*, cette valeur correspond à l'instant où le puits doit se réveiller.

La place *Send* est accédée quand le puits émet des données pendant son slot de temps. La place *Receive* est atteinte quand le puits reçoit des données des nœuds. Un accusé de réception est envoyé par la suite via le canal *ack*.

L'état *ReceiveAlert* est accédé quand le puit reçoit un message d'alerte. Le puits reste dans cet état pendant la durée d'un slot (*x<=currentSlot*10*). La garde *isAlert()* est utilisée pour différencier les messages d'alertes.

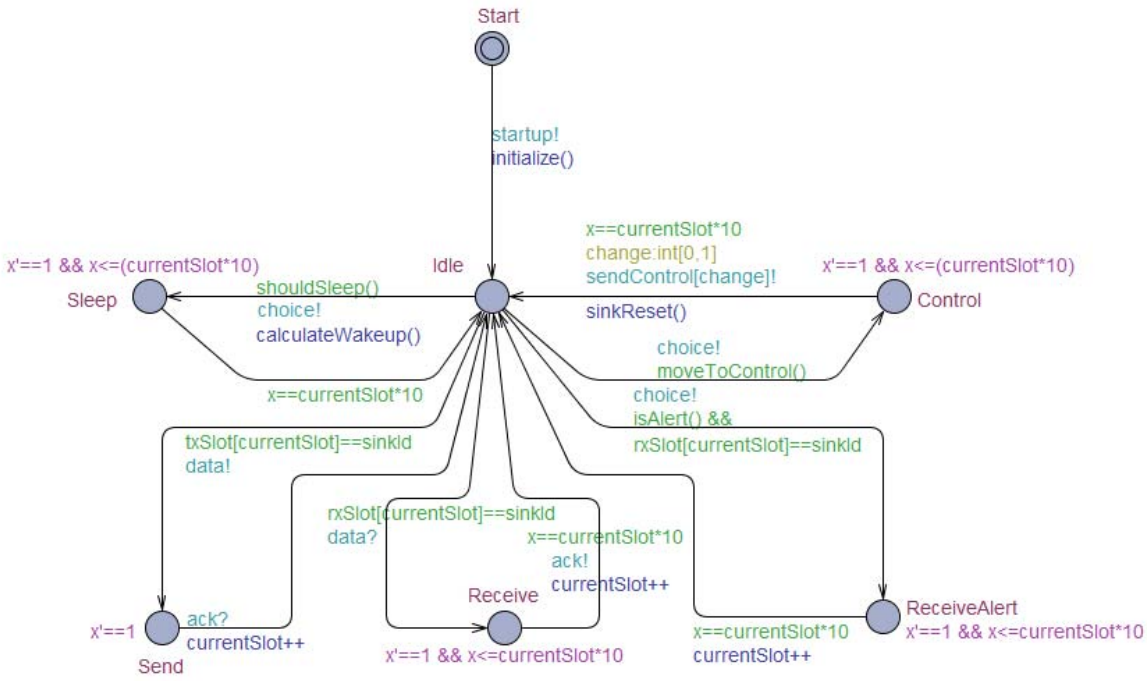



Fig. 6. Modèle UPPAAL du puits

### 1.5 Modélisation des nœuds

Un nœud peut être soit un capteur, soit un actionneur. Le modèle UPPAAL des nœuds est donné par la figure **Fig. 7**. Le modèle est similaire à celui du puits excepté pour les états *Control* et *SendAlert*. Similairement à ce qui a été fait pour le modèle du puits, nous utilisons aussi une variable locale (*y'*) de type *clock* pour nous permettre de pauser (*y'==0*) ou non (*y'==1*) l'horloge principale *y* du modèle.

L'automate accède à l'état SendAlert quand le nœud a un message d'alerte à émettre. Pour rappel, un message d'alerte est émis pour faire savoir au puits :

- la perte d'un nœud voisin
- l'apparition d'une période durant laquelle le débit alloué par le puits au nœud n'arrive plus à supporter la transmission des données disponibles.

### 1.6 Vérification

Nous vérifions certaines propriétés comportementales de base liées au fonctionnement du modèle. Le but est d'obtenir un haut degré de confiance dans le modèle construit. Nous utiliserons le langage de vérification d'UPPAAL qui est un variant du langage TCTL (Timed Computation Tree Logic) [8].

Nous vérifions que les propriétés basiques suivantes sont vraies.

**Communication des données au niveau des nœuds**

$$\forall\, i,j \in nodeId\ st.parent[j] == i : A <> (Node(i).Send \wedge Node(j).Receive$$

**Communication des données au niveau du puits**

$$\forall\, i \in nodeId\ st.parent[i] == sinkId : A <> (Node(i).Send \wedge Sink.Receive$$

**Absence d'état bloquant**

$$A[\ ]\ !\ deadlock$$

**Changement de débit suite à l'émission d'un message d'alerte**

$$\forall\, i \in nodeId : E <> (Node(i).savedAlert \wedge Sink.moveToControl()$$

$$\forall\, i \in nodeId : E <> (Node(i).alertSent\ imply\ Node(i).nodeReset())$$

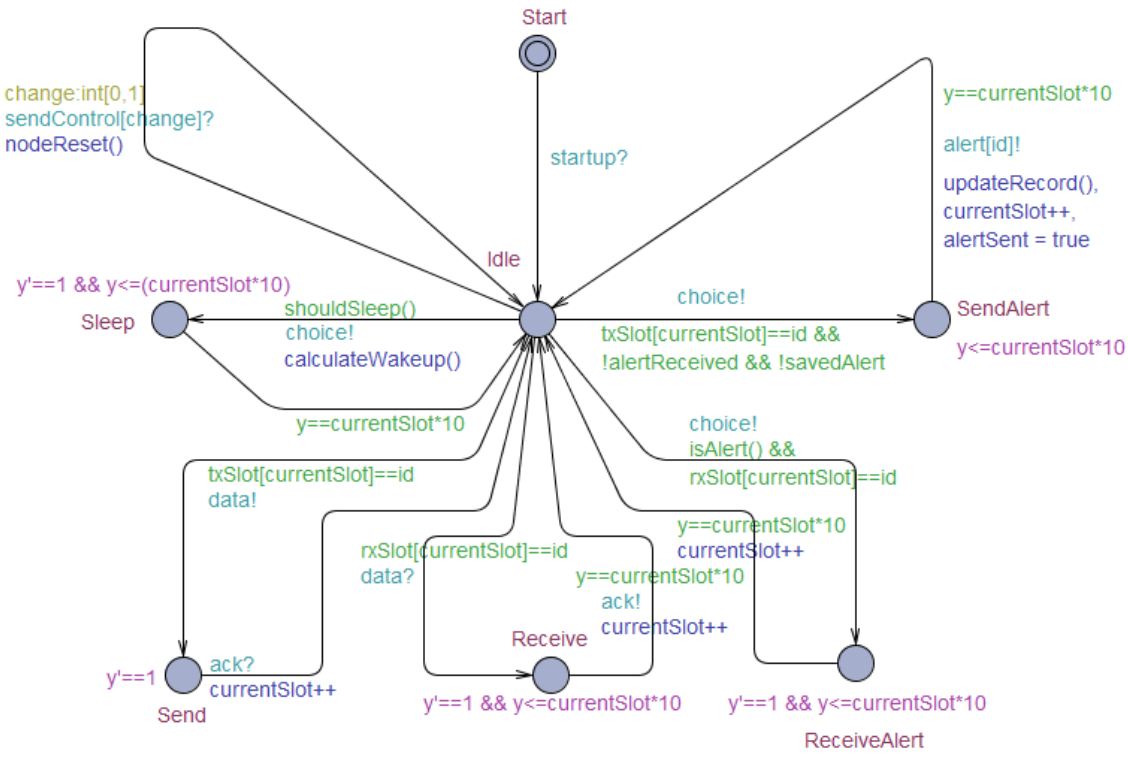



Fig. 7. Modèle UPPAAL des nœuds

## VII. CONCLUSION

Le protocole proposé dans cet article a été conçu pour les réseaux de capteurs et d'actionneurs sans fil hétérogènes. Dépendant

de la performance demandée, chaque nœud émet une demande d'augmentation ou de réduction de débit au puits. Ce dernier recalcule l'ordonnancement TDMA et diffuse ensuite la nouvelle table d'allocation de slots au niveau de chaque nœud. Le protocole que nous avons proposé a été modélisé en tant qu'automate. Les vérifications formelles effectuées sur les modèles nous ont permis de confirmer que toutes les propriétés requises pour un fonctionnement correct du protocole proposé sont satisfaites.

## REFERENCES

[1] J. S. Raza, M. Faheem, M. Guenes. "Industrial wireless sensor and actuator networks in industry 4.0: Exploring requirements, protocols, and challenges — A MAC survey." WILEY Int J Commun Syst 2019, 2019.

[2] P. Park, S. Coleri Ergen, C. Fischione, C. Lu and K. H. Johansson, "Wireless Network Design for Control Systems: A Survey" IEEE Communications Surveys & Tutorials", vol. 20, no. 2, 2018.

[3] B. Djukic, S. Valaee, "Link scheduling for Minimum Delay in Spatial Re-use TDMA," IEEE INFOCOM, Juin 2007.

[4] B. Hajek, G. Sasaki, "Link scheduling in polynomial time," IEEE Trans. Inform. Theory, vol. 34, no. 5, Septembre 1988.

[5] M. A. Odijk, "Railway timetable generation," Ph. D., Technische Universiteit Delft, January 1998.

[6] M. R. Garey, D. S. Johnson, "Computers and Intractability : A guide to the Theory of NP-Completeness", ISBN 0-7167-1044-7, Bell Telephone Laboratories, 1979.

[7] A. D. Gerd Behrmann, K.Larsen, "A tutorial on UPPAAL.", In Formal Methods for the Design of Real-Time Systems, Volume 3185 of Lecture Notes in Computer Science, 2004.

[8] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking for real-time systems. In 5th Symposium on Logic in Computer Science (LICS'90), 1990.