Formal QoS Validation Approach on a Real-Time MAC Protocol for Wireless Sensor Networks

Thomas Watteyne, Isabelle Augé-Blum, Stéphane Ubéda

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Formal QoS Validation Approach on a Real-Time MAC Protocol for Wireless Sensor Networks

Thomas Watteyne — Isabelle Augé-Blum — Stéphane Ubéda

N° 5782

Décembre 2005

Thème COM

rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--5782--FR+ENG

Formal QoS Validation Approach on a Real-Time MAC Protocol for Wireless Sensor Networks

Thomas Watteyne[*] , Isabelle Augé-Blum[*] , Stéphane Ubéda[*]

Thème COM — Systèmes communicants
Projet ARES

Rapport de recherche n° 5782 — Décembre 2005 — 27 pages

Abstract: Several wireless sensor network applications are currently popping up, in various domains. Their goal is often to monitor a geographic area. When a sensor detects a monitored event, it informs a sink node using alarm messages. The area surveillance application needs to react to such an event with a finite, bounded and known delay: these are real-time constraints. The network being linear, routing becomes unnecessary. This work proposes a new real-time MAC protocol with realistic assumptions on sensor networks. We present a formal validation of this protocol, and explicit the worst case times for the services offered by the protocol (initialization and alarm transmission using different modes).

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

[*] CITI Laboratory, INSA Lyon, France.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 —Télécopie +33 4 76 61 52 52

Protocole temps-réel pour réseaux de capteurs: une approche de validation formelle de QdS

Résumé : De nombreuses applications pour réseaux de capteurs sans fils émergent actuellement dans de nombreux domaines. Leur but est souvent de surveiller une zone géographique. Lorsqu'un capteur détecte un évènement redouté, il informe un noeud puits en utilisant des messages d'alarme. L'application de surveillance de la zone géographique doit répondre à cet évènement avec un temps borné et connu à l'avance: ce sont des contraintes temps-réel. Le réseau étant linéaire, le routage devient non nécessaire. Ce travail propose un nouveau protocole MAC temps-réel avec des hypothèses réalistes sur le réseau de capteurs. Nous présentons une validation formelle de ce protocole et explicitons les temps pires pour les services offerts par ce protocole (initialisation et remontée d'alarmes selon différents modes).

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

1 Introduction

Sensor networks are used in many application areas, ranging from Defense and surveillance [1] to Health or intelligent homes [2] [3]. In a surveillance application, a possibly large number of sensors are deployed inside the area that needs to be covered. In a forest fire detection scenario, a fire is detected by one or more nodes, the resulting alarm is sent to a sink node. This node collects all the alarm messages and is able to launch the appropriate action.

A sensor is an embedded component capable of doing three complementary tasks: sensing a physical phenomenon (temperature, noise . . .), processing the sensed value and communicating with other sensors using wireless technologies [4, 5]. It therefore needs to cope with the performance limitations of embedded systems, mainly processing power, memory size and available energy.

Sensor networks are to be deployed rapidly over the area that needs to be covered. Using the example of a volcanic eruption surveillance scenario, a random deployment can be done by dropping the sensors from a helicopter. Due to the dangerous nature of the covered area, human intervention on the sensors can not be count on. This is especially constraining for energy, as battery replacement is not possible [6].

Due to the random nature of the deployment, no fixed infrastructure is present in the network. What's more, even though the sensors do not move, sensor loss due to battery failure for example can result in topological changes. Communication between node and sink is performed in a multi-hop way. Indeed, the node's embedded radio interface is not powerful enough for its signal to reach the sink in one hop, so intermediate nodes are used to relay the message. Ad-hoc networks answer those three constraints; sensor networks are an application domain of ad-hoc networks.

Sensor networks need to address many constraints. Quality-of-Service constraints importance is growing, and among various parameters, guaranteeing a timeliness behavior is considered as a key challenge for research on wireless sensor networks [7]. Indeed, with security related applications (*e.g.* monitoring of forest fires), the system's reaction time needs to be bounded and guaranteed. These real-time constraints depend on the application's nature.

To take the application's real-time constraints into account, not only do the hardware and the software need to provide bounded delays, but so does the underlying communication network [8]. We focus on the communication part of the problem.

For critical applications to function correctly, a formal validation of those guarantees needs to be given. To our knowledge, no formal validation has been presented so far for real-time sensor networks.

As an application class, we consider linear monitored areas. The peculiarity of this application class is that as the monitored area is linear, the underlying network is linear too. This way, as finding a path from source to destination is trivial, we free ourselves from routing considerations. Linear application examples include highway accident detection, train positioning on a railway, or production line monitoring. Network linearity is constraining, but our approach is to work gradually, a two-dimensional extension being considered future work. Even though linearity is assumed, no assumptions are made on the deployment, so random linear deployment is possible.

This paper's goal is to propose a novel MAC protocol and to formally validate both its behavior and its timeliness characteristics

The remainder of the paper is organized as follows: We provide an investigation of current proposals in Section 2. Section 3 presents a detailed description of our novel MAC protocol. The protocol's formal validation is given in Section 4, we obtain the Quality-of-Service parameters of our protocol, especially worst case transmission times. Section 5 concludes this paper and presents work that still needs to be done.

2 Related Work

Two different definitions of real-time are commonly used: hard-real time and soft real-time [9, 10, 11]. A hard real-time communication system needs to guarantee that all messages reach destination before their deadline. The deadline is the end of the validity interval of the sensor data. Worst case times are therefore determined; they need to be known a priori, and to be bounded and guaranteed. Two worst case times are used in communication systems: Worst Case Execution Time ($WCET$) and Worst Case Transmission Time ($WCTT$).

In soft-real time systems, the application is loose enough to accept that a given portion of the sent messages reach destination after their deadline. This portion is commonly expressed in percentage and is called "miss ratio".

2.1 Soft Real-Time Solutions

In a soft-real time sensor network, two flows of messages are present: "normal" messages and so-called "real-time" messages [12]. The goal of the network's communication mechanisms is to deal with real-time messages with a higher priority than normal messages, in order to reduce the miss ratio. Real-time constraints can be taken into account at each OSI layer [13], so different works focus on different layers.

At the MAC layer (part of network layer 2), a differentiation mechanism is needed. Existing MAC protocols can be adapted to support differentiation. In 802.11's MAC protocol (DCF), $backoff$ and IFS timers are used to regulate medium access. By reducing those timers for real-time messages, differentiation is achieved. Besides, nodes can be asked to emit synchronously a signal proportional to the priority of the message they want to transmit. If the node still senses an occupied medium after it has stopped transmitting this signal (*i.e.* another node is still emitting its priority signal), it can not transmit. This way, only the message with the highest priority will be transmitted [14].

RAP [15] presents a communication architecture covering network layers 1, 2 and 3. The overall idea is not to use timeliness proximity of a message's deadline as its urgency metric, but to assign a required velocity to each message. Information velocity is defined as deadline over approximate geographical distance. Based on the message's destination, a required velocity is calculated at the source and attached to the message. Routing decisions

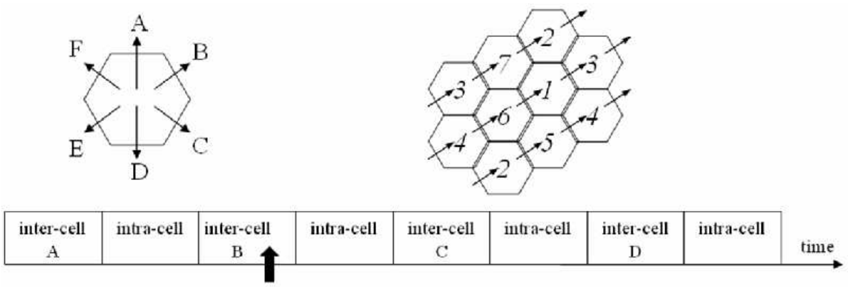

Figure 1: Communication using *I-EDF*.

during multi-hop transmission are taken in order to keep this velocity. If the velocity can not be kept, the message is discarded. No real guarantees are given. If a message has traveled too fast up to a certain node, it has gathered some spare time, the node can then re-affect this spare time to other messages.

It is important to consider real-time constraints (thus message urgency) in routing protocols. SPEED [16] adds a field to the routing tables which represents the approximate velocity of a path. This velocity is determined using probe messages. A message will use a given path if the approximate velocity of this path is higher than the velocity the message needs to travel at to reach its destination on time. Due to the use of information velocity, SPEED can be used as a routing protocol in the RAP communication architecture [17].

Apart from fully distributed solutions, medium access can also be regulated using a centralized scheduler [18, 8, 19]. The scheduler needs to know the positions of all other nodes, and take into account the number of hops a message needs, to reach destination. Due to inherent scalability limitations, this type of solution is well suited for collaborative robots [20, 1, 21].

All these solutions' aim is to reduce the number of messages that reach destination late, but they don't avoid late arriving of messages. As a late message is useless, message can be lost. Depending on the application, a message arriving late can be unacceptable, and hard real-time communication systems, that guarantee on-time delivery, are needed.

2.2 Hard Real-Time Solutions

Even though soft real-time solutions minimize miss ratio, no messages can arrive late for critical applications. Transmission times need to be guaranteed and bounded; hard real-time solutions are thus needed. New solutions are needed since it is nearly impossible to modify an existing non-real-time protocol [22] and turn it into hard-real time. What's more, for a communication architecture to be hard real-time, all layers need to be hard real-time [23].

To our knowledge, only [24] presents a true hard real-time solution: Implicit Earliest Deadline First (*I-EDF*). This MAC protocol is based on Earliest Deadline First scheduling [24]. The message with the closest deadline is scheduled first. As depicted in Fig. 1, assumptions include a two-dimensional network with nodes organized in hexagonal cells. Each cell has 6 neighboring cells, and a router node is present in the center of each one. Communication and interference radii are assumed to be equal to the distance between two neighboring router nodes. *I-EDF* differentiates inter- and intra-cellular communication. As for intra-cellular communication, each cell is assigned a different frequency; 7 frequencies are sufficient to avoid inter-cell interference during intra-cellular communication. Inside a cell, each node knows its neighbors and the characteristics of all messages that need to be exchanged (frequency, deadline, duration). Using the EDF scheduling algorithm [24], all nodes of a cell construct the same scheduling table. Collisions are therefore avoided for intra-cellular communication. As for inter-cellular communication, the six direction of the hexagon are numbered A, B, $C\ldots$ and intra-cellular communication slots alternate with inter-cellular communication slots in a given direction. A cell's router emits using the frequency of the destination cell, resulting in directional inter-cellular communication. Thus, within the same inter-cellular communication slot, all routers will emit in the same direction (*e.g.* C). This direction is changed in a round-robin manner at each new inter-cellular communication slot. Using time-based scheduling together with multiple frequencies guarantees the collision-free nature of *I-EDF*.

Even though *I-EDF* is hard real-time [25], the assumptions are hard to satisfy. The rigid cell-based organization of the network does not seem compatible with a random deployment. What's more, equality between communication and interference radii is physically impossible. Finally, due to the large number of embedded components (GPS-like dynamic synchronization, multiple frequency transceivers) the sensor's individual cost would be too high for large-scale use.

We have proposed in [26] a first version of this hard real-time MAC protocol for linear wireless sensor networks. Even though some ideas are kept in the protocol we present in Section 3, initialization phase of the protocol presented in was not satisfying. What's more, no formal validation with QoS parameter extraction of the protocol has been proposed so far in .

3 Proposition

Our goal is to present a novel hard-real time MAC protocol with lower assumptions than *I-EDF*. We have decided to focus our attention on the MAC layer because, as hard real-time needs to be dealt with at each layer in communication architectures, a hard-real time routing layer for example would need a hard real-time MAC layer.

3.1 Hypothesis

The node's individual price needs to be as low as possible, on one side because a network can have a large number of nodes, on the other side because sensor loss is possible. To minimize production costs and to simplify deployment, all sensors are considered identical (no router nodes). The radio interface can only function at a single frequency, with a

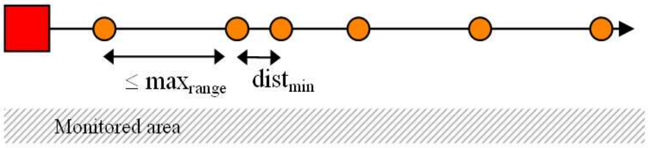

Figure 2: The network after deployment.

predefined, constant power. Processing power will not be considered high enough to use a CDMA scheme (Code Division Multiple Access), a technique where different logical channels are created by using orthogonal coding of information [27]. As nodes will not move, GPS-like dynamic positioning is considered useless. Synchronization on a global clock (which is possible using GPS) is therefore also excluded. Each node needs to know a certain number of global variables. It has to know its absolute position A, its maximum range max_{range}, and its radio interface's bandwidth BW. The node can know learn its absolute position during deployment, or it can be determined during a pre-initialization phase.

The monitored area needs to be linear, so as to free ourselves from routing considerations. A network is considered linear when a node's emission is heard at least at each border of the monitored area, *i.e.* communication range needs to be larger than the monitored area's width. Node position can then be represented by their projection on a line. Two neighboring nodes need to be able to communicate; they should therefore be within communication range. The distance between the two closest neighboring nodes $dist_{min}$ needs to be known by all. Finally, a sink node needs to be placed at one end of the network.

The wireless link is considered bidirectional: when A hears B, B assumes A can hear him. If node C is located between A and B, if A and B can communicate, C can communicate with either one of them. We consider a fading propagation equation, but no errors are taken into account (message loss for example).

Finally, alarms can be generated by any node, all of the alarms having the same priority. They need to reach the sink node in a multi-hop way before a deadline dictated by the application. Our goal is to propose a protocol that can guarantee a known and bounded transmission time. This time needs to be compatible with the application's deadline.

The network is represented in Fig. 2. The square represents the sink node, circles represent nodes.

3.2 General Idea

The network is organized into cells during an initialization phase. Cells are created so that each node of a cell can communicate with each node of a neighboring cell. Thanks to the cell creation mechanism, a certain level of reliability is guaranteed during this communication. Alarm transmission during run-time is done using two modes: protected mode and unprotected mode. Run-time starts in unprotected mode with a near optimal transmission speed, but several messages can collide. In case of collision, the network switches into a protected mode which is slower but collision-free (thus with a guaranteed delay). The sink decides when to switch back to unprotected mode.

3.3 Initialization

The initialization phase's goal is to organize the network nodes into cells. Each node will have two unique identifiers: its absolute position A and its identification with respect to the cells. Two types of signaling messages are used: $CREATION$ and END_INIT. Each message contains the sender's absolute position A_{sender}. A message is considered received if and only if it has been received with a Signal-to-Noise Ratio (SNR, [28]) higher than a threshold we will explicit. This guarantees a certain reliability in cell-to-neighboring-cell communication.

3.3.1 Medium access

For two messages not to collide during initialization, a backoff scheme is used. When a node emits, all nodes that have received the message determine a backoff time proportional to the difference between the node's position A and the position of the emitting node A_{sender}.

$$backoff = \frac{A - A_{sender}}{W_{initialization}} \tag{1}$$

The node can access the medium only when its backoff expires. $W_{initialization}$ is called "wave speed" because the overall view of expiring backoff timers forms a wave. $W_{initialization}$, in meters per second, is bounded. Indeed, if x and y are two neighboring nodes, with $A_x < A_y$, y needs to have received x's message entirely before its backoff expires. $W_{initialization}$ therefore needs to take into account the following parameters: the message's propagation time $T_{propagation}$, the time needed for the radio interface to switch between receiving and emitting modes $T_{inversion}$, and the message's length in bits $length_{CREATION}$ over the bandwidth BW.

$$W_{initialization} \leq \frac{dist_{min}}{T_{propagation} + \frac{length_{creation}}{BW} + T_{inversion}} \tag{2}$$

3.3.2 Initialization algorithm

1. The sink node emits $CREATION(1)$, creating cell 1. All the nodes that receive this message determine their backoff. During the backoff time, each node records the number of $CREATION$ messages it has received, the time of reception of the last one, and the number of the last created cell.

2. When a node's backoff expires and it has only received one $CREATION(i)$ message, it emits $CREATION(i+1)$ and is part of cell $i+1$.

3. When a nodes backoff expires and it has received $CREATION(i)$ and $CREATION(i+1)$, it is part of cell $i+1$. It starts a $timer_{error}$ (to be explained) from the moment it receives $CREATION(i+1)$. If $timer_{error}$ expires before it has received any other message, it emits $CREATION(i+2)$ and is part of this new cell.

4. When emitting a $CREATION$ message, a node also starts $timer_{last}$. If it expires before receiving any new message, the node knows it is the last node of the network, and sends out an $END\_INIT(1)$ message

5. A node that receives END_INIT(i) sends out END_INIT(i+1) if it has sent out a $CREATION$ message before.

$timer_{error}$ is used to detect the situation when $CREATION(i+1)$ has not been heard by more nodes than $CREATION(i)$. In this case, none of the nodes that have heard both will emit a new $CREATION$ message, resulting in an uncompleted initialization. To overcome this problem each node starts $timer_{error}$ on receiving the second $CREATION$ message $CREATION(i+1)$. When $timer_{error}$ elapses, the node emits $CREATION(i+2)$. Of course, all $timer_{error}$ have different values (depending on the nodes' locations) so as to elect only one emitting node (the one with the highest absolute position A).

$$timer_{error} = \frac{2 \times max_{range} - (A - A_{CREATION(i+1)})}{W_{initialization}} \tag{3}$$

$timer_{last}$ is used for a node to determine if it is the last node of the network. This timer is started by a node when it sends out a $CREATION$ message, and is aborted whenever another $CREATION$ message is heard. If none is heard and $timer_{last}$ expires, the node knows it is the last node of the network and sends out an END_INIT message to inform the sink node in a multi-hop way the initialization phase has ended.

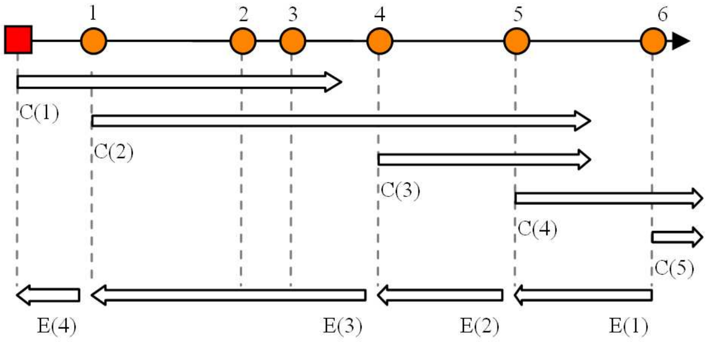

Figure 3: The cell creation mechanism.

Fig. 3 illustrates an example. $CREATION$ and END_INIT are shortened to C and E, respectively. Nodes are identified by their rank in the network, the nodes' communication

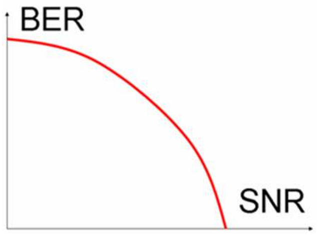

Figure 4: Relationship between BER and SNR.

ranges are represented by the arrows. The sink node sends out $CREATION(1)$, creating cell 1. Nodes 1, 2 and 3 which hear this message set their backoff timers. Nodes 1's backoff elapses first, it sends out $CREATION(2)$. When the backoff timers of nodes 3 and 4 elapse, they have received 2 $CREATION$ messages, and therefore take no action. The process goes on with node 4 which sends out $CREATION(3)$ when its timer elapses. $timer_{error}$ is useful at node 5. When its timer elapses, it takes no action, but as $CREATION(3)$ is not heard by more nodes than $CREATION(2)$, $timer_{error}$ will elapse and node 5 will send out $CREATION(4)$ not to stop the initialization process. Finally, node 6 sends out $CREATION(5)$ and after $timer_{last}$, it knows it is the last node of the network and sends out $END\_INIT(1)$ which is relayed in a multi-hop way until it reaches the sink node.

3.3.3 Identification

Each node is uniquely identified by the 2-tuple $[I, R]$, I being the node's cell number, R its relative position inside the cell. Relative position is calculated using the absolute positions of the node, its cell head and the next cell head. R is expressed in percentage.

3.3.4 Reliability

The protocol assumes that a message emitted by cell i is to be received by any node at cell $i-1$ or $i+1$ with a reliability higher than a threshold. We therefore need to construct our cells so that their length is small enough for communication between neighboring cells to be reliable.

Physical reliability metric is Bit Error Rate (BER): the ratio between the number of bits received with errors over the total number of bits received. BER can not be measured by nodes directly. Nevertheless, as depicted in Fig. 4, there is a one-to-one relation between BER and Signal-to-Noise Ratio (SNR) which is measurable by a node.

As depicted in Fig. 5, for a node to consider it has received a message, it needs to receive it with a SNR higher than a threshold corresponding to the BER threshold defined. This way, communication reliability is guaranteed, equal to the BER threshold known a priori by all nodes.

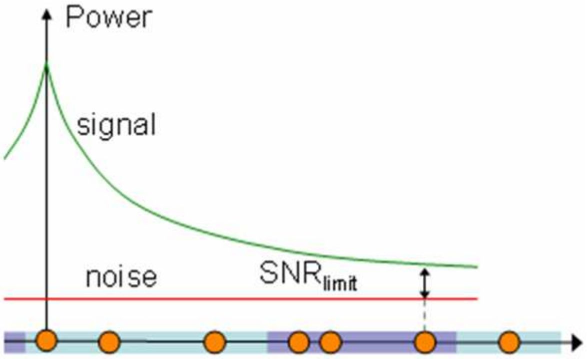

Figure 5: Realiability.

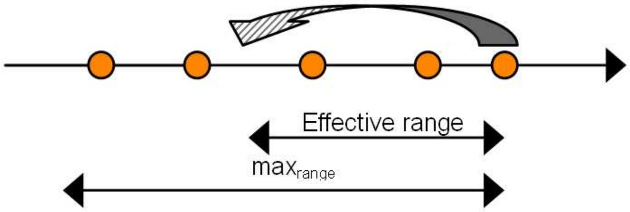

Figure 6: Unprotected mode.

3.4 Run-time in unprotected mode

This transmission mode is used as long as no collisions occur. Transmission speed towards the sink is near-optimal.

As depicted in Fig. 6, when a node sends out a message, the nodes that hear it start $backoff_{unprotected}$ calculated using the nodes absolute position A, the absolute position of the sender node A_{sender}, max_{range}, and $W_{emission}$ (which we will explicit). When no message has been received and $backoff_{unprotected}$ expires, the node is elected relaying node. The relaying node is the one closest to the sink which has heard the message. Apart from the relaying node election process (thus duration), the transmission speed is optimal.

$$backoff_{unprotected} = \frac{A - (A_{sender} - max_{range})}{W_{emission}} \tag{4}$$

$W_{emission}$ is the wave speed used in the election process, and needs to be bounded. Indeed, we want a node to be able to detect its neighboring node has started emitting before its $backoff_{unprotected}$ expires (thus it needs to relay the message). The speed can be very high, propagation and detection of a signal times ($T_{propagation}$ and $T_{detection}$, respectively) being very low.

$$W_{emission} \leq \frac{dist_{min}}{T_{propagation} + T_{detection}} \tag{5}$$

3.5 Switching from unprotected to protected mode

Run-time starts in unprotected mode. Nevertheless, if a node does not hear its alarm reemitted during a certain time period, it has detected that its message has collided. It then sends out a JAM signal. When a node hears such a message, it reemits it once. This way, the complete network will be jammed. It is worth nothing that if a JAM message collides with an alarm message, this collision will be detected and a new JAM message will be generated; the overall jamming process is not stopped. After sending the JAM message, all nodes wait for the worst case execution time of this process ($WCET_{jam}$, to be detailed), the messages that collided and any new message is sent in protected mode.

3.5.1 Synchronization

As no GPS-like chips are embedded, synchronization on a global clock is not possible. Nevertheless, protected mode needs network-wide synchronization to avoid collisions. We have created the novel concept of "synchronization wave", which are waves of expiring timers. We associate to each one of them a virtual wave speed. We propose to synchronize the network using a continuous series of synchronization waves.

After waiting $WCET_{jam}$, the sink node sends out a $SYNC$ message that will be heard by a group of nodes. These nodes start $backoff_{sync}$, equal to the difference between their relative position and that of the sender, using a wave speed W_{sync} we will explicit.

$$backoff_{sync} = \frac{R - R_{sender} + (A - A_{sender}) \times 100}{W_{sync}} \tag{6}$$

When $backoff_{sync}$ expires, the node starts a periodic timer T.

$$T = \frac{600}{W_{sync}} \tag{7}$$

This time period corresponds to the duration a synchronization wave needs to travel 6 cells. Indeed, W_{sync} is expressed in percentage of a cell per second, and $600 = 6 \times 100\%$ of one cell. The overall idea is to have a series of waves continuously traveling from the sink node at the same speed, all separated by exactly 6 cells.

This process will be used to avoid collisions. Indeed, we have considered that a node emitting at cell i will interfere with cells $i-1$ and $i-2$, thus 3 cells. For another node to send at the same time it needs to be separated by at least $2 \times 3 = 6$ cells for their messages not to collide. A node needs to wait for its $backoff_{sync}$ to expire before starting to emit, thus nodes starting to emit simultaneously are separated by 6 cells, their messages can not collide.

Exactly as during initialization, W_{sync} is bounded. W_{sync} is equal to $W_{initialization}$, the only different being that distances are expressed in percentage of a cell.

$$W_{sync} \leq \frac{\frac{dist_{min}}{max_{range}}}{T_{propagation} + \frac{length_{sync}}{BW} + T_{inversion}} \tag{8}$$

3.6 Run-time in protected mode

The network switches from unprotected to protected mode as soon as a collision occurs. This new mode needs to guarantee a collision-free functioning with finite and bounded transmission times. The processes involved in guaranteeing those requirements slow down the transmission speed. The overall idea is to reserve a portion of the network before sending out the alarm message. Within this portion, no new alarms can be generated and thus collision is avoided. Reservation is done using signaling messages. For those messages not to collide either, the wave synchronization is used. Finally, not to loose signaling messages, they are only exchanged between neighboring cells (where reliability is guaranteed). Three types of messages are used ($SILENCE$, ACK_EXP, $DATA$) each one containing the sending node's unique identifier: $[I, R]$ for the first two, A for $DATA$ messages.

3.6.1 Protection algorithm

This algorithm is depicted in Fig. 7. $SILENCE$ is shortened to S.

1. A node at cell i which wants to send out an alarm message waits for its $backoff_{sync}$ to expire and sends out $SILENCE(1)$. All nodes of cells i and $i-1$ are then reserved.
2. A node at cell $i-1$ is elected and sends out $SILENCE(2)$ in order to reserve the nodes of cell $i-2$ (load-balanced algorithm to be presented).
3. Steps (1) and (2) are repeated until cell $i-5$ is reserved. The elected node at this cell then sends out an ACK_EXP message.
4. This message is sent in a multi-hop mode to the initiating node using the unprotected mode algorithm (no collisions can happen as the portion is reserved).

3.6.2 Relaying node election

Only one relaying node must be elected. The algorithm depicted in Fig. 8 is used; its maximal duration is T. The idea is to keep the relaying node at a relative position closest to that of the sending node for load-balancing purposes. This algorithm is robust as a relaying node is elected as soon as there is a node in the cell. It is depicted in Fig. 8, where the emitting node's relative position is 50%.

1. All nodes of a cell hear a message from there neighboring cell and record the sender's relative position R_{sender}.
2. Knowing its relative position R, the instant when its $backoff_{sync}$ expires, and W_{sync}, it calculates the instant a synchronization wave enters the cell, and waits for that instant.
3. If $R < R_{sender}$ (*i.e.* the node is closer to the sink, relatively), it waits for a time proportional to (R-R_{sender}), relatively to W_{sync}.
4. If $R_{sender} < R$, it waits for a time proportional to R, relatively to W_{sync}.
5. The first node which timer elapses is elected relaying node and sends out the corresponding message ($SILENCE$ or ACK_EXP). This aborts the other nodes' timers.

We argue that, by giving the opportunity to talk first to nodes further away from the sender, we compensate the natural phenomenon where a signal is heard better closer to the sender. This solution thus avoids over-utilizing cell extremity node, which would reduce their lifetime.

3.6.3 Transmission

After this protection phase, the message can be transmitted. As the reserved portion is long enough for an alarm issued at cells further away not to collide with our alarm message, this

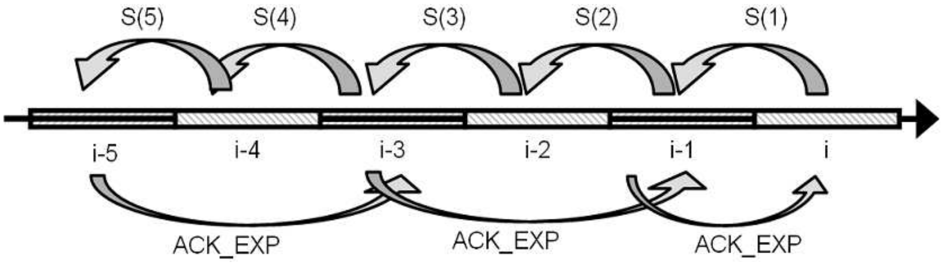

Figure 7: Protected mode.

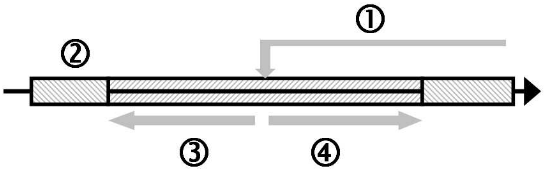

Figure 8: Relaying node election.

message can be sent in unprotected mode. The relaying node will start a new protection phase by sending out a $SILENCE(1)$ message. Alternation between protection and transmission causes the protected portion to be shifted, and the alarm to be sent to the sink with no collisions.

In order to absorb the flow of alarm messages, the sink will need to have the same behavior as the first node of cell 1 and as all the nodes of virtual cells 0, -1 . . . This way, the sink node absorbs the flow in a transparent way for the network.

We here consider that reserving 6 cells is enough. Nevertheless, it is possible to adapt to different ratios between communication and interference radii by reserving more (or less) cells and increasing (or decreasing) T.

3.7 Switching from protected to unprotected mode

No external event forces the network to switch back from protected to unprotected mode. Nevertheless, staying in protected mode implies reducing the multi-hop transmission speed. The sink is responsible for determining the switching back instant, which should be chosen in order to have a small collision probability. The sink node therefore waits for an idle period equal to the worst case transmission time (to be explicated in Section 4), and sends out a JAM signal. As for switching from unprotected to protected mode, the complete network is jammed. After waiting for $WCET_{switch}$, nodes send out their collided or new messages (if any), in unprotected mode.

Our MAC protocol regulates medium access based on the nodes' geographical position. Run-time is divided into two modes: an unprotected mode prone to collisions but with a near-optimal transmission speed; and collision-free yet slower protected mode. Initialization and run-time have bounded execution and transmission times (as validated in Section 4). It is therefore possible to re-launch initialization during run-time to cope with changes in transmission range for example (in case of rain).

4 Modeling and Validation

The protocol needs to guarantee a given Quality-of-Service. For our application class, the key parameter is bounded execution and transmission times. These times together with the protocol's behavior need to be formally validated. Roughly, we first analytically determine worst times for all phases of our protocol, and then validate those times using a formal model of the protocol's behavior.

4.1 Validation methodology using a formal model

To validate our protocol, it first needs to be modeled using a modeling formalism. The choice of this formalism depends on three criteria: expression power (*can the model express all the protocol's characteristics?*), modeling power (*can these characteristics be easily expressed?*)

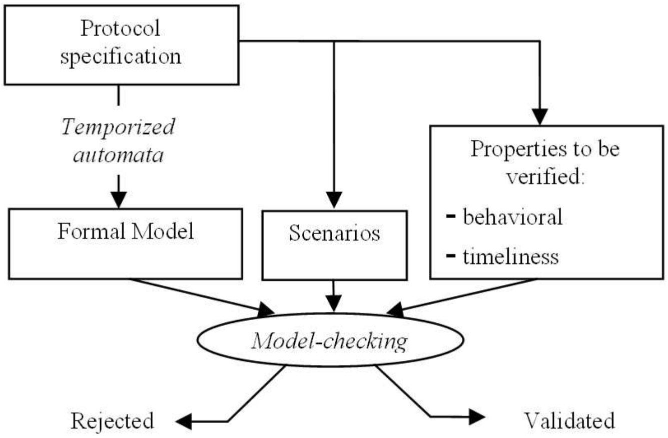

Figure 9: Model-checking methodology.

and analyzing/ verification power (*can this model be used to efficiently analyze the protocol it represents?*).

As for the expression power, our needs include communication protocol needs (parallelism and concurrence). What's more, concurrently executing instances need to be able to communicate. This communication can be done by signals. Finally, it should be possible to express both choice and timeliness constraints. As for analysis power, it should be possible to use the resulting model for validating time bounds.

To validate given time bounds, we chose to use exhaustive exploration of all execution paths by model-checking [29]. Even though this is not the only formal validation approach, [30] and [31] shows that model-checking is particularly suited for validation of real-time communication protocols. The model-checking methodology is depicted in Fig. 9. The properties that need to be verified by the protocol and the scenarios on which these properties need to be verified are extracted from the protocol specification (properties include behavioral and timeliness). A model-checking tool exhaustively explores the execution tree in order to validate a given property over a given scenario, using the protocol's formal model. The protocol is formally validated when all properties are validated over each scenario.

UPPAAL [32] is a complete modeling and formal validation architecture, which answers all criteria; it will be used in this work. We will briefly describe UPPAAL, for a more detailed description, see [33]. UPPAAL's embedded modeling formalism is TSA (Times Safety Automata) which is an extension of timed automata [34]. The system is represented by a set of concurrent state-transition automata. Local and global variables can be defined; numerical variables and clocks are defined in the same way. synchronization between automata is done by signals.

An automaton is composed of states and transitions. The system can stay in a state only if the associated invariant is satisfied. For passing a transition, the associate guard

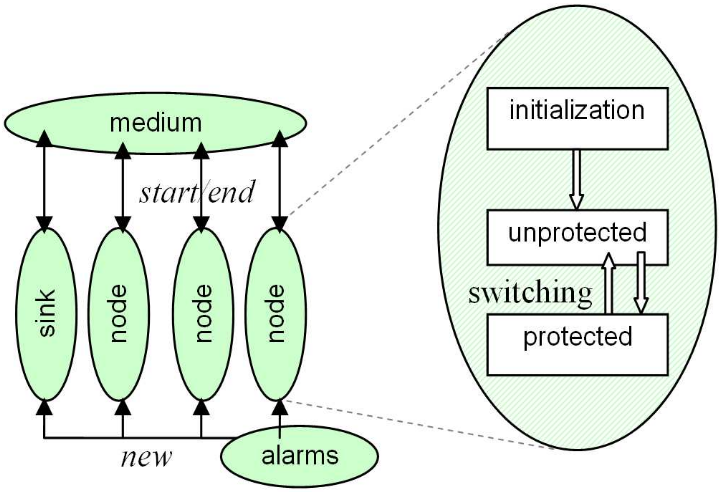

Figure 10: Modeling methodology.

needs to be satisfied, and it is possible on passing this transition to update a variable, or to send/receive a signal. Transitions are passed instantaneously. Different concurrently executing automata can communicate using signals; a signal can only be sent if another automaton is waiting for the signal.

Properties to be verified are expressed in a formal timed logic called TCTL (Timed Computation Tree Logic, [35]). Verification is done by exhaustively exploring all paths of the system's execution graph; properties are therefore expressed on execution paths.

4.2 Modeling methodology

The system can be described using components. Modeling such a component with UPPAAL is done in object oriented logic: automata are used to create different instances. This way, modeling the complete system is reduced to modeling four automata, describing the behavior of a node, the sink node, the medium and the alarm generation. It is to be noted, as depicted in Fig. 10, that the node's behavior is really modeled by four different automata, one four each phase of the protocol (initialization, unprotected mode, protected mode, switching). A complete topology is generated by creating a series of node instances, and by tuning the medium automata by injecting the ranges of each node's communication.

Communication between automata is done by synchronous signals (*new*, *start*, *end* in Fig. 10). With synchronous signals, an automaton needs to be willing to receive the signal for it to be sent, and signal transmission is instantaneous. Two modeling problems arise from using synchronous signals for communication. First, it is not possible to affect a variable to a signal, so message passing is not straightforward. This problem is nevertheless solved by using a global variable: the emitting node writes the passed message into this variable, and informs the receiver using a signal that it needs to read this variable. What's more signal passing being instantaneous, message duration modeling is not straightforward, and it is necessary to model both starting and ending instants of a message being transmitted (*start* and *end*, respectively).

When a message is emitted by a node, it is received by the medium automaton which will retransmit the message to the appropriate nodes, depending on the emitting node's transmission range.

4.2.1 Modeling example

By lack of space, we can not describe all automata, but chose to show a simplified version of the automaton modeling the behavior of a node in unprotected mode (Fig. 11). Even though it is the smallest one, this automaton is representative. Emission and reception of a message start is written *start*? And *start*!, respectively.

The automaton starts in the initial state (double circled). It waits in that state until it starts receiving a message (*start*?). When reception is completed (*end*?) it determines and waits for its $backoff_{unprotected} = A-(message[1]-max_{range})$ to elapse, and then sends out the message it has received. This transmission lasts for a duration called $duration_{data}$. After relaying the message, the automaton goes back in the idle initial state. If the node receives a message while it is waiting for its $backoff_{unprotected}$ to elapse, it aborts its waiting and goes back to initial state (another relaying with a smaller $backoff_{unprotected}$ has been elected). Finally, the node can receive a new signal. This signal comes from the alarm generation automaton, and means that the sensor part of the node has detected a monitored event, so a new alarm needs to be generated and sent to the sink node.

4.3 Formal validation of our protocol

Formal validation is done in two phases: analysis of the protocol and formal validation of the bounds found during the analysis. Validation is done using a formal model.

We first analyze our protocol and identify the worst case scenarios in terms of execution or transmission times. We then explicit those $WCET$ (initialization, switching) and $WCTT$ (unprotected mode, protected mode) using the protocol parameters. A formal validation of those analytically-based worst times needs to be done. A UPPAAL model of the protocol's behavior is constructed, and a formal validation of the protocol's behavior and its timeliness characteristics is then performed using the methodology presented in part 4.2 (above). Both methods are complementary. Analysis of our protocol gives parameterized $WCET$ and $WCTT$ formulas; model-checking offers a formal behavioral and timeliness validation and lets use verify the time bounds found during the analysis of our protocol.

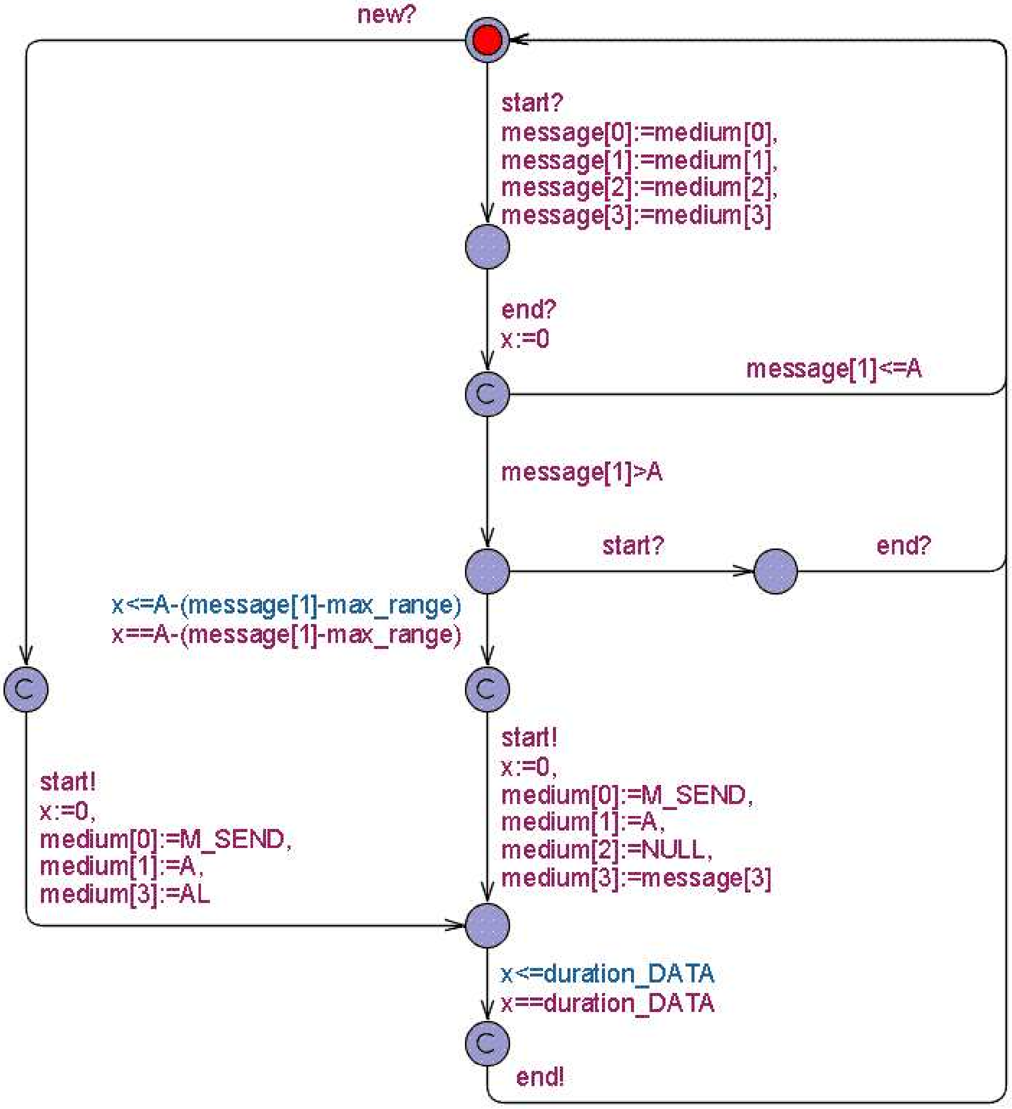

Figure 11: Simplified model of the node's behavior in unprotected mode.

4.3.1 Analytical worst case times

In this part, we will explain the analysis done for determining the worst case transmission time of an alarm in unprotected mode ($WCTT_{unprotected}$), protected mode ($WCTT_{protected}$), initialization ($WCET_{initialization}$) and switching between modes ($WCET_{switch}$).

As for $WCTT_{unprotected}$, worst case transmission time is obtained when the alarm needs to travel the longest distance, thus be generated by the last node of the network. What's more, transmission time is maximized when each node needs to relay the message, *i.e.* hop distance is equal to the inter-node distance of the sender and the next-hop node. At each hop, we will have a total time equal to the sum of transmission time and relaying node election time. Transmission time is calculated using the message length in bits $length_{data}$ and the available bandwidth BW. Relaying node election only takes place when the emitting node's cell number is higher than 2; at cell 1, the node can send the alarm in one hop to the sink, so relaying node election is not needed. Election duration is calculated using $W_{emission}$. The "virtual wave" will need to travel for a mean distance equal to max_{range}-($length_{network}/number_{nodes}$). Those observations are assembled in the equation of $WCTT_{unprotected}$.

$$WCTT_{unprotected} = number_{nodes} \times \left[\frac{length_{data}}{BW} + \left(\frac{max_{range} - \frac{length_{network}}{number_{nodes}}}{W_{emission}} \right)_{ifI>2} \right] \quad (9)$$

To construct $WCTT_{protected}$, a different approach is taken. Worst case transmission time is still obtained when the last node emits the alarm (thus the last cell), but at each hop, the alarm will be sent to the neighboring cell. We sum the total transmission time with the total protection phase duration. One-hop transmission time is calculated using $length_{data}$ over BW, which is then multiplied by the number of hops ($number_{cells} - 1$). The next summed up terms refer to the duration of the protection phase. It is to be noted that protection phase is carried out completely (*i.e.* cell reservation is done until cell $i-5$, i being the alarm emitting node's cell number) when cell number is higher or equal than 7. In case the sender's cell number is smaller, the sink will reproduce the behavior of cells 0, -1, ... but will instantaneously acknowledge the reservation of those cells. This explains why total protection phase duration summed up terms are increasing with cell number ($number_{cells} \geq 2$, ..., $number_{cells} \geq 7$). A complete protection phase (possible if $number_{cells} \geq 7$) lasts for a time equal to the sum of the duration of 5 ACK_EXP message emissions and 6 waiting times for a synchronization wave to pass.

$$\begin{aligned}
WCTT_{protected} = {} & (number_{cells} - 1) \times \frac{length_{data}}{BW} + \\
& (3 \times T + \frac{length_{ack\_exp}}{BW})_{if number_{cells} \geq 2} + \\
& (3 \times T + 2 \times \frac{length_{ack\_exp}}{BW})_{if number_{cells} \geq 3} + \\
& (4 \times T + 3 \times \frac{length_{ack\_exp}}{BW})_{if number_{cells} \geq 4} + \\
& (5 \times T + 4 \times \frac{length_{ack\_exp}}{BW})_{if number_{cells} \geq 5} + \\
& (6 \times T + 5 \times \frac{length_{ack\_exp}}{BW})_{if number_{cells} \geq 6} + \\
& (6 \times T + 5 \times \frac{length_{ack\_exp}}{BW})_{if number_{cells} \geq 7} \times (number_{cells} - 5)
\end{aligned} \tag{10}$$

$WCET_{initialization}$ is determined by summing up the worst case times of each phase in the initialization process: the initialization wave traveling from sink node to last node, the error case happening on half of the nodes (worst case), determination of the last node, and multi-hop transmission of the END_INIT message from last node to sink node. The initialization wave travels at a predetermined constant speed $W_{initialization}$, over a length of $network_{length}$. An initialization error case occurs when a node should send a $CREATION$ message in order not to stop initialization when it already received two $CREATION$ messages. Protocol analysis shows that this process takes up to the duration the initialization wave takes to travel for a distance of $2 \times max_{range}$. Finally, at each hop, the END_INIT message changes cells, so $number_{cells}$-1 hops are needed to reach the sink node.

$$\begin{aligned}
WCET_{initialization} = {} & \frac{network_{length}}{W_{initialization}} + \\
& \left\lceil \frac{number_{nodes} - 1}{2} \right\rceil \times \frac{2 \times max_{range}}{W_{initialization}} + \\
& \frac{2 \times max_{range}}{W_{initialization}} + \\
& (number_{cells} - 1) \times \frac{length_{end\_init}}{BW}
\end{aligned} \tag{11}$$

As for $WCET_{switch}$, worst case time is obtained when the last node of the network detects a collision. The JAM message reaches the sink after a maximum of $number_{nodes}$ hops, one hop lasting for the transmission time of a JAM message. The synchronization wave travels at a constant speed of $W_{synchronization}$ expressed in percentage of a cell per second, and it

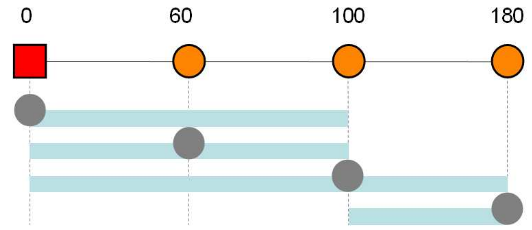

Figure 12: The studied deployment.

needs to travel $number_{cells}$-1 cells of 100%. Finally, the $SYNC$ message is emitted one last time by the network's last node.

$$\begin{aligned} WCET_{switch} = {} & number_{nodes} \times \frac{length_{jam}}{BW} + \\ & \frac{(number_{cells} - 1) \times 100}{W_{synchronization}} + \\ & \frac{length_{sync}}{BW} \end{aligned} \tag{12}$$

As shown by this analysis, all worst case times depend on the network length, be it in meters, number of cells or number of nodes. A larger network implies larger initialization and transmission times. Scalability can be achieved, but it is necessary to validate that the worst case times at a given network size are acceptable by the application. If not, a large network can be subdivided in smaller ones, a sink node then needs to be place at one extremity of each portion.

4.3.2 Detailed scenario

Formal validation of the analytical worst case times is done by using the formal model. As presented in the model-checking methodology (Part 4.2, above), validation is done using scenarios. These scenarios are complementary and should put the protocol in all behavior cases. All scenarios are different in term of number of nodes, node positions, ranges and alarm generation.

In this part, we describe in detail one scenario, and the behavioral and timeliness validation it offers. It is to be noted that a large number of other scenarios have been used in order to validate all the aspects of the protocol detailed in Part 3). We use the deployment depicted in Fig. 12, where the square represents the sink and the circles the nodes (identified by there absolute position); the horizontal bars represent the nodes' ranges. $max_{range} = 100$, $W_{initialization} = 1$, $BW = 1$, $length_{creation} = length_{END\_INIT} = 3$, $length_{data} = 10$.

We will study the alarm's multi-hop transmission in unprotected mode. As duration increases with the distance to travel, we consider that the alarm is generated at the last node ($A = 180$). We will note t the absolute time. Node at 180 sends out the alarm from $t = 0$ to $t = 10$. It is received by the node at 100 which determines $backoff_{unprotected} = 20$. Node at 100 relays the alarm between $t = 30$ and $t = 40$. Multi-hop transmission lasted for 40. This value is confirmed by UPPAAL.

We now use the analytical formula $WCTT_{unprotected}$: $WCTT = 3 \times (10 + (100 - 60)) = 150$. The difference between this case and the worst case is that node at 60 does not need to relay the message. UPPAAL has proven in this scenario that the protocol's behavior is what we expected it to be, and that the $WCET_{unprotected}$ is larger that the effective transmission time.

4.3.3 Behavior cases validated by the scenarios

We have so far detailed one scenario, by validating the behavior and timeliness characteristics of our protocol "by hand" and using UPPAAL. Validating all scenarios by hand would of course be time-consuming and error-prone; therefore UPPAAL is used for systematic validation of behavioral and timeliness characteristics.

By confronting UPPAAL and the analytical formulas of $WCET$ and $WCTT$, we have formally validated the following characteristics of our protocol:

- Initialization phase:
    - Determination of $backoff_{initialization}$
    - Error case during initialization, when a node should emit a new $CREATION$ message not to stop initialization, even though it has already received two $CREATION$ messages
    - Detection by a node that it the last one in the network
    - Initialization of the network, and identification of the nodes
    - $WCET_{initialization}$
- Unprotected mode:
    - Multi-hop alarm transmission
    - $WCTT_{unprotected}$
- Switching between modes:
    - Switching between modes
    - $WCET_{switch}$

- Protected mode:
    - Relaying node election
    - Switching between protection and transmission phases
    - Shifting of the protected area
    - Multi-hop alarm transmission
    - $WCTT_{protected}$

We argue that all characteristics of the protocol have been covered and formally validated thanks to the set of scenarios we have used.

5 Conclusion and Future Work

The originality of our work is that we have considered applications that need a network Quality-of-Service in term of guaranteed transmission times.

We proposed a novel MAC protocol with assumption as realistic as possible. Our solution, based on alternation between two modes, offers both a near-optimal multi-hop transmission time towards the sink in unprotected mode, and a deterministic collision-free protected mode. Radio link reality is taken into account, using a cell based organization, cells being constructed in order to achieve reliable transmission.

A behavioral and timeliness validation of our protocol has been presented, using a model-checking methodology, and UPPAAL. The $WCET$ (Worst Case Execution Time) and $WCTT$ (Worst Case Transmission Time) of our protocol have been determined and formally validated.

Currently, we have considered only one alarm generated per cell. It would be interesting to determine the number of alarms per cell and per time unit the network can support. The system of course always needs to provide hard real-time characteristics.

What's more, at the physical level, we have considered a fading link with distance to the sending node, but we have not considered message loss in a communication between two in-range nodes. Node loss has not been taken into account. It would be interesting to study this aspect and add fault-tolerance mechanisms to our protocol.

Even though one dimensional applications exist, a two-dimension extension of the protocol would greatly widen the spectrum of possible applications. This extension would nevertheless imply adding a routing layer that needs to offer hard-real time guarantees for the overall communication architecture to be hard real-time.

Finally, it would be interesting to compare our protocol's mean time performances to performances of other, possible non real-time, existing protocols. A formal prove that the other protocols do not guarantee hard real-time constraints could be given.

References

[1] R. Lin, Z. Wang, and Y. Sun, "Wireless sensor networks solutions for real time monitoring of nuclear power plant," in *World Congress on Intelligent Control and Automation*, June 2004.

[2] I. F. Akyildiz and I. Kasimoglu, "Wireless sensor and actor networks: research challenges," in *International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. Fort Lauderdale, Florida, USA: IEEE, December 2004.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks (Elsevier) Journal*, vol. 38, no. 4, pp. 393–422, March 2002.

[4] "Crossbow, inc., berkeley mote," last visited on 16/11/2005. [Online]. Available: http://www.xbow.com/Products/Wireless_Sensor_Networks.htm

[5] "Smart dust research project homepage," last visited on 16/11/2005. [Online]. Available: http://robotics.eecs.berkeley.edu/~pister/SmartDust/

[6] V. Rajavavivarme, Y. Yang, and T. Yang, "An overview of wireless sensor network and applications," in *Southeastern Symposium on System Theory*. Auburn, AL, USA: IEEE, March 2003.

[7] J. A. Stankovic, "Research challenges for wireless sensor networks," *SIGBED Review: Special Issue on Embedded Sensor Networks and Wireless Computing*, vol. 1, no. 2, pp. 1–4, July 2004.

[8] T. Facchinetti and G. Buttazzo, "Integrated wireless communication protocol for ad-hoc mobile networks," in *International Workshop on Real-Time Networks (RTN)*, Catania, Italy, June 2004, pp. 43–46.

[9] P. Boone, "Real-time communication and coordination in wireless embedded sensor networks," April 2004, unpublished.

[10] T. F. Abdelzaher, S. Prabh, and R. Kiran, "On real-time capacity limits of multi-hop wireless sensor networks," in *Real-Time Systems Symposium (RTSS)*. Lisbon, Portugal: IEEE, December 2004.

[11] J. A. Stankovic, T. F. Abdelzaher, C. Lu, L. Sha, and J. C. Hou, "Real-time communication and coordination in embedded sensor network," in *Proceedings of the IEEE*, vol. 91, no. 7, July 2003, pp. 1002–1022.

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

[13] A. S. Tanenbaum, *Computer Networks, Fourth Edition*, A. S. Tanenbaum, Ed. Prentice Hall, August 2002.

[14] J. Sheu, C. Liu, S. Wu, and Y. Tseng, "A priority mac protocol to support realtime traffic in ad-hoc networks," *ACM Wireless Networks*, vol. 10, pp. 61–69, January 2004.

[15] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He, "Rap: A real-time communication architecture for large-scale wireless sensor networks," in *Real-Time Technology and Application Symposium (RTAS)*. San Jose, CA, USA: IEEE, September 2002.

[16] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher, "Speed: a stateless protocol for real-time communication in sensor networks," in *International Conference on Distribued Computing Systems (ICDCS)*. Rhode Island, USA: IEEE, May 2003.

[17] T. F. Abdelzaher, J. A. Stankovic, S. Son, B. Blum, T. He, and A. Wood, "A communication architecture and programming abstractions for real-time embedded sensor networks," in *International Conference on Distributed Computing Systems (ICDCS) Workshops*, Rhode Island, USA, May 2003.

[18] T. Facchinetti, L. Almeida, G. Buttazzo, and C. Marchini, "Real-time resource reservation protocol for wireless mobile ad-hoc networks," in *Real-Time Systems Symposium (RTSS)*. Lisbon, Portugal: IEEE, December 2004, pp. 382–391.

[19] T. Facchinetti, G. Buttazo, M. Caccamo, and L. Almeida, "Wireless real-time communication protocol for cooperating mobile units," in *Euromicro Conference on Real-Time Systems*, Belek near Antalya, Turkey, July 2003.

[20] H. Li, P. Shenoy, and K. Ramamritham, "Scheduling communication in real-time sensor application," in *Real-Time and Embedded Technology and Applications Symposium (RTAS)*. Toronto, Canada: IEEE, May 2004, pp. 10–18.

[21] "Sebastian thrun's homepage," last visited on 16/11/2005. [Online]. Available: http://robots.stanford.edu/

[22] A. Chandra, V. Gummalla, and J. Limb, "Wireless medium access control protocols," *IEEE Communications Surveys and Tutorials*, vol. 3, no. 2, pp. 2–15, 2000.

[23] H. Li, P. Shenoy, and K. Ramamritham, "Scheduling messages with deadlines in multihop real-time sensor networks," in *Real-Time and Embedded Technology and Applications Symposium (RTAS)*. San Francisco, CA, USA: IEEE, March 2005.

[24] M. Caccamo, L. Y. Zhang, L. Sha, and G. Buttazzo, "An implicit prioritized access protocol for wireless sensor networks," in *Real-Time System Symposium (RTSS)*. IEEE, December 2002.

[25] M. Caccamo and L. Y. Zhang, "The capacity of implicit edf in wireless sensor networks," in *Euromicro Conference on Real-Time Systems (ECRTS)*. IEEE, 2003.

[26] T. Watteyne and I. Augé-Blum, "Proposition of a hard real-time mac protocol for wireless sensor networks," in *International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*. Atlanta, GA, USA: IEEE, September 2005, pp. 532–535.

[27] G. Orfanos, J. Habetha, and L. Liu, "Mc-cdma based ieee 802.11 wireless lan," in *International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*. Volendam, The Netherlands: IEEE, October 2004, pp. 400–405.

[28] S. Saunders, *Antennas and propagation for wireless communication systems*, S. Saunders, Ed. Wiley, 1999.

[29] R. Alur, C. Courcoubetis, and D. D.L., "Model-checking in dense real-time," *Journal of Information and Computation*, vol. 104, no. 1, pp. 2–34, 1993.

[30] K. Godary, "Validation temporelle de réseaux embarqués critiques et fiables pour l'automobile," Ph.D. dissertation, CITI Laboratory, INSA de Lyon, France, November 2004.

[31] K. Godary, I. Augé-Blum, and A. Mignotte, "Sdl and timed petri nets versus uppaal for the validation of embedded architecture in automotive," in *Forum on Specification and Design Languages (FDL)*, Lille, France, September 2004.

[32] "Uppaal home page," last Visited on 16/11/2005. [Online]. Available: http://www.uppaal.com/

[33] G. Larsen, P. Petterson, and W. Yi, "Uppaal in a nutshell," *International Journal on Software Tools for Technology Transfer*, vol. 1, no. 1, pp. 134–152, December 1997.

[34] J. Bergson and W. Yi, "Timed automata: Semantics, algorithms and tools," *Lectures on Concurrency and Petri Nets*, vol. 3098, pp. 87–124, 2004.

[35] R. Alur and D. Dill, "Automata for modeling real-time systems," in *International Colloquium on Automata, Languages and Programming (ICALP)*, M. Paterson, Ed., vol. 443. Warwick University, England: Springer, July 1990, pp. 321–335.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399