Modeling Local Broadcast Behavior of Wireless Sensor Networks with Timed Automata for Model Checking of WCTT *

Alexandre Mouradian and Isabelle Augé-Blum
Université de Lyon, INRIA, INSA-Lyon, CITI, F-69621, France
firstname.lastname@insa-lyon.fr

ABSTRACT

Wireless Sensor Networks (WSNs) are usually deployed in order to monitor parameters of an area. When an event occurs in the network an alarm is sent to a special node called the sink. In critical real-time applications, when an event is detected, the Worst Case Traversal Time (WCTT) of the message must be bounded. Although real-time protocols for WSNs have been proposed, they are rarely formally verified. The model checking of WSNs is a challenging problem for several reasons. First, WSNs are usually large scale so it induces state space explosion during the verification. Moreover, wireless communications produce a local broadcast behavior which means that a packet is received only by nodes which are in the radio range of the sender. Finally, the radio link is probabilistic. The modeling of those aspects of the wireless link is not straightforward and it has to be done in a way that mitigate the state space explosion problem. In this paper we particularly focus on the modeling of the local broadcast behavior with Timed Automata (TA). We use TA because they have sufficient expressiveness and analysis power in order to check time properties of protocols, as shown in the paper. Three ways of modeling local broadcast with synchronizations of TA are presented. We compare them and show that they produce different state space sizes and execution times during the model checking process. We run several model checking on a simple WSN protocol and we conclude that one model mitigate the state explosion problem better than the others. In the future, the next step will be to enhance this model with the probabilistic aspect of radio communications and to show it remains the best one.

Categories and Subject Descriptors

C.4 [**PERFORMANCE OF SYSTEMS**]: Modeling techniques; C.3 [**SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS**]: Real-time and embedded systems; C.2.2 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Protocols—*Protocol verification*

General Terms

VERIFICATION

1. INTRODUCTION

A Wireless Sensor Network (WSN) is composed of nodes deployed in an area in order to monitor parameters of the environment. Those nodes are able to send information to dedicated nodes called sinks without the need of a fixed network infrastructure. Every node is able to forward messages from other nodes. They usually run on batteries so they should consume as little energy as possible in order to increase the network lifetime. Because WSNs can contain thousands of nodes, the financial cost of a node should be as low as possible, this leads to design nodes with poor capabilities (computation, radio, memory, etc...). For these reasons, network protocols have been designed mainly in order to optimize energy consumption and to provide autonomous network mechanisms. Nevertheless some applications need more than these characteristics. Indeed, critical applications require more reliability and the respect of time constraints. For example volcano monitoring application [19] should give guarantees on the delivery and end-to-end delay of alarm packets. The use of formal methods is essential in order to produce such guarantees.

In this paper we focus on time properties of WSNs protocols. We use the model checking approach because it has already been proven to be efficient to validate protocol designs [7] but mainly in wired networks. Theorem proving could have been considered because it can allow to obtain more general conclusions than model checking, but the authors of [7] consider that it is a slower and more error-prone process. There exists several formalisms for representing timed systems (Time Petri Nets, Process Algebra and higher level formal languages). We choose Extended TA formalism and UPPAAL [10] for several reasons explained in section 2.

The main specificities of WSNs are their large scale and the use of radio medium to communicate. The large scale aspect implies that the verifying method must handle large systems and thus avoid, as far as possible, state space explosion. The radio medium implies that, when a node sends a message, it is received by the nodes which are in the communication range of the sender, we name this local broadcast. This behavior requires that the representation of the connectivity

*This work has been partially founded by French Agence Nationale de la Recherche under contract VERSO 2009-017.

in the network is provided in the model. The radio medium is also unreliable which means that there is a probability that the packet is not received. Thus, when a real-time WSN protocol is verified, the guaranty on the WCTT must be given with a corresponding probability that depends on the reliability of the radio link.

In this paper we focus on the modeling of the local broadcast because it is an essential characteristic of WSNs. Moreover, the state explosion due to local broadcast modeling has to be addressed before we can take into account the probability aspect of the radio link. Indeed, adding probabilities increases the complexity of the model and thus it may worsen the state explosion problem. This second aspect of wireless communications will be treated in future works. Usually, to model a network [15] [4], the behavior of each node is described with transition system and the communications between nodes are represented using synchronizations between transition systems. Solutions to model local broadcast in wireless networks have been proposed in the literature [22] [11] [20] [8]. We describe and compare those solutions in terms of convenience for the user and performances such as the size of the state space and execution duration of model checking of Worst Case Traversal Time (WCTT) bounds. Indeed, the WSNs are large scale networks so the state space explosion problem has to be mitigated in order to verify properties on WSNs protocols. We give a method in order to efficiently represent local broadcast and mitigate state explosion.

In section 2 we motivate the choice of Extended TA and detail the propositions of the literature to model local broadcast. In section 3 we give the problem statement. In section 4 we define precisely three ways of modeling local broadcast for WSNs and we compare the propositions by the mean of simple examples. In section 5 we confirm the results of the comparison by checking WCTT bounds of a simple WSNs protocol and we give a method to integrate the connectivity information into the model. We present the performances of the three methods and discuss the pros and cons of each. In section 6 we conclude and give future works.

2. DEFINITIONS AND RELATED WORK

Over the years, many formalisms have been proposed to formally describe concurrent systems and protocols. In this section we present such formalisms and motivate our choice to use Extended TA of UPPAAL. We give some useful definitions about TA. We then describe the different solutions of the literature to model local broadcast.

Some formalisms provide explicit representation of time whereas others do not. If no explicit representation of time is provided, delay is modeled by several silent transitions (transition with an unobservable action, called tick action). This representation of delays leads to very large transition systems. Nevertheless discrete time can be used in the case of synchronous systems, a clock pulse is represented by a transition. For asynchronous systems the elements can have different clocks and different speeds so continuous time domain is more convenient for representing the system and its properties. WSNs can be seen as asynchronous because there is no global clock in the system (each node has its own clock). Since we want to verify time properties of WSNs protocols, we focus on formalisms which provide an explicit representation of time.

One of the first formalisms to represent concurrency was Petri nets [3]. Time Petri Nets (TPN) allow explicit representation of time. Modeling and verification tools such as TINA [2] and ROMEO [9] are available for this formalism. Nevertheless, [5] shows that TA are strictly more expressive than TPN. Moreover network protocols are often described with Finite State Machine (FSM) which are conveniently translatable in TA.

Milner then introduced process algebra with CCS [14]. A process algebra is a mathematical structure which satisfies axioms on basic operators, a process being an element of the algebra. Process algebras have been extended with explicit representation of time [21]. In [18] a process calculus for mobile ad hoc network is proposed but it does not provide explicit representation of time and there is no model checking tool available for this process algebra. Nevertheless, process algebra defines synchronization between process which is useful to represent the communications in a network.

TA have been introduced by Alur and Dill [1] but most of tools use Time Safety Automata (TSA) [12]. In this paper, as in the literature, TSA are referred as TA. A TA is composed of locations and transitions between locations. Transitions are instantaneous and time can pass in locations. Conditions on transitions named guards can be defined.

Formally a TA is a tuple $TA = (Loc, Act, X, \rightarrow, Loc_0, Inv, L)$ where:

- Loc is a set of locations and $Loc_0 \subseteq Loc$ is a set of initial locations
- Act is a set of actions
- X is a set of clocks
- $\rightarrow \subseteq Loc \times \zeta(X) \times Act \times 2^X \times Loc$ is the transition relation
- $Inv : Loc \rightarrow \zeta(X)$ is the invariant assignment function
- $L : Loc \rightarrow 2^{AP}$ is a location labeling function with AP a set of atomic propositions

$\zeta(X)$ represents clock constraints. In a location time can pass if the invariant is not violated. Any transition which guard is satisfied can be taken. One enabled transition is selected nondeterministically. A configuration of a TA is a tuple (l, v) where $l \in Loc$ is a location and $v \in \nu(X)$ is a clock valuation.

A Network of TA (NTA) is the parallel composition of $\{A_i\}_{1 \leq i \leq n}$ TA where $A_i = (Loc_i, Act, X, \rightarrow_i, Loc_i^0, Inv_i, L_i)$ for $1 \leq i \leq n$. A configuration of a NTA is a tuple $(\bar{l}, v)$ where $\bar{l} \in L_1 \times ... \times L_n$ is a location vector and $v \in \nu(X)$ is a clock valuation. Since $\nu(X) \in \mathbb{R}^+$, the number of configurations is infinite and uncountable. In order to mitigate that problem zones have been introduced, they are defined as sets of clock valuations (details can be found in [4]). Thus with zones a configuration of a NTA become a tuple $(\bar{l}, z)$ where $\bar{l} \in L_1 \times ... \times L_n$ is a location vector and $z \in Z$ is a zone,

for the remainder of the paper we will use this definition of configuration.

UPPAAL [10] is a state of the art model checker. The modeling formalism is Extended TA which are TA extended with data types (integer, arrays ...). These data types allow to represent variables of the protocol and store the connectivity information. Moreover, special locations called urgent and committed locations are defined. In urgent locations time cannot pass, an edge has to be taken immediately. In committed location time cannot pass and if any automaton of the network is in a committed location, the next transition must be taken from a committed location. Special actions called synchronizations can be defined, these actions are taken from CCS [14].

There also exit languages based on TA such as IF[15]. In IF, a set of processes can be specified. Each process is a TA thus the set of processes defines a NTA. The processes can exchange messages using a FIFO queue, synchronizations are transitions triggered by the reception of a message. As in UPPAAL, IF provides urgent transitions. Nevertheless the tools associated with IF provide less features than UPPAAL. For example UPPAAL-Pro allows to define probabilistic models which could be used to represent the probabilistic radio link.

Real-time Maude [16] is a time rewrite theory formalism. It can be used to describe WSNs protocols. The nodes are described as objects with a location, a communication range, an amount of energy and so on. This representation seems very intuitive but the definition of the behavior of a node requires a good knowledge of rewrite theory. In [16] the authors describe and verify a WSN protocol. Nevertheless due to state space explosion the verification is performed on networks composed of only up to 6 nodes.

For all those reasons we choose to use Extended TA of UPPAAL in order to model the network and then perform the model checking. Moreover UPPAAL is state-of-the-art model checker for timed systems which has been successfully used to verify network protocols [17].

Now that we motivated the choice of the formalism and model checking tool, we focus on the modeling of the local broadcast behavior. The literature provides different ways of modeling this behavior. They are all based on synchronizations of TA. They are described in the following paragraphs.

In [10] the modeling of a network communication is done by a synchronization of TA with value passing. Communications in WSNs and ad hoc networks correspond to what authors call one-way conditional value passing synchronization, where the condition is that there is connectivity between the sender and the receiver and the value passed is the information contained in the transmitted packet. It does not explicitly provide a way of modeling local broadcast but it inspired the next references.

In [22] the authors verify properties of an ad hoc routing protocol (LUNAR) with SPIN [13] and UPPAAL. SPIN is a model checker which uses automata formalism with no explicit representation of time. In the paper the connectivity among nodes is represented using a two dimensional array of boolean. It is symmetric so it is assumed that the links are bidirectional. In the case of SPIN, the local broadcast is modeled by synchronizing to all the neighbors of the sender. One unicast synchronization is done for each neighbor. UPPAAL provides broadcast synchronization. Thus in the case of UPPAAL modeling, the sender synchronizes with the whole network, each node of the network then verifies whether it is connected to the sender or not by checking the connectivity boolean array. With UPPAAL the unicast synchronization is not used because it would obligate the modeler to define a synchronization channel per pair of connected nodes. In [20] the same modeling method is used to check Quality of Service properties of Biomedical Sensor Networks with UPPAAL. The authors also compare their model checking results to simulations results.

In [8] the authors model and verify the LMAC protocol using UPPAAL. They check every possible connected topologies of four and five nodes. As in [22] and [20] a connectivity matrix is used. Nevertheless in this case the receivers synchronize with the sender only if they are connected to it. The connectivity is checked before the synchronization. The authors are able to find relevant faults but only in small networks.

In [11] another approach is used. The authors propose to model ad hoc networks with UPPAAL by defining mobility of nodes, unicast communication and local broadcast. The mobility is defined with the notion of location, the change of location signifies mobility. The location of a node is defined with a set of groups. What the authors called groups in the paper are actually maximal cliques. There is no information on how the groups are computed and updated. The local broadcast is modeled with a broadcast synchronization and a guard, the guard prevents the nodes which are not in the same group as the sender to synchronize. The information on the groups of the sender is stored in a global variable. The way the global variable is updated is not mentioned. Moreover the usage of groups is not motivated in the paper, a two dimensional array of boolean could be used as well. It is also not clear if there is a broadcast channel per group or only one for the whole network.

The previously cited process calculus for mobile ad hoc networks [18] is an interesting proposition because the local broadcast modeling is native. Like [11], it uses the notion of maximal clique. Nevertheless the behavior of the nodes have to be modeled using a process calculus formalism which is less convenient to manipulate than TA.

In this paper, we are interested in the influence of the local broadcast modeling on the number of accessible configurations of the NTA i.e., the state space size. The state space explosion problem is indeed a main issue of model checking. In the literature solutions in order to model local broadcast have been proposed but, up to our knowledge, the impact on the state space size has not been studied.

3. PROBLEM STATEMENT

Model checking method has been successfully used to verify wired network protocols, but the WSNs have some specifities:

- Large scale (up to thousands of nodes)
- Wireless communications are unreliable and have a local broadcast behavior
- Dynamic topology (the nodes can run out of energy)

In this paper we focus on the modeling of the local broadcast and the state space size it induces during the model checking process. It is an important parameter since the WSNs are large scale and thus lead to state space explosion during the model checking. Choosing an appropriate local broadcast model is necessary to model WSNs and reduce state explosion. This work will allow to add the probabilistic aspect of the link and the dynamicity of the topology in future works, on a basis that already mitigates the state explosion problem.

The local broadcast behavior implies that, when a node sends a message, it is received by the nodes which are in the communication range of the sender. The formalism describes the local broadcast behavior with synchronizations, the nodes must synchronize according to the connectivity in the network. A representation of the connectivity graph must thus be included in the model.

The model must induce a state space size that fits in the memory of a computer. The way the local broadcast is modeled can influence the size of the state space. In the previous section we motivate the choice of TA formalism and we show that several methods to model the local broadcast behavior exist. In the remainder of the paper we compare those methods and discuss the results.

4. DEFINITION AND COMPARISON OF THE MODELING METHODS

In this section we define and compare 3 ways of modeling local broadcast. In order to compare these 3 solutions we consider a network depicted on Figure 1 of 5 nodes with a node which has a message to send (the source) and 4 nodes which are waiting for messages. We will compare the solutions by counting the number of accessible configurations in each case. Indeed, if the number of configurations induced by a solution is less than the other, this solution mitigates the state space explosion problem.

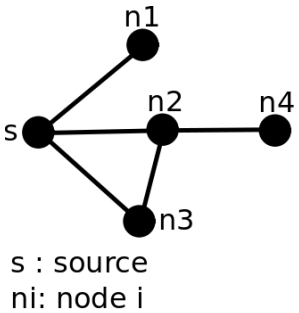
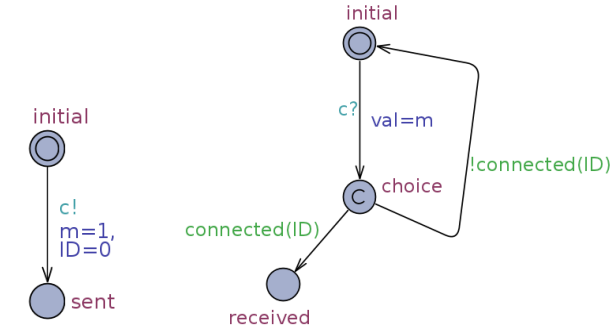


Figure 1: Network connectivity graph

4.1 Modeling methods

1. In the first modeling method, the connectivity is checked after the synchronization: this is the method described in [22] and [20]. A broadcast channel is used. The sender synchronizes all the nodes of the network and updates a global variable with its identification number and a global variable that corresponds to the information of the message. All the nodes then check if they are neighbors of the sender, if it is the case they save the information, otherwise they ignore it.

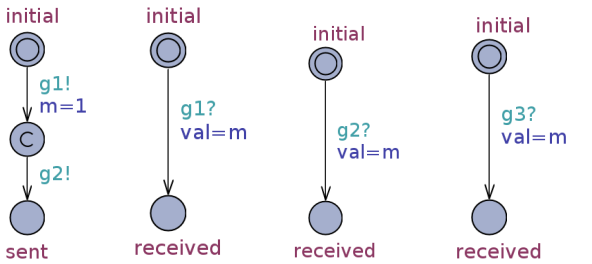


(a) source (b) node

Figure 2: Model of case 1

The TA of the source and the TA of the nodes for this case are depicted respectively on Figure 2a and Figure 2b. The identification number of the source is stored in the global variable ID. The nodes synchronize with the source and store the value of m in the local variable val. Then each node checks if the it is directly connected to the source in order to know if the message can actually be received. This is done by the $connected(ID)$ function which checks connectivity in a two-dimensional boolean array.

2. The second method defines one broadcast channel per group (or maximal cliques): we keep the idea of groups from [11] but we choose to use one broadcast channel per group. Thus the model has to be produced in function of the connectivity graph of the network, the TA of each node depends on which channel it synchronizes i.e., to which group it belongs.



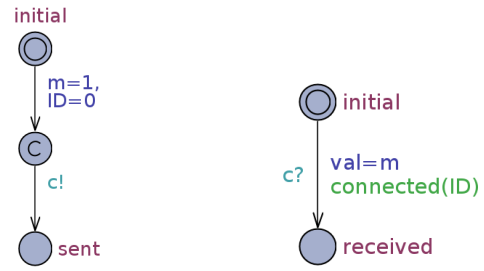
(a) source (b) group1 node (c) group2 node (d) group3 node

Figure 3: Model of case 2

Nodes belong to groups which are maximal cliques. In the example network there are 3 maximal cliques: $g1 = (s, n2, n3)$,

$g2 = (s, n1)$ and $g3 = (n2, n4)$. The TA of the source, the TA of $g1$ nodes, the TA of $g2$ nodes and the TA of $g3$ nodes are depicted respectively on Figure 3a, Figure 3b, Figure 3c and Figure 3d. To each group corresponds a broadcast channel. The source synchronizes only on the channels which correspond to the groups it belongs to (in this example $g1$ and $g2$). The m value is updated only once since it has to be the same for all the receiver. The identification number of the source does not need to be sent because the connectivity information is embedded in the model in this case.

3. In the third case the connectivity is checked before the synchronization: this solution is similar to the first one but the updates of the sender are done before the synchronization and there is a guard on the synchronization edge of receiving nodes that prevent non connected nodes to synchronize with the sender. It is the method used in [8].



(a) source (b) node

Figure 4: Model of case 3

The TA of the source and the TA of the nodes are depicted respectively on Figure 4a and Figure 4b. This case seems very similar to the first one. But the source sends information before synchronization. Thanks to that information and the guard (the $connected(ID)$ function), only the nodes that are directly connected to the source synchronize.

4.2 Number of accessible configurations

In each case we are interested in the number of configurations induced by the model in order to conclude on which method mitigates the state space explosion. In this part we do not consider time valuations in the configurations since time is not used in the examples. Thus a configuration is a vector of states, one state for each TA of the NTA.

In the first case (Figure 2), the number of accessible configurations is 18: there is *(initial, initial, initial, initial, initial)*, after the synchronization there is *(sent, choice, choice, choice, choice)* and then there is an interleaving between the 4 nodes giving $2^4 = 16$ possibilities (it is a power of 2 because each node can be in *choice* or either in *received* or *initial*). Because of this interleaving the number of configurations induced by the communication is exponential in the number of nodes in the modeled network. In the case of large scale networks such as WSNs this is an issue.

In the second case (Figure 3), the number of accessible configurations is 3: initially, there is *(initial, initial, initial, initial, initial)*, then *(_, initial, received, received, initial)* (_ is the committed location of the source TA) and finally *(sent, received, received, received, initial)*. In this case the number of configurations induced by the communication grows linearly with the number of groups the source belongs to.

In the third case (Figure 4), the number of accessible configurations is 3: first, *(initial, initial, initial, initial, initial)*, then *(_, initial, initial, initial, initial)* and finally *(sent, received, received, received, initial)*. In this case, the number of configuration induced by the communication should always be 3 if there is one sender at a time.

The way a local broadcast communication is modeled influences the number of accessible configurations and thus can mitigate or accentuate the state space explosion problem. When verifying time properties on WSNs this can be an issue so the more efficient modeling technique has to be chosen. This comparison shows that the first case which is proposed in [22] and [20] should be avoided. The two other cases should not induce too large state spaces. Nevertheless we can note that in the second case the connectivity information is embedded directly in the TA (because of the channels on the edges) so mobility (or changes in the topology) seems difficult to model. In the third case the connectivity information is in a two-dimensional array which can easily be modified in order to model mobility.

In the next section we choose a simple WSNs protocol and compare the performances of the UPPAAL model checker when the communications are modeled with the three presented cases. This shows how to use these methods with a real case and confirms the conclusions of the comparison.

5. COMPARISON USING A WSN PROTOCOL

We choose to use the GRAB[23] protocol because it is a simple WSNs protocol so we can focus in the modeling methods more than the modeling of the protocol in itself.

5.1 The GRAB protocol

GRAB is a routing protocol for WSNs. At the initialization of the network, each node retrieves its distance to the sink in hop-count. The forwarding scheme is depicted on Figure 5. Data packets are routed using gradient-routing: when a packet is emitted, the nodes with smaller hop-counts than the sender are potential forwarders. As many nodes with the same hop-count can hear the packet, the selection of the forwarder can be based on the signal strength (so the furthest node from the sender is elected) or on a random value, and multiple forwarders can be elected, creating multiple paths. In this paper we consider the case in which there is only one forwarder. When the first potential forwarder emits the packet it prevents the other to send it a second time.

For each of the cases defined in section 4 we produce a UPPAAL model which is the basis of our tests. We also define a tool that inserts the graph of the network in the UPPAAL model. In the cases where the connectivity is checked after the synchronization and before the synchronization, the graph is stored in a two-dimensional array of boolean. Thus the array has to be written in the global declarations of the UPPAAL model. In the case of the usage of the groups it is more complicated, the graph has to be analyzed in order to find the maximal cliques.

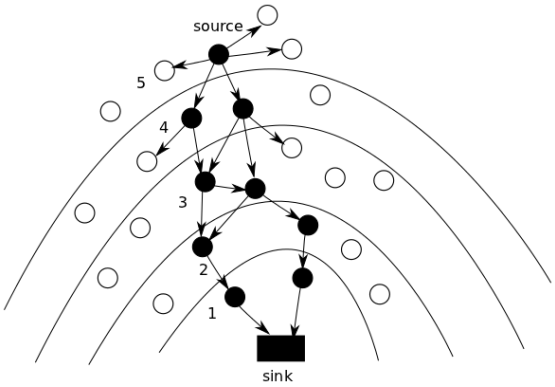


Figure 5: Grab forwarding scheme in the multi-path case. The black nodes forward the packet. The numbers and lines represents the hop-count (the rank).

5.1.1 Case "check after"

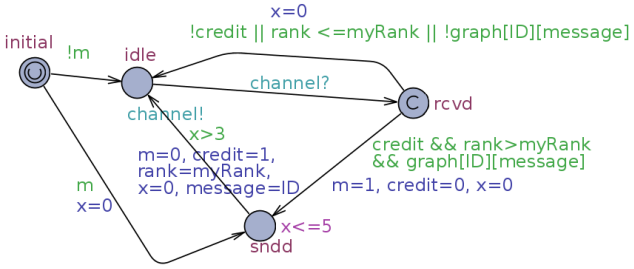


Figure 6: TA in the case the connectivity is checked after

Figure 6 depicts the TA of a node in the case where the connectivity is checked after the synchronization. All the nodes start in the *initial* location. If a node has a message to send it goes to location *sndd*, if it has no packet to send it goes in location *idle* and waits for a synchronization. A node in location *sndd* wait between 3 and 5 time units before taking the synchronization edge (the time for a node to decide to be a forwarder is at least 3 time units and at most 5). When taking the edge it updates several values: the rank of the sender of the message (the rank is the hop-count), the ID of the sender and the credit of the message (the credit corresponds to the possible number of forwarders, in our case it is one). Simultaneously, the nodes that are in *idle* location synchronize by taking the edge to *rcvd* location. If a receiver is connected to the sender, it remains credit and the rank of the receiver is smaller than the rank of the sender, the edge to *sndd* is taken in order to forward the packet. Otherwise it takes the edge to *idle*.

We created a script tool that takes the TA of Figure 6 and the graph of the network. It produces the NTA to be checked with the two-dimensionnal boolean array that represents the network as a global variable.

5.1.2 Case "groups"

Figure 7 presents the basis TA used in the case of the usage of groups. In this case the connectivity is not verified because a node receives the message only if it is in one group of the sender. In fact, the Figure 7 depicts the TA which serves as a basis in order to build the model. Indeed a node

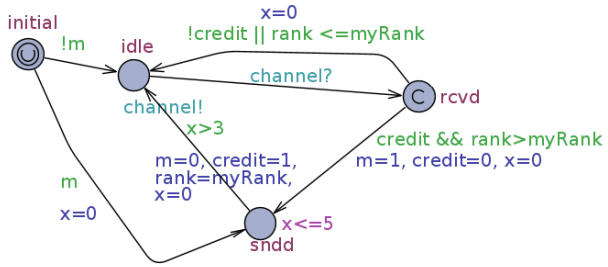


Figure 7: Basis TA in the case of groups

which belongs to several groups has multiple edges from *idle* to *rcvd*, one for each group it belongs to. It has also multiple edges and committed locations between *sndd* and *idle* as in the case of the source in Figure 3a. On Figure 8 an example of node specific TA is given, it corresponds to the *node*2 of the topology of Figure1.

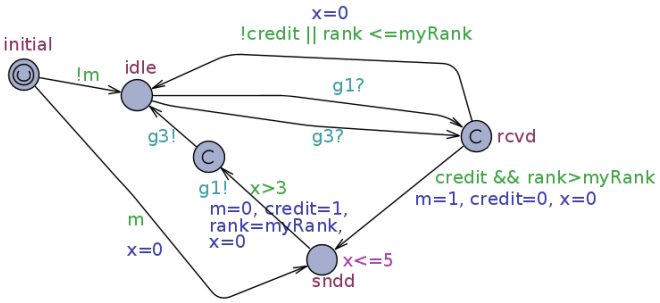


Figure 8: Example of TA in the case of groups

We developed a tool that takes the basis TA and the network graph as input and gives the NTA to be checked as output. The tool retrieves the groups (maximal cliques) from the graph by using the Bron-Kerbosch algorithm [6]. For each node it creates a TA which corresponds to the basis TA with the synchronizations that corresponds to its groups. It adds the produced TA to the NTA to be checked.

5.1.3 Case "check before"

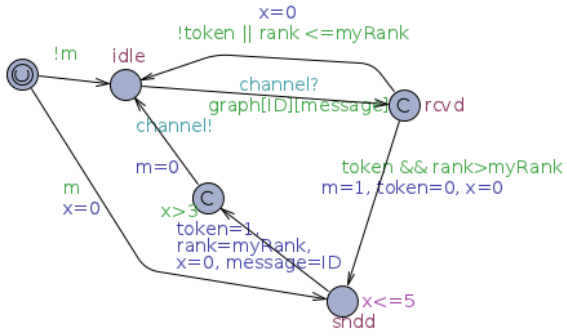


Figure 9: TA in the case the connectivity is checked before

Figure 9 depicts the TA of a node in the case where the connectivity is checked before the synchronization. A node

that has a message to send can access the location *sndd*, it can stay there between 3 and 5 time units. Then it takes the transition to the committed place. It updates the global variables of the message. Finally, it synchronizes by taking the edge to *idle* location. Nodes that are in *idle* location synchronize only if they are connected with the sender.

In this case the tool which produces the NTA is the same as in the case the connectivity is checked after the synchronization.

5.2 Performances and discussion

5.2.1 *Performances*

In this section we present results on the efficiency of the model checking using the different aforementioned approaches. We are interested in the size of the state space and the duration of the model checking.

The NTAs produced by the tools presented in the previous section are checked with UPPAAL. The graphs used in order to realize the tests are geographic random connected graphs (nodes are placed randomly on a plane and there is a link between two nodes if they are in each others radio range). The number of nodes varies from 6 to 21 (including the sink node) and for each number of nodes we perform the model checking on 10 random graphs. For each model checking, only one node has a message to send. The node is picked among the nodes which have the highest hop-count.

We want the model checker to generate, store and explore the maximum number of states in order to compare the worst case of each solution. We thus choose to check a safety time property, so it should be true in all states of the model. UPPAAL uses on-the-fly method, so if the property is verified in all states it has to explore all the states. If it is false in one state the model checking ends when this state is reached even if it remains unexplored states. We check the property:

$$\texttt{A[]}\ \texttt{n0.m}\ \texttt{imply}\ (\texttt{z} > \texttt{N} * 3\ \texttt{and}\ \texttt{z} \leq (\texttt{N}+1) * 5)$$

`N` is the rank of the source, `n0.m` signifies that the sink has received a message and `z` is a clock. The values 3 and 5 come from the model, indeed a node wait between 3 and 5 time units before sending its message. This property means that the packet must arrive at the sink during a defined time window and thus that the WCTT must be bounded.

Figure 10 presents the average state space size with 95% confidence interval in function of the node number in the graph for the 3 solutions. The scale of the ordinate is logarithmic thus the curves are exponential. From section 4 we deduce that the case where the connectivity is checked after should induce exponential growth of the state space whereas in the two other cases it should be constant or linear with the number of groups the sender belongs to. Here it is exponential in all cases, but it is not only due to the modeling of the local broadcast behavior. The transition between the *initial* state and the *idle* or *sndd* state produces interleaving, so do the choice of only one forwarder among receivers of the packet.

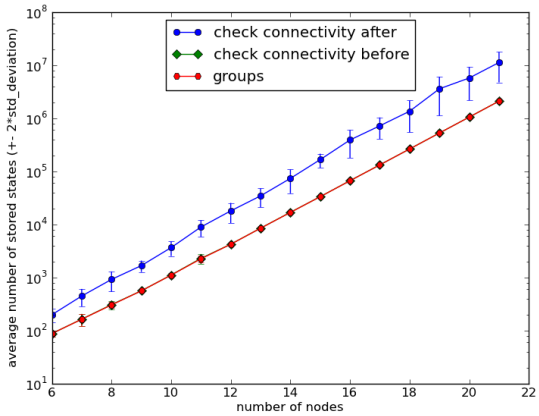


Figure 10: Average number of states stored during the model checking process in function of the number of nodes

Anyway, the solution which leads to the largest state space size is the case where the connectivity is checked after the synchronization. This is because of the interleaving induced by the verification of connectivity by the potential receivers as we showed in section 4. The two other cases have similar performances, the curves are nearly perfectly superposed. In section 4 we said that the case of groups should induce a little more states than the case where the connectivity is checked before if the nodes belong to several groups. Nevertheless the difference is so small that it cannot be seen on Figure10. The case where the connectivity is checked after has a large confidence interval, nevertheless, it does not overlap with the ones of the other solutions. They can thus be considered to be better.

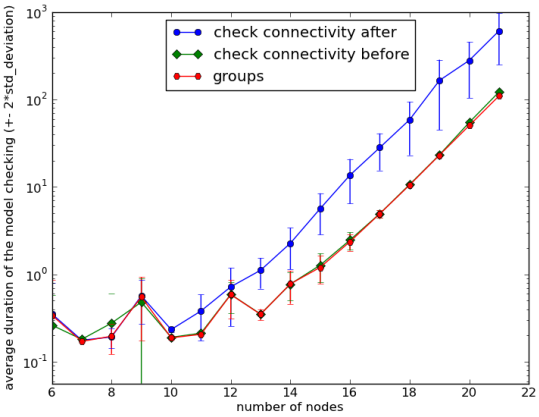


Figure 11: Average duration of the model checking process in function of the number of nodes

Figure 11 represents the average duration of model checking with 95% confidence interval in function of the number of nodes in the graph. In this case the scale of the ordi-

nate is also logarithmic. The results under 1 second are not strictly monotonic because the impact of potential executions of other processes during the model checking. Those executions have less impact on longer model checking durations. The forms and differences among curves for each methods are very similar to those of Figure10. This is because the more state to be stored and explored, the longer the model checking process.

5.2.2 Discussion

The results of the tests we perform for the different solutions with GRAB confirm the results of the comparison of section 4. The case where the connectivity is checked after the synchronization leads to state explosion with a smaller model than the other cases. It should thus be avoided when modeling large scale systems such as WSNs. The two other cases have equivalent performances in terms of state space and duration of execution of the model checking. Nevertheless the solution where the connectivity is checked before the synchronization allows to model mobility (or more generally changes in the topology) whereas the groups solution leads to model check a static network. Moreover the tool which constructs the NTA in the case of the groups is more complex. Thus the solution where the connectivity is checked before should be used in order to model the local broadcast behavior of WSNs.

6. CONCLUSIONS AND FUTURE WORKS

In this paper we are interested in the efficient modeling of local broadcast behavior of WSNs for the model checking of time properties (mainly WCTT bounds). We present three ways of modeling local broadcast that comes from the literature or are adaptations or improvements of existing solutions. We compare the solutions by using a simple example. The solution which checks the connectivity after the synchronization leads to a larger state space than the solution which relies on maximal cliques and the one which checks the connectivity before the synchronization. We then compare the solutions with a simplified version of GRAB protocol. We give a method to automatize the integration of the connectivity information in the NTA model for each solution. The results with the WSN routing protocol confirm the results of the first comparison. It shows that, when modeling WSNs, the issue of local broadcast has to be carefully considered in order to mitigate state space explosion problems. The solution which checks the connectivity before the synchronization should be preferred. Because it induces as few states as the group solution, but allows to model mobility or dynamicity of the topology, and induces lighter treatments for the production of the NTA.

In the future we plan to use the local broadcast modeling presented in this paper as a basis to represent wireless communications more accurately. We will add the probabilistic aspect of the radio link and the dynamicity of the topology. We also plan to model and verify more realistic WSNs protocols on larger topologies.

7. REFERENCES

[1] R. Alur and D. Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[2] F. V. B. Berthomieu, P.-O. Ribet. The tool tina – construction of abstract state spaces for petri nets and time petri nets. *International Journal of Production Research*, 42(14), 2004.

[3] J. Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335(2-3):131–146, 2005.

[4] C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[5] B. Bérard, F. Cassez, S. Haddad, D. Lime, and O. Roux. Comparison of the expressiveness of timed automata and time petri nets. *Formal Modeling and Analysis of Timed Systems*, pages 211–225, 2005.

[6] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, 1973.

[7] E. Clarke and J. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys (CSUR)*, 28(4):626–643, 1996.

[8] A. Fehnker, L. Van Hoesel, and A. Mader. Modelling and verification of the lmac protocol for wireless sensor networks. IFM'07, pages 253–272, Oxford, UK, 2007.

[9] G. Gardey, D. Lime, M. Magnin, and O. Roux. Roméo: A tool for analyzing time petri nets. CAV'05, Edinburgh, Scotland, UK, 2005.

[10] A. D. Gerd Behrmann and K. Larsen. A tutorial on uppaal. In *Formal Methods for the Design of Real-Time Systems*, volume 3185 of *Lecture Notes in Computer Science*, pages 33–35. 2004.

[11] J. C. Godskesen and O. Gryn. Modelling and verification of security protocols for ad hoc networks using uppaal. 18th Nordic Workshop on Programming Theory, 2006.

[12] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:394–406, 1992.

[13] G. Holzmann. The spin model checker, primer and reference manual. 2003.

[14] R. Milner. Communication and concurrency. In *Series in Computer Science. Prentice-Hall International*, 1989.

[15] L. Mounier, L. Samper, and W. Znaidi. Worst-case lifetime computation of a wireless sensor network by model-checking. PE-WASUN '07, pages 1–8, Crete Island, Greece, 2007.

[16] P. Olveczky and S. Thorvaldsen. Formal modeling and analysis of wireless sensor network algorithms in Real-Time Maude. *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, 2006.

[17] A. P. Ravn, J. Srba, and S. Vighio. A formal analysis of the web services atomic transaction protocol with uppaal. ISoLA'10, pages 579–593, Heraklion, Greece, 2010.

[18] A. Singh, C. Ramakrishnan, and S. A. Smolka. A process calculus for mobile ad hoc networks. *Science of Computer Programming*, 75(6):440 – 469, 2010.

[19] R. Tan, G. Xing, J. Chen, W.-Z. Song, and R. Huang. Quality-driven volcanic earthquake detection using wireless sensor networks. RTSS '10, San Diego, CA, USA, 2010.

[20] S. Tschirner, L. Xuedong, and W. Yi. Model-based validation of qos properties of biomedical sensor networks. EMSOFT '08, pages 69–78, Atlanta, USA,

2008.

[21] Y. Wang. Real-time behaviour of asynchronous agents. *CONCUR '90 Theories of Concurrency: Unification and Extension*, 458:502–520, 1990.

[22] O. Wibling, J. Parrow, and A. Pears. Automatized Verification of Ad Hoc Routing Protocols. pages 343–358, 2004.

[23] F. Ye, G. Zhong, S. Lu, and L. Zhang. GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks. *Wireless Networks*, 11(3):285–298, 2005.