Wireless Personal Communications (2020) 114:2855–2888
https://doi.org/10.1007/s11277-020-07507-8

# On Formal Modeling and Validation of Wireless Sensor Network Protocols

Rachid Bechar[1] · Mounir Tahar Abbes[1] · Freha Mezzoudj[1] · Ladjel Bellatreche[2]

**Abstract**
Formal verification is becoming more and more important in the field of wireless networks (WSN). The general purpose formal method called Event-B is the latest incarnation of the B Method: it is a proof based approach with a formal notation and refinement technique for modeling and verifying systems. Refinement enables implementation level features to be proven correct with respect to an abstract specification of the system. This paper proposes an initial attempt to model and verify consistency and correctness of a WSN operation in its different layers. Several formal models are introduced for this type of networks. In the first time, coloured Petri net are used to elaborate network layer models, then each one will be detailed by an Event-B formalism, while proofs are carried out using the RODIN platform which is an integrated development framework for Event-B.

**Keywords** WSN · Event-B method · CPN · Formal methods · Validation

## 1 Introduction

Wireless sensor networks (WSN) have a big number of application fields. They are used mainly for environmental sensing, industrial monitoring, target tracking, assisted living and recently internet of things [1]. As a networked system, WSN application needs sensor node as hardware part, operating systems, some networking protocols and the application itself often uses other technologies and techniques such as Ad hoc networks [2], multi-agent systems [3] and data processing [4]. So, its performance depends on all of these factors. Before deploying

✉ Rachid Bechar
bechar.rachid@gmail.com

Mounir Tahar Abbes
mtaharabbes@yahoo.fr

Freha Mezzoudj
f.mezzoudj@univ-chlef.dz

Ladjel Bellatreche
bellatreche@ensma.fr

1 Department of Computer Science, FSEI, Hassiba Ben Bouali Uiversity of Chlef, Ouled Fares, Algeria

2 ISAE – ENSMA, University of Poitier, Poitier, France

and installing a WSN, it is imperative to evaluate performances of the new system. There are several methods which can be used for WSN protocols evaluation and validation. The testbed method consists to produce a prototype with a real experimentation, it is the most realistic and complete method but the most costly. For this reason, several works on WSN especially in the academic field use simulation method where protocols are implemented and tested in various simulators like NS2/3 [5], Omnetpp/Castalia [6, 7], Glomosim [8] and J-Sim [9]. However, it is not easy to obtain solid conclusions from a simulation study. This is due to the suitability of a special tool of simulation and correctness of the used models. In addition, using simulators requires an important effort and makes an important time processing. The third way to evaluate performance is emulation which is a compromise between simulation and real testbed. Emulation combines between elements of real tests and some assumptions obtained by simulation. It generally has realistic parameters which are directly incorporated by software into the architecture being used [10]. So, real and virtual nodes can be used in the same time. As WSN emulator, there is SPSim, EmPro, EmStar, SunSpot Manager and FreeMote emulator. To sum up, protocol verification and validation using simulations or development of prototypes is sometimes difficult, requiring experience in software development, hardware configuration and a considerable working time. Finally, it is possible to use formal methods for protocol validation. These methods are based on mathematic formulation and analyze concluding to obtain exact and proved results. Formal validation needs two fundamental steps, beginning with modeling part which is the most delicate task in this approach. The second step is to study and analyze the system characteristics and performances through the selected model. The most used tools in this field are Markov models [11–13], probabilistic and logical modeling and analysis [42], Event B-method [14–18] and Petri nets [19, 20]. This paper proposes to validate WSN energetic model using some formal methods where the system will be described using an Event B-method formulation and the system operation will be modeled by a coloured Petri net. The work consists in defining and modeling each layer in the protocol stack using Petri nets passing by formal specifications and verifications in order to study and validate the quality of protocols managing message routing and energy consumption in a WSN. The main objective is to analyze how the system will perform. So, it is possible to correct, enhance and validate the system operation before deploying it in service. The paper is organized as follows: in Sect. 2, some preliminary notions are introduced such as Event B-method and Petri nets. Section 3 presents related works which are based on these formal tools to validate WSN protocols. Section 4 discusses the proposed model for WSN formal validation. Results and analysis of this proposition are discussed in Sect. 5 and finally, Sect. 6 presents conclusions and future work.

## 2 Preliminaries

Before giving details on formal validation of WSN energetic model, it is important to recall some fundamental notions related to the formal tools used for the proposed modeling. These tools are mainly the Event- B method, Petri nets and coloured Petri nets.

## 2.1 Event B Method and Program Proving

The event-B [14, 16] is a software development language based on B method, a tool-supported formal method based on an abstract machine notation; it can be used to model all sorts of discrete event systems, among them sequential programs [15].

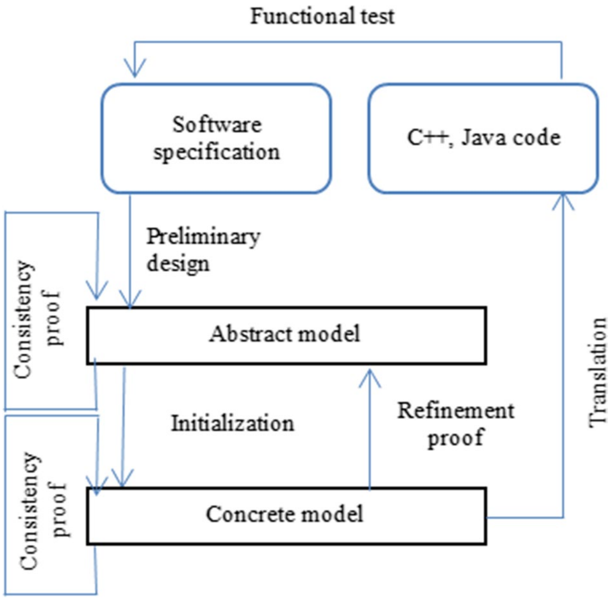
System developing with B-method proposes a cycle shown in Fig. 1 and passing by the following steps:

1. Translation of informal system requires a specification into the abstract model or the abstract machine notation.
2. Refinements through a sequence of detailed versions that must be proved to be consistent with the previous one.
3. Possibility of translation into some classical programming language such as C++ and Java.

An event-B project has the following components:

1. Context: representing the static structure of a discrete system, it is a sort of first-order theory that contains declaration of constants and axioms about these constants using an abstract language based on predicates of first order logic, simple sets theory and arithmetic on Z; set of integers. For example, the context for searching a value in an array of strictly positive integers can be described as:

**Fig. 1** B-method methodology



```
CONTEXT Array
CONSTANTS
    n array size
    T the array
    v value to search
AXIOMS
    axm1:  $n \in \mathbb{N}_1$
    axm2:  $T \in 1\,..\,n \rightarrow \mathbb{Z}$
    axm3:  $v \in \mathbb{Z}$
END
```

2. Abstract machine: it describes the dynamic structure or the system evolution using constants and axioms which are imported from context by the SEES clause. The abstract machine must describe:

   – The system's state through a set of variables;
   – The consistency of the state by a set of invariants (some formulas to satisfy);
   – Some events to define the possible evolutions of the machine's state.

   The abstract machine associated to the previous example can be written as:

```
MACHINE S1
SEES Array
VARIABLES
    result1
    Elmt
INVARIANTS
    inv1:  $result1 \in \mathbb{Z}$
    inv2:  $Elmt \in \mathbb{Z}$
EVENTS
Initialisation
    begin
        act1: $result1 := 1$
        act2: $Elmt := 0$
    end
Event evt1 ⟨ordinary⟩ $\widehat{=}$
    when
        grd1:  $n \geq 10$
    then
        act1: $Elmt := 2$
    end
END
```

There are some tools which can be used for Event-B development. The most known is Rodin [17, 18], an open source platform based on Eclipse IDE for Event-B that provides effective support for refinement and mathematical proof of system description.

## 2.2 Coloured Petri Net

In general, a petri net PN [19] is a famous mathematical model for distributed and event discrete systems. This formal tool can be used to describe concurrent and synchronous activities. Graphically, a PN is represented by a directed bipartite graph in which vertexes represent transitions (modeling events that may occur and are represented by bars) and places (modeling conditions that are represented by circles). As all graphs, a PN is represented by an incidence matrix and a marking vector. So, the system evolution can be obtained simply by multiplication of the matrix and the marking vector of each step. This formulation allows studying several qualitative system proprieties like reachability, boundedness and liveness. To improve its power of analysis, there are several extensions of PN such as stochastic PN, timed PN and coloured PN. The last one will be used in this paper. Coloured Petri Nets (or CPNs) is a language mainly used for modeling and validation of concurrent and distributed systems in which concurrency, synchronization, and communication play a major role [20]. CPN models are formal and hence, they can be used to prove properties about the modeled systems. This is done by model checking based on state space exploration. The main characteristic of CPN is the possibility to have different types of marks or tokens specified by different colours values in each place. More interesting, several types of transitions can be expressed, depending on the combinations of coloured tokens. Formally, a CPN is defined by a tuple [21, 22]:

$$CPN = (S, P, T, A, N, C, G, E, I) \tag{1}$$

where:

- S is a finite set of non empty types, called color sets.
- P is a set of n places P = {P1, P2,..., Pn}.
- T is a set of m transitions T = {T1, T2,..., Tm}.
- A is a finite set of directed arcs relating places to transitions or transitions to places, so:

$$A \subseteq (P) \cup (T) \tag{2}$$

- N is a node function defined from A into $(P) \cup (T)$.

- C is a color function defined from P into S.
- G is a guard function defined from T into expressions such as:

$$\forall t \in T\ :\ [Type(G(t)) = Boolean \wedge Type(Var(G(t))] \subseteq S \tag{3}$$

- E is an arc expression function defined from A into expressions such that:

$$\forall t \in T\ :\ [Type(E(a)) = C(p(a)_{MS}) \wedge Type(Var(E(a))] \subseteq S \tag{4}$$

  where p(a) is the place of N(a).

- I is the initialization function from L into closed expressions such that:

$$\forall p \in P\ :\ [Type(I(p)) = C(p)_{MS}] \tag{5}$$

In (4) and (5) the MS notation signifies a multi sets type. In practice, there are some tools used to study and evaluate CPN models such as CPN-tools [23] and TransCPN [24].

## 3 Related Works

### 3.1 Works Using Event-B

A detailed explanation of Event-B modeling and semantics of each part is found in [43]. Authors give a case study in this context and propose a stepwise correct-by-construction model to build asynchronous distributed systems which a priori realize their choreographies by relying on a necessary realizability condition and then applying several refinement steps to generate the distributed peers of the network.

In [44], a fairness based method is proposed by integrating temporal and first order logic with event-B/Rodin proof for modeling and verification of safety and liveness properties in distributed systems. More exactly, they use the temporal logic for actions applied on three examples of protocols as a theoretical framework for computability reasoning about WSN in order to extract certain repeating patterns in the proofs which can be easily identified and reused for similar modeling.

An Event-B formal verification model is used in [25] for checking WSN-ZigBee protocol stack in a layered manner, where protocol semantics are well defined to be embedded in the Event-B language. In that model, each layer with its attributes is modeled in Event-B machine then, different events like initialization events are used to model interfaces between layers, where Event-B guards are used to define preconditions for those events. Finally, for modeling correctness properties of the protocol, author uses Event-B invariants and proof obligations.

In [26], authors propose a formal co-simulation framework for WSN. The goal of that work is to provide a high level abstraction for software engineering practice in WSN by combining existing simulation and proof-based formal verification approaches. So, they use the refinement method of the Event-B and its Rodin toolkit to model and verify the sensor operation from application to MAC layer in a layered and formal manner. In the other hand, MiXiM simulations are used to provide full confidence about reliability and performance analysis of physical environment.

The work in [27] is focused on wireless sensor–actor networks, where both ordinary sensor nodes and more sophisticated and powerful nodes, called actors, are present. In that paper, authors introduce several formal models for that type of wireless networks. Those models formulas are recently introduced with an algorithm for recovering actor–actor coordination links via the existing sensor infrastructure. They prove via refinement that that recovery is correct and that it terminates in a finite number of steps. In addition, they propose a generalization of the formal development strategy, which can be reused in the context of a wider class of networks. They elaborate their models within the Event-B formalism, while their proofs are carried out using the RODIN platform. They formally model a distributed recovery algorithm for wireless sensor–actor networks. They develop the model in four increasing levels of abstraction that refine each other. They prove the correctness and successful termination of the algorithm, tool-assisted. They put forward three types of coordination links in wireless sensor–actor networks. They finally generalize the formal model to a wider class of networks via refinement patterns.

Modeling distributed algorithms for mobile Ad-hoc networks and looking at their assumptions is the subject of [28]. Proving the correctness of these algorithms for dynamic networks is a topic of intensive research. In fact, the solutions which have been proposed previously to express and prove the correctness of distributed algorithms are usually done manually. In addition, all these solutions lack a consensus about their development and their proof. The main contribution of that paper is to propose a general and formal model for dynamic networks based on evolving graphs and Event-B formal method. In fact, evolving graphs is a powerful tool to express fine-grained properties. That model allows to handle topological events and to characterize the concept of time with some particularities. It is implemented with Event-B, based on refinement technique. The proposed model is illustrated by an example of a distributed algorithm encoded by local computations models.

An other work on ZigBee protocol is found in [29] but it focused on secure network authentification. Like the previous work, the Zigbee protocol behavior is described by an Event-B abstract machine adding key management mechanism and other security parameters for context and axioms part to obtain the complete model which is analyzed in Rodin platform.

In [30], the proactive routing protocol OLSR is modeled and analyzed based on event-B and the Rodin theorem proving platform. The protocol complexity is defined by five distinct abstraction layers which are linked to each other by refinement. This approach can be used as a proof-of-concept to be adapted to model and verifying other routing protocols used in large-scale WSN.

In [31], authors formally analyze the zone routing protocol (ZRP), a hybrid routing framework, using Event-B. Since Ad hoc routing protocols are responsible for searching a route from the source to the destination under the dynamic network topology. Hybrid routing protocols combine the features of proactive and reactive approaches. So, the formal specification of a hybrid routing protocol in the dynamic network environment is a challenge. The authors develop the formal specification by the refinement mechanism. It allows them to gradually model the network environment, the construction of routing zones, route discovery based on border casting service and routing update. They prove the stabilization property in the inactive environment. In addition, they demonstrate that discovered routes hold the loop freedom and validity in each reachable system state. To show that the formalization is consistent with the informally expressed requirements, they adopt an animator, ProB, to validate that model. That work provides reference to analyze extensions of the ZRP and other hybrid routing protocols.

## 3.2 Works Using CPN

Petri nets are widely used to model and verify concurrent and distributed systems functionalities especially in WSN. Several works on formal methods for WSN modeling and validation are interested by the MAC layer like [32] that uses the hierarchical modeling capability of CPN for modeling and evaluating the performance of the S-MAC protocol where it defines a CPN level for wireless channel, node scheduling, neighbors scheduling and message control parts.

The work in [33] presents a CPN based model which allows verifying and validating the design properties and structural behavior of wireless sensor and actuator networks. It is a computation model based on predefined components and CPN specifications giving possibility to generate the equivalent code in C language. The system components can perform two types of activities, periodic ones running with constant time intervals and aperiodic ones which are initiated by the occurrence of external events. For interaction between activities, authors use specific scheduler and timing components.

Authors of [34] are interested in a kind of intelligent wireless sensor networks (IWSN) which is set up from the point of view of multi-agent systems. IWSN is composed of some sensor node agents, some cluster head agents, a base agent and a command agent. According to the characteristics of these agents, intelligent wireless sensor networks model (IWSNM) based on Petri nets is proposed, which can accurately and unambiguously model the overall and individual characteristics of the networks. Moreover, IWSNM can be analyzed, verified and validated by the supporting tools of Petri nets. Consequently, the defects in early design stage can be detected, and the security and reliability can be improved. Finally, IWSNM are employed to develop the target tracking systems.

For role based routing protocol in WSN, [35] presents a formal study for the Network rOle-based Routing Algorithm (NORA) protocol using CPN as a modeling language to obtain unambiguous and complete specifications of system behavior. It begins by writing a pseudo-code describing the protocol then they construct the corresponding CPN that shows all possible states of node, its role and activities with necessary parameters. CPN-Tools have been used to evaluate and study correctness using state space exploration.

Because nesC is the most used programming language for implementing WSN applications, authors of [36] present a tool that generates power consumption models as CPN from the nesC code of the WSN application. That approach uses three levels to define CPN models, basic models representing operators and statements in the application code. Next, these basic models are composed into function models to express the power consumption of commands and events and finally the entire WSN application is obtained by the composition of all function models. So, it is possible to evaluate the power consumption by determining consumption of each level in that hierarchical CPN.

In [37], authors use a components oriented model and the expressiveness of Coloured Petri Nets to model and estimate network's energy consumption. Each particular component of a sensor is modeled alone then it is interfaced to other components to obtain the global behavior. The most interesting part in that work is radio and MAC behavior.

In [38], the WSN-PN tool is proposed, it allows the congestion detection on a WSN setting. A recent work in [39] concerns modeling of MAC protocols using CPN for platform independent modeling, initial verification and simulation. Platform-specific implementations for multiple platforms are generated by the Petri Code tool, including MiXiM for simulation and TinyOS for deployment.

An other technique is proposed in [40] for verifying congestion probability on WSN which is modeled by CPN. That work consists to attach reliable probability on each node allowing knowing the probability of reaching the sink according to the network topology. These probabilities combined to routing probabilities are used to transform that topology into a discrete time stochastic CPN. The paper gives some analyzes by tracing the state space of the CPN model. Finally, it estimates that approach is more observable, scalable, and portable than Markov model.

In [41], authors propose a modeling and performance analysis for IEEE 802.15.4 which is the standard protocol for low-rate, low-power wireless personal area networks, including the physical layer and media access control layer specifications. Particularly, some mechanisms for certain purposes, such as transmission efficiency and power saving, are introduced. It is worthwhile to evaluate how the protocol can fulfill the quality of service requirements. Coloured Petri nets are chosen for the modeling and analyzing purposes because of the enhanced modeling power and abundant analysis techniques. The modeling method for wireless network protocols using CPN is summarized, which is general enough and can be directly used to model various wireless network protocols like IEEE 802.15.4. During the modeling process, some modeling techniques are utilized, e.g., building the model in the hierarchical and modular way, reducing the model by the folding technique, and normalizing the modeling process by the modeling patterns. Simulation is conducted on the CPN model to compute some performance metrics such as throughput, delivery ratio, delay, and energy cost. All the above show that coloured Petri nets can play an important role in modeling and analyzing wireless network protocols.

It is to note that related works cited here are based on Event-B modeling for the first part and sometimes CPN modeling combined to some language notations such as C or Nesc code. However, there is not works that try to represent the CPN model corresponding to an Event-B model. So, a solution in this context is given in details in this paper.

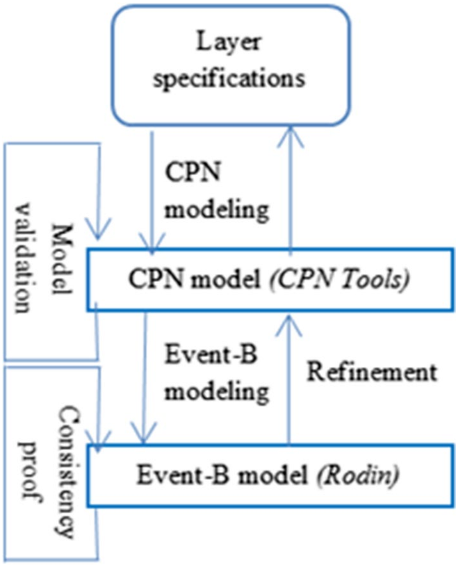
## 4 Proposed Model for WSN Formal Validation

In order to model the proposed solution on formal validation in WSN, an event-B definition is created for each layer in the network according to the OSI model, especially the application, transport, network and MAC layers. In this work, correctness of each layer has been studied in two steps like it is summarized in Fig. 2.

1. In the first step, general mechanism of concerned layer has been modeled by a CPN using the CPNTools environment [23] with verification.
2. In the second step, the Rodin platform has been used to model and proof correctness of network by detailing the corresponding event-B model for each layer.

The diagram of Fig. 3 shows a general idea on aspects to be treated and modeled for each layer in this work.

**Fig. 2** Flow diagram of modeling and proof



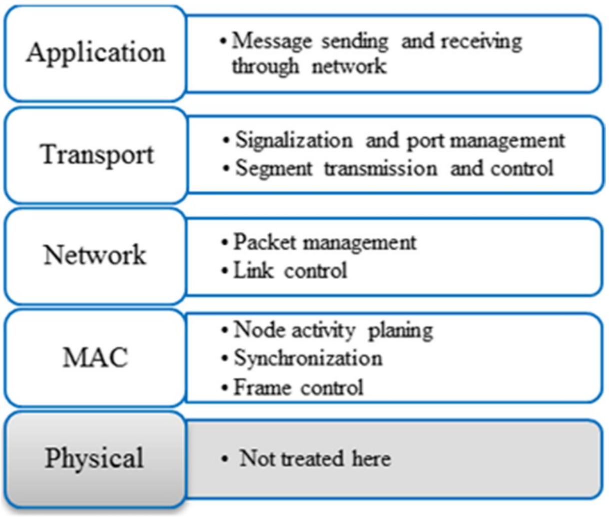


**Fig. 3** General aspects concerned by the modeling

This layered and encapsulated principle is similar in formal methods such as event-B, which is based on refinement as illustrated in Figs. 2 and 3. The refinement allows to build a model gradually by making it more and more precise, so that it is closer to the reality. So, it is not about building a single model representing once and for all the reality; this is clearly impossible due to the size of the state space and the number of its transitions. It would also make the resulting model very difficult to master. In theory there are no precise limit while making refinements of a model that is very safe and rigorous, that is why CPN are used here to get a readable number of states and visible for readers. If there is a large number of states in a single module refined several times, there is a need for module decomposition, then a threshold is fixed in this case.

## 4.1 Application Layer

Application layer is the highest level of a network operation where details of transmission are not treated. At this level, sending and receiving messages between nodes through a network are modeled without mentioning other parameters to specify how and when do something. So, the CPN of Fig. 4 explains what do the application layer.

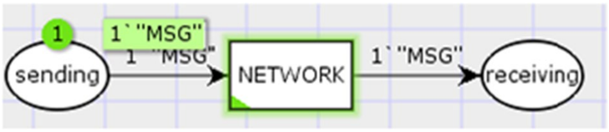


**Fig. 4** CPN model for application layer

In a second time, an event-B model is created for checking correctness of network operation at this level. Such model contains two parts, context and machine. The static part of application layer is defined by the following context named context_app written in event-B as:

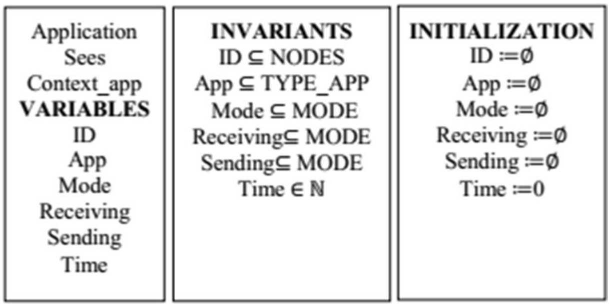
```
CONTEXT context_app
SETS
    NODES
    TYPE_APP
    MODE
AXIOMS
    MODE=sending,receiving: ⊤
END
```

This context describes the WSN application by a set of nodes, application type and a set of modes of each node in the SETS closure. So, a node can be in one of sending or receiving modes which is shown by the AXIOMS closure in the previous context. The dynamic part of a WSN application is defined by an event-B machine named application for which sets of variables, invariants and events must be given like in Fig. 5.

The first part shown at left in the previous model defines node parameters which are its identifier ID, application type installed on the node app, different modes or status of a node noted, receiving and sending. Finally the variable time represents the time counter initialized to zero. The rest parts INVARIANT and INITIALISATION closures describe respectively possible and initial values of variables.

The effective operation or dynamic comportment of nodes is managed by a set of events. For each event, it is necessary to verify some conditions in the form of guards noted grdi then the list of possible actions to perform by the node noted actj according to each situation. For example, in the event SEND, there is grd1 and grd2 used for verifying ID and time of the concerned node. If conditions of guards are verified, node must permit its mode to SEND and increment its time counter.

**Fig. 5** Application layer machine description



| | | |
|---|---|---|
| Application<br>Sees<br>Context_app<br>**VARIABLES**<br>ID<br>App<br>Mode<br>Receiving<br>Sending<br>Time | **INVARIANTS**<br>ID ⊆ NODES<br>App ⊆ TYPE_APP<br>Mode ⊆ MODE<br>Receiving⊆ MODE<br>Sending⊆ MODE<br>Time ∈ ℕ | **INITIALIZATION**<br>ID :=∅<br>App :=∅<br>Mode :=∅<br>Receiving :=∅<br>Sending :=∅<br>Time :=0 |

```
MACHINE Application
    ......

EVENTS
Initialisation
    then
        act1: $ID := \varnothing$
        act2: $app := \varnothing$
        act4: $mode := \varnothing$
        act6: $IDLE := \varnothing$
        act9: $power\_off := \varnothing$
        act10: $receiving := \varnothing$
        act13: $sending := \varnothing$
        act14: $time := 1$
        act15: $on := \varnothing$
        act16: $pas := 1$
    end
Event SEND ⟨ordinary⟩ $\widehat{=}$
    when
        grd1: ⟨theorem⟩ $(ID \subseteq NODES) \land (app \subseteq TYPE\_APP)$
        grd2: $pas > 1$
    then
        act1: $mode := sending$
        act3: $time := pas + 2$
    end
Event RECEIVE ⟨ordinary⟩ $\widehat{=}$
    when
        grd1: $(ID \subseteq NODES) \land (app \subseteq TYPE\_APP)$
    then
        act1: $mode := receiving$
        act3: $time := time + 1$
    end
END
```

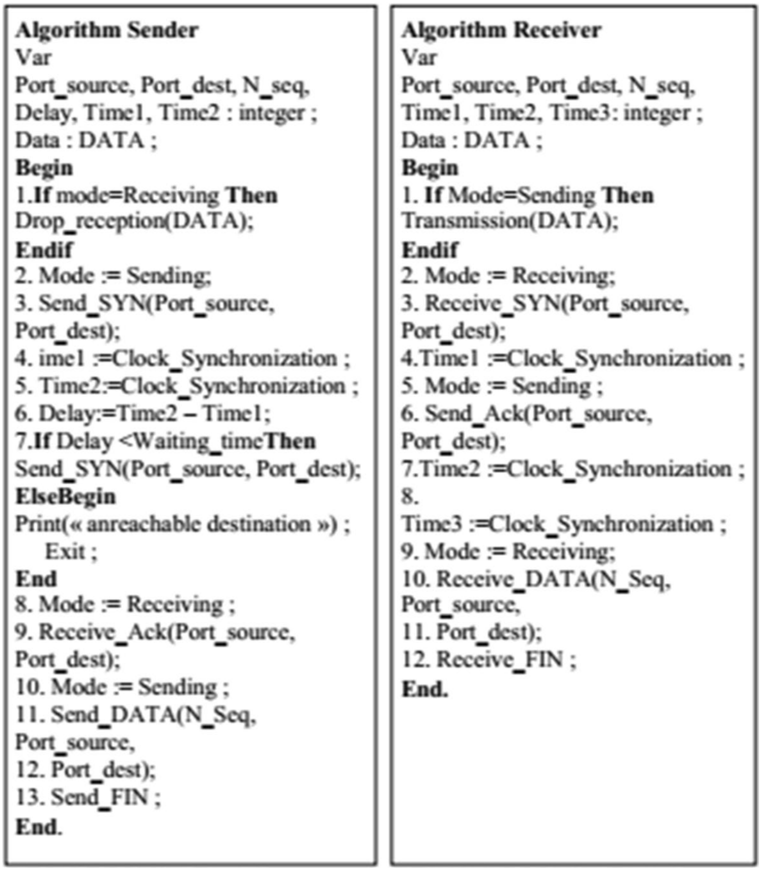
## 4.2 Transport Layer

The main role of transport layer is to establish and terminate connection trough source and destination ports of corresponding nodes. Ports are used to distinguish between applications on the same node. In addition to that, the transport layer is responsible to guaranty deliverance of segments that is noted DATA basing on sequence numbers n_Seq and their acknowledgment noted Ack. In order to well understand what happen in the transport layer between a pair of sensor nodes, it is written as two algorithms, one for each side like it is shown in Fig. 6.

The CPN of Fig. 7 models this principle by detailing states and actions to do according to some conditions by the sender and the receiver nodes as well as intermediate nodes of network.

To establish the corresponding event-B model, the transport layer mechanism is split into two operations:

1. Signalization: to transmit data in the transport layer, the sender must verify that destination node and port exist really in the network, and then it must follow deliverance of messages by acknowledgment.
2. Data: it is the proper action of data transmission when conditions verified in the signalization step are favorable.



```
Algorithm Sender
Var
Port_source, Port_dest, N_seq,
Delay, Time1, Time2 : integer ;
Data : DATA ;
Begin
1.If mode=Receiving Then
Drop_reception(DATA);
Endif
2. Mode := Sending;
3. Send_SYN(Port_source,
Port_dest);
4. ime1 :=Clock_Synchronization ;
5. Time2:=Clock_Synchronization ;
6. Delay:=Time2 – Time1;
7.If Delay <Waiting_timeThen
Send_SYN(Port_source, Port_dest);
ElseBegin
Print(« anreachable destination ») ;
    Exit ;
End
8. Mode := Receiving ;
9. Receive_Ack(Port_source,
Port_dest);
10. Mode := Sending ;
11. Send_DATA(N_Seq,
Port_source,
12. Port_dest);
13. Send_FIN ;
End.
```

```
Algorithm Receiver
Var
Port_source, Port_dest, N_seq,
Time1, Time2, Time3: integer ;
Data : DATA ;
Begin
1. If Mode=Sending Then
Transmission(DATA);
Endif
2. Mode := Receiving;
3. Receive_SYN(Port_source,
Port_dest);
4.Time1 :=Clock_Synchronization ;
5. Mode := Sending ;
6. Send_Ack(Port_source,
Port_dest);
7.Time2 :=Clock_Synchronization ;
8.
Time3 :=Clock_Synchronization ;
9. Mode := Receiving;
10. Receive_DATA(N_Seq,
Port_source,
11. Port_dest);
12. Receive_FIN ;
End.
```

**Fig. 6** Sender and receiver algorithms in transport layer

For the signalization part, the following context is proposed as static part coded in Rodin by:

```
CONTEXT  context_transport
EXTENDS  context_app
SETS
      PORT
      TYPE_MSG
AXIOMS
      axm1:  ⊤
END
```

It is clear from the previous context named context_transport that the transport layer has need to information sent by the application layer. It is the EXTENDS closure which allows to implement this principle in event-B. Indeed, the encapsulation mechanism in a network consists to pass data and parameters from a layer to the inferior one. In addition to the application context, the transport layer uses other parameters integrated by the SETS closure essentially the PORT and TYPE_MSG variables representing respectively

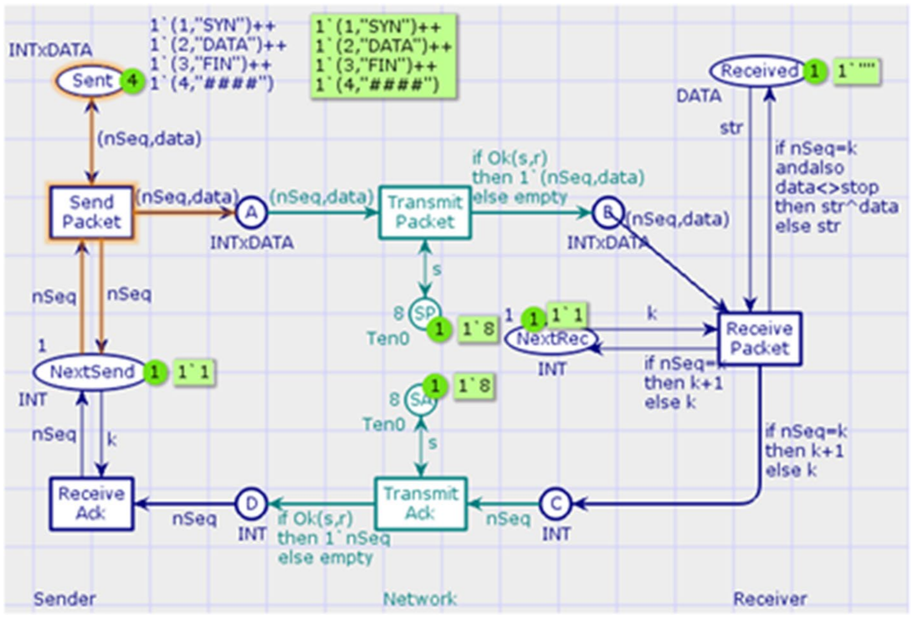


**Fig. 7** CPN model for transport layer

the number of port and the message type used to send or receive data. The machine part of this model is shown in Fig. 8.

As explained previously, the operation of signalization in the transport layer is based on some variables and invariants which are mainly syn, fin and ack included in the TYPE_MSG set and representing messages used for opening, closing and acknowledging a message respectively. Other variables are port_source for source port, port_destination for destination port included in the PORT set and Time counter.

| | | |
|---|---|---|
| Trans_syn<br>Sees<br>Context_transport<br>**VARIABLES**<br>Syn<br>Ack<br>Fin<br>Port_source<br>Port_dest<br>Type_msg<br>Time | **INVARIANTS**<br>Syn⊆ TYPE_MSG<br>Ack⊆ TYPE_DATA<br>Fin ⊆ TYPE_DATA<br>Port_source ⊆ PORT<br>Port_dest⊆ PORT<br>Type_msg⊆ TYPE_MSG<br>Time ∈ N | **INITIALIZATION**<br>Syn := ϕ<br>Ack := ϕ<br>Fin := ϕ<br>Port_source := ϕ<br>Port_dest := ϕ<br>Type_msg := ϕ<br>Time := 0 |

**Fig. 8** Event-B machine for signalization in transport layer

As initialization event, all these variables are set to the empty set and the value zero for the time variable. In event-B using Rodin, the signalization machine for the transport layer is coded as following:

```
MACHINE transport
SEES context_transport
VARIABLES
    ack
    syn
    FIN
    port_destination
    port_source
    type_msg
    R_ack
    R_syn
INVARIANTS
    inv1: ack ⊆ TYPE_MSG
    inv2: syn ⊆ TYPE_MSG
    inv3: FIN ⊆ TYPE_MSG
    inv5: port_destination ⊆ PORT
    inv6: port_source ⊆ PORT
    inv7: type_msg ⊆ TYPE_MSG
    inv8: R_ack ⊆ TYPE_MSG
    inv9: R_syn ⊆ TYPE_MSG
EVENTS
Initialisation
  begin
      act1: ack := ∅
      act2: syn := ∅
      act3: FIN := ∅
      act5: port_destination := ∅
      act6: port_source := ∅
      act7: type_msg := ∅
      act8: R_ack := ∅
      act9: R_syn := ∅
  end
END
```

Other events of this phase are generated every sending or receiving operation for the three types of messages. So, there is send_ACK and receive_ACK, send_SYN and receive_SYN, FIN_syn and FIN_ACK. For each event, there are some guards for verifying conditions such as the source and destination ports are different, the sequence number of acknowledgment and control like it is shown in the code of receiving an ACK message:

```
MACHINE transport
SEES context_transport
    .............
Event EVENT receive_ack ⟨ordinary⟩ ≙
    any
        receive
        N_seq
        N_ctrl
    where
        grd1:  port_source ≠ port_destination
        grd2:  MODE = receiving
        grd3:  N_seq > 0
        grd4:  N_ctrl > 0
    then
        act1: type_msg := R_ack
    end
END
```

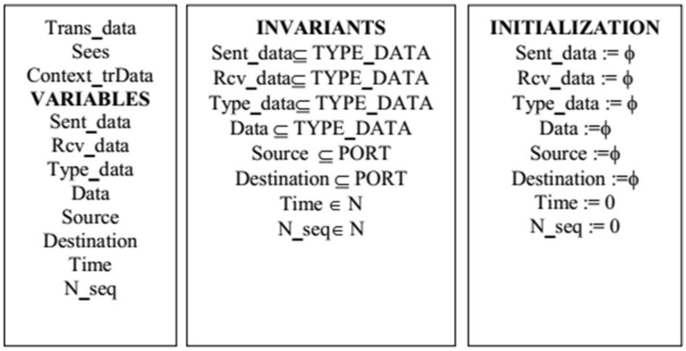
For the data transmission phase, the following event-B model is proposed:

As context for this part, two sets are needed, where the first one is TYPE_DATA used to specify the type of sent or received data for example scalar, image or video data. The second set is named DATA representing information to transmit. So, the appropriate context is shown below:

```
CONTEXT context_trData
EXTENDS context_transport
SETS
    TYPE_DATA
    DATA
END
```

Then, the corresponding machine is represented in Fig. 9.

When reading variables and invariants parts of this model, the following elements can be removed:



| | | |
|---|---|---|
| Trans_data | **INVARIANTS** | **INITIALIZATION** |
| Sees | Sent_data⊆ TYPE_DATA | Sent_data := ϕ |
| Context_trData | Rcv_data⊆ TYPE_DATA | Rcv_data := ϕ |
| **VARIABLES** | Type_data⊆ TYPE_DATA | Type_data := ϕ |
| Sent_data | Data ⊆ TYPE_DATA | Data :=ϕ |
| Rcv_data | Source ⊆ PORT | Source :=ϕ |
| Type_data | Destination ⊆ PORT | Destination :=ϕ |
| Data | Time ∊ N | Time := 0 |
| Source | N_seq∊ N | N_seq := 0 |
| Destination | | |
| Time | | |
| N_seq | | |

**Fig. 9** Event-B machine for data transmission in transport layer

– Sent_data: included in the set TYPE_DATA and representing data transmitted by a node.
– Rcv_data: included in the set TYPE_DATA and representing data received by a node.
– Type_data: it can be scalar, image, video or all other kind of data which can be transmitted or received by sensor nodes.
– Data: it is the proper information to be treated and transferred.
– Segment : the complete segment structure.
– Data_size : data size to transmit.
– Source: included in the set PORT and representing the source port of data transport.
– Destination: included in the set PORT and representing the destination port of data transport.
– Time: it is the time counter.
– N_seq: each message must have a sequence number to be differentiated from other messages. This number is important for the signalization phase too.

At the beginning, all these variables are initialized to the empty set, except Time and N_seq which are initialized to zero. In data transport, three events occur in the system: INITIALIZATION, send_DATA and receive_DATA. The first event is used in the beginning of operation as explained in the previous paragraph. The second event noted send_DATA is based on six guards for verifying logical conditions and calculating some parameters to send information by a node, like: source and destination ports are different; the node is in the sending mode and how to calculate N_seq from data and segment size. Actions to perform for this event are sending data and incrementing time counter. The third event noted receive_DATA is based on ports and mode verification as guards and receiving data then incrementing the time counter as actions. After combining all the previous events, the following model implemented in Rodin platform can be obtained:

```
MACHINE transport_DATA
SEES context_trData
VARIABLES
      Sent_data
      Rcv_data
      type_data
      segment
      source
      destination
      time
      N_Seq
      Data_size
INVARIANTS
      inv1: $Sent\_data \subseteq TYPE\_DATA$
      inv2: $Rcv\_data \subseteq TYPE\_DATA$
      inv3: $type\_data \subseteq TYPE\_DATA$
      inv4: $segment \subseteq type\_data$
      inv5: $source \subseteq PORT$
      inv6: $destination \subseteq PORT$
      inv7: $time \in \mathbb{N}$
      inv8: $N\_Seq \in \mathbb{N}$
      inv9: $Data\_size \subset DATA$
EVENTS
Initialisation
    begin
          act1: $Sent\_data := \varnothing$
          act2: $Rcv\_data := \varnothing$
          act3: $type\_data := \varnothing$
          act4: $segment := \varnothing$
          act5: $source := \varnothing$
          act6: $destination := \varnothing$
          act7: $time := 0$
          act8: $N\_Seq := 0$
          act9: $Data\_size := 0$
    end
Event send_DATA ⟨ordinary⟩ $\widehat{=}$
    any
          send
          Win_size
          N_Segment
          som_size_segemnt
    where
          grd1: $source \neq destination$
          grd2: $MODE = send$
          grd3: $N\_Seq = N\_Seq + 1$
          grd4: $N\_Seq = Win\_size$
          grd5: $N\_Seq = (Win\_size * N\_Segment) + 1$
          grd6: $Data\_size = som\_size\_segemnt$
    then
          act1: $Sent\_data := segment$
          act2: $time := time + 1$
    end
Event receive_DATA ⟨ordinary⟩ $\widehat{=}$
    any
          receiving
    where
          grd1: $source \neq destination$
          grd2: $MODE = receiving$
    then
          act1: $Rcv\_data := segment$
          act2: $time := time + 1$
    end
END
```

## 4.3 Network Layer

It is known that the network layer is responsible to manage the packets routing operation. The main problem is to define routes or links between each pair of nodes. In this level, two sets named PACKET and LINKS are used as in the following context:

```
CONTEXT context_Network
EXTENDS context_transport
SETS
      PACKET
      LINKS
END
```

For the machine part of the network layer model, the used variables and invariants are described in Table 1.

In the first time, all these variables are initialized to the empty set. That is modeled by the INITIALIZATION event for the corresponding machine. Other events are necessary to complete definition of the network layer operation. For managing links, ADD_LINK and DEL_LINK events are used to add and delete routes for a node. For managing packets, SEND, RECEIVE, FORWARD and LOSING events allow respectively to send, receive and forward a packet, the last one is used to trait situation of losing a packet. This is written in event-B as a machine like it is below:

```
MACHINE Network
SEES context_Network
VARIABLES
        sent
        got
        lost
        links
        intmed_node
        chstor
INVARIANTS
        inv1:  $sent \subseteq PACKET$
        inv2:  $got \subseteq PACKET$
        inv3:  $lost \subseteq PACKET$
        inv4:  $links \in (NODES \leftrightarrow NODES)$
        inv5:  $intmed\_node \subseteq NODES$
        inv6:  $chstor \in NODES \leftrightarrow PACKET$
EVENTS
Initialisation
    begin
            act1: $sent := \varnothing$
            act2: $got := \varnothing$
            act3: $lost := \varnothing$
            act4: $links := \varnothing$
            act5: $intmednode := \varnothing$
            act6: $chstor := \varnothing$
    end
Event SEND ⟨ordinary⟩ $\widehat{=}$
    any
            source
            destination
            packet
    where
            grd1:  $source \in NODES$
            grd2:  $destination \in NODES$
            grd3:  $source \neq destination$
            grd4:  $packet \in PACKET$
    then
            act1: $sent := sent \cup \{packet\}$
    end
Event RECEIVE ⟨ordinary⟩ $\widehat{=}$
    any
            source
            destination
            packet
    where
            grd1:  $source \in NODES$
            grd2:  $destination \in NODES$
            grd3:  $source \neq destination$
            grd4:  $packet \in PACKET$
    then
            act1: $got := got \cup \{packet\}$
    end
Event ADD_LINK ⟨ordinary⟩ $\widehat{=}$
    any
            X
            Y
    where
            grd1:  $X \in NODES$
            grd2:  $Y \in NODES$
            grd3:  $X \mapsto Y \notin links$
            grd4:  $X \neq Y$
    then
            act1: $links := links \cup \{X \mapsto Y\}$
    end
```

```
Event DEL_LINK ⟨ordinary⟩ $\widehat{=}$
    any
        X
        Y
    where
        grd1:  $X \in NODES$
        grd2:  $Y \in NODES$
        grd3:  $X \mapsto Y \in links$
        grd4:  $X \neq Y$
    then
        act1: $links := links \setminus \{X \mapsto Y\}$
    end
Event LOSING ⟨ordinary⟩ $\widehat{=}$
    any
        s
        d
        packet
    where
        grd1:  $packet \in sent \setminus (got \cup lost)$
        grd2:  $s \in NODES$
        grd3:  $d \in NODES$
        grd4:  $s \neq d$
        grd5:  $s \mapsto d \notin links$
        grd6:  $packet \in PACKET$
    then
        act1: $lost := lost \cup \{packet\}$
    end
Event FORWARD ⟨ordinary⟩ $\widehat{=}$
    any
        s
        t
        packet
    where
        grd1:  $s \in NODES$
        grd2:  $t \in NODES$
        grd3:  $packet \in PACKET$
        grd4:  $packet \in sent \setminus (got \cup lost)$
        grd5:  $s \mapsto t \in links$
        grd6:  $s \mapsto packet \in chstor$
        grd7:  $t \mapsto packet \notin chstor$
    then
        act1: $chstor := (chstor \setminus \{s \mapsto packet\}) \cup \{t \mapsto packet\}$
    end
END
```

**Table 1** Variables for network layer model

| Variable | Description |
|---|---|
| Sent | A packet to send |
| Got | A received packet |
| Lost | A lost packet when trying to transmit it |
| Links | Existing routes to a destination node |
| Intmed_node | Intermediate nodes for making a route |
| chstor | Packets to forward |

## 4.4 MAC Layer

The main role of the MAC layer is to manage problems due to medium access. Indeed, in a WSN, nodes communicate through shared frequency radio channels which create many problems and affects energy consumption of nodes. There are many solutions for these problems as MAC protocols generally based on altering the state of each node between active or sleep state in addition to manage the way of planning activities of a node according of its neighbors activities that can being Transmit, Receive, Sleep or Idle. This mechanism allows avoiding signal collision. It is the same mechanism used by the IEEE 802.11 standard witch use a preamble frame to determine collision probability due to hidden terminal problem known in CSMA/CA mechanism.

Before transmitting data frames, this mechanism uses two control frames called RTS (for Request To Send) and CTS (for Clear To Send). The medium access time for each node is managed by an allocation vector called NAV (for Network Allocation Vector). This vector contains time counters calculated as function of other parameters like the Back off value which is the random time to wait before transmitting, the DIFS or Distributed Inter Frame Spacing value and the SIFS or Short Inter Frame Spacing value. This mechanism is explained in Fig. 10.

Then, the node state changing can be modeled by the state diagram shown in Fig. 11.

So, the detailed operation of the MAC layer is modeled by the CPN of Fig. 12.

To validate this model, an event-B model is used too in the Rodin platform. The corresponding context contains necessary sets to manage frames and time called TYPE_FRAME and PERIODE respectively like it is shown here:

```
CONTEXT MAC_Context
EXTENDS context_Network
SETS
    TYPE_TARAME
    PERIODE
END
```

For the machine part of MAC layer, lot of variables and their invariants are used to manage node states, time counters, energy consumption and frame exchange like it is shown in Table 2.

The MAC layer machine model contains six events in terms of Rodin modeling. These events are INITIALIZATION, TRANSMIT_RTS, TRANSMIT_CTS, TRANSMIT_DATA, TRANSMIT_ACK and FIN. The first event concerns system initialization where it is important to set all numeric variables to zero and those of type set to the empty set. In the second event TRANSMIT_RTS shown after, any node that want to transmit data, it must send an RTS frame to ensure synchronization with other neighbors. In this event, it has to verify some guards where the most important are:

- grd1 that means the node must be in the TRANSMITTING or IDLE mode;
- grd2 that guaranties to avoid transmission where the value of NAV is zero, which means a risk of collision.
- grd5 to ensure that available time is sufficient to transmit the frame according to its size, the band width and the synchronization time.

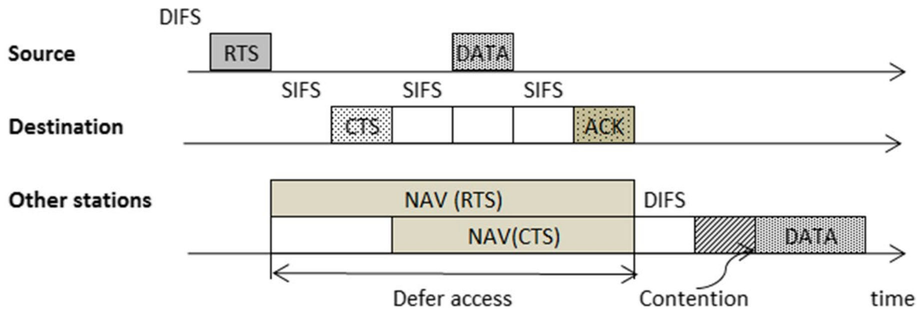


**Fig. 10** Time allocation principle in MAC layer (802.11)

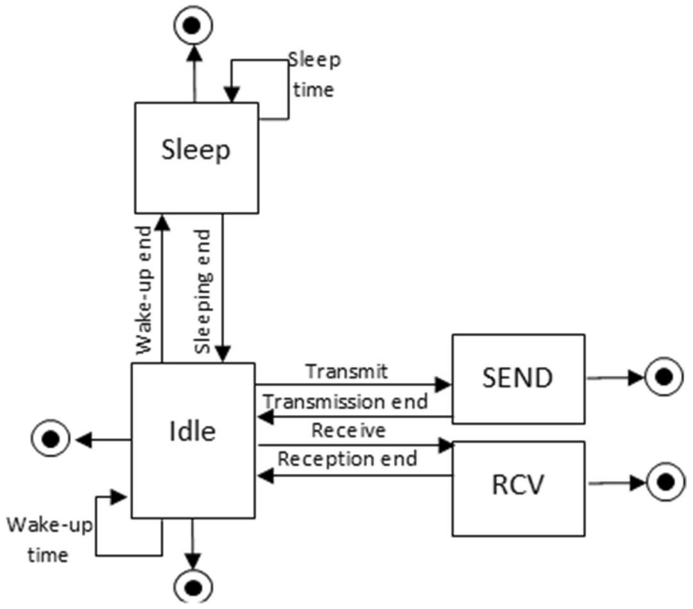


**Fig. 11** Node states diagram for MAC layer

– grd6 it is necessary to inform neighbors which will awake up that a transmission operation will happen for a time great than sleeping time.

Actions to perform in this event are preparing the RTS frame, computing SIFS value for ACK synchronization and updating energy value.

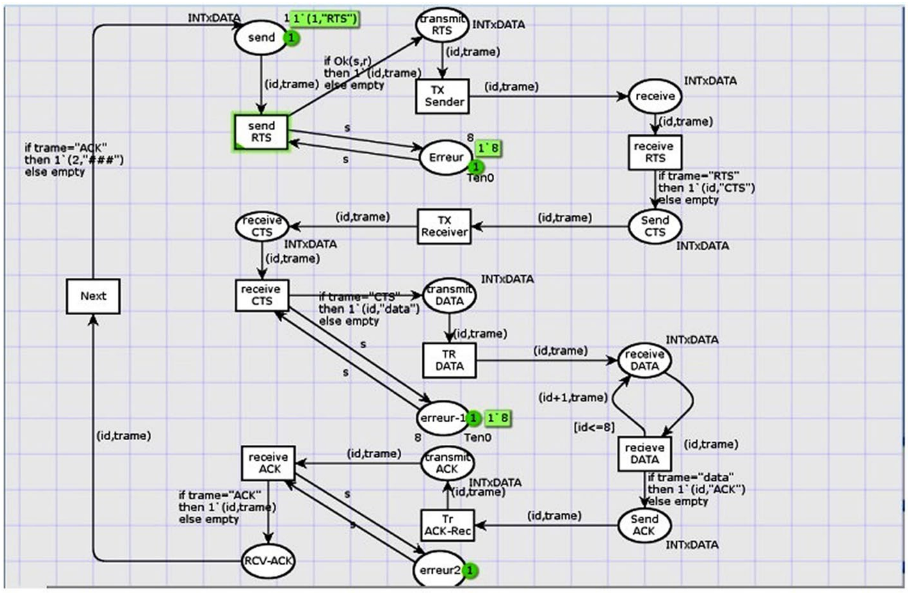

send
send RTS
transmit RTS
TX Sender
receive
receive RTS
Send CTS
TX Receiver
receive CTS
transmit DATA
TR DATA
receive DATA
recieve DATA
Send ACK
Tr ACK-Rec
transmit ACK
receive ACK
RCV-ACK
Erreur
erreur-1
erreur2
Next
INTxDATA
(id,trame)
if Ok(s,r) then 1`(id,trame) else empty
if trame="ACK" then 1`(2,"###") else empty
if trame="RTS" then 1`(id,"CTS") else empty
if trame="CTS" then 1`(id,"data") else empty
if trame="data" then 1`(id,"ACK") else empty
if trame="ACK" then 1`(id,trame) else empty
(id+1,trame)
[id<=8]
1`(1,"RTS")
1`8
Ten0
s

Fig. 12 CPN model for MAC layer

```
MACHINE Mac_Machine
EVENTS
Event TRANSMIT_RTS ⟨ordinary⟩ ≙
    any
        RTS
    where
        grd1: $MODE = Send \vee MODE = Idle$
        grd2: $NAV = 0$
        grd3: $RTS \subseteq Type\_frame$
        grd4: $BandWidth \neq 0$
        grd5: $Trans\_delay = Packet\_Size + BandWidth + Synchronization\_time$
        grd6: $Time\_trans > Time\_sleep$
    then
        act1: $type\_frame = RTS$
        act2: $SIFS = T\_last\_RTS/T\_First\_CTS$
            ......
    end
END
```

The same structure is used for the third event TRANSMIT_CTS but by replacing RTS frame by CTS one. Some other guards are added too, where the most important is to ensure that node can't transmit a CTS frame before receiving an RTS one. For TRANSMIT_DATA event, node must verify that a CTS frame has been received. This constraint is added as a guard for this event. With the same manner and to transmit an ACK frame, node has to wait until receiving a DATA frame. This is a guard that is added to the TRANSMIT_ACK event. Finally, the event called FIN is used at the end of a transmission session between a pair of nodes. So, receiving an ACK is a condition for this event being a guard in the event-B model.

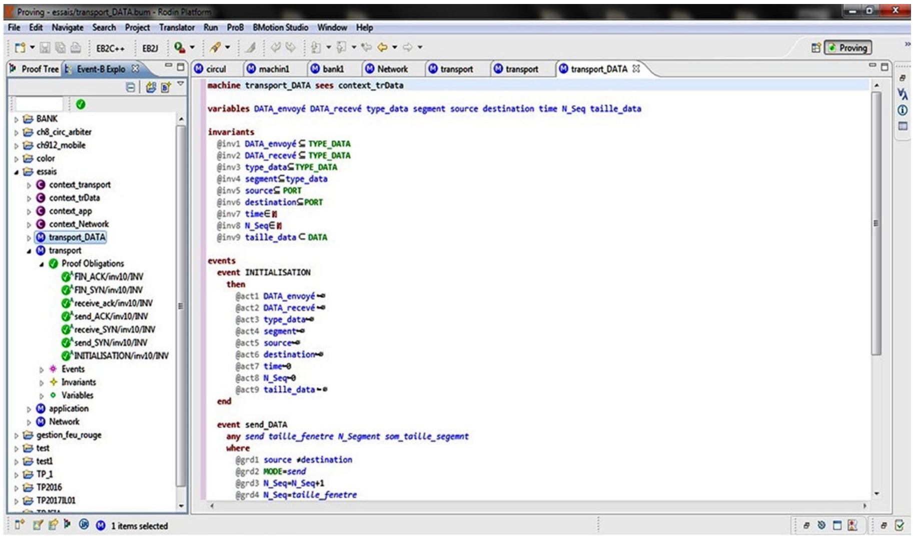
**Table 2** Variables for MAC layer model

| Variable | Description |
|---|---|
| Type_Rcvframe | Type of a received frame |
| Type_frame | A frame type, can be RTS or CTS |
| NAV network | allocation vector, for change node state and mode |
| Transmit | Transmitting mode |
| Reception | Receiving mode |
| IDLE | Idle mode |
| T_first_RTS | Time of an RTS frame beginning |
| T_last_RTS | Time of an RTS frame ending |
| T_first_CTS | Time of an CTS frame beginning |
| T_last_CTS | Time of an CTS frame ending |
| SIFS | Short Inter Frame Spacing value |
| DIFS | Distributed Inter Frame Spacing value |
| T_first_data | Time of data transmission beginning |
| T_last_data | Time of data transmission ending |
| T_first_ACK | Time of ACK transmission beginning |
| Trans_delay | Delay of data transmission |
| Time_trans | Time of data transmission |
| Time_sleep | Time of sleeping for a node |
| Energy | Available energy of a node |
| Packet_size | Size of a data packet |
| Bandwidth | Bandwidth of used network |
| Synchronization_time | Necessary time for synchronization |

## 5 Discussion and Results

The proposed approach relies on proof-based methodology (Event-B); it follows the same method as the one performed on orchestration [45] where refinement and proof-based methods with Event-B have been used for building correct services compositions. Event system is designed from a proved structure. Following refinement steps ensures that the protocol designed from a Petri network (with CPN tools, V4.0.1) model is synchronizable and well formed. So, the iterative approach to check and verify protocols can be avoided. Based on refinement, it is also possible to guarantee that exchanged messages are in same order as specified by their architecture. That provides a rigorous method for the performance evaluation and protocol development for WSNs through theorem proving. All proprieties and events are introduced in Rodin platform version 3.4.0, after many refinement and correction of all syntax and semantic errors, the obtained model and generated results are illustrated in Figs. 13 and 14.

As indicated in previous sections, verification with CPN tools is also made in this work. At first, the state space is used to check proprieties of the Petri net model. The state space is also called occurrence graphs, reachability graphs or reachability trees [46]. The state space tool integrated with CPN Tools can compute the state space and save it as a report file. In order to enter the state space, it must verify the following conditions:



```
machine transport_DATA sees context_trData

variables DATA_envoyé DATA_recevé type_data segment source destination time N_Seq taille_data

invariants
  @inv1 DATA_envoyé ⊆ TYPE_DATA
  @inv2 DATA_recevé ⊆ TYPE_DATA
  @inv3 type_data⊆TYPE_DATA
  @inv4 segment⊆type_data
  @inv5 source⊆ PORT
  @inv6 destination⊆PORT
  @inv7 time∈ℕ
  @inv8 N_Seq∈ℕ
  @inv9 taille_data ⊂ DATA

events
  event INITIALISATION
    then
      @act1 DATA_envoyé≔∅
      @act2 DATA_recevé≔∅
      @act3 type_data≔∅
      @act4 segment≔∅
      @act5 source≔∅
      @act6 destination≔∅
      @act7 time≔0
      @act8 N_Seq≔0
      @act9 taille_data≔∅
  end

  event send_DATA
    any send taille_fenetre N_Segment som_taille_segemnt
    where
      @grd1 source ≠destination
      @grd2 MODE=send
      @grd3 N_Seq=N_Seq+1
      @grd4 N_Seq=taille_fenetre
```

**Fig. 13** Rodin implementation for a WSN

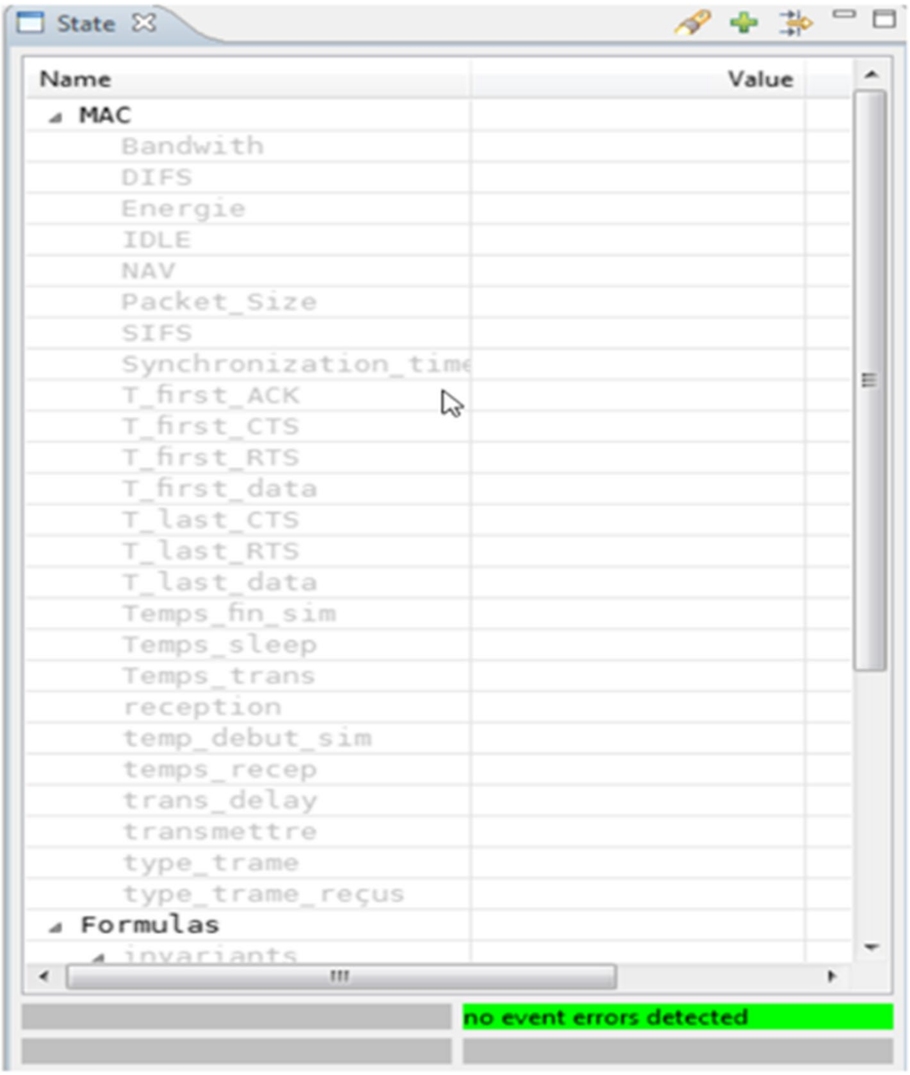
– There is no syntax error in the network.
– All transitions and places should have names.
– Names are required to be unique and to be alphanumeric.

The generated text file contains a standard report providing information about statistics (size of state space and strongly connected components graph or SCC graph), boundedness properties (inter and multi-set bounds for place instances), liveness properties (dead marking, dead/live transition instance) and fairness properties (impartial/fair/just transition instances). The proposed approach focuses on two important layers to evaluate witch are Mac and transport layers.

## 5.1 MAC Layer

### 5.1.1 Behavioral Proprieties

Figure 15 shows the first part of the CPN Tools state space report for the CPN model shown in Fig. 12. This part provides some state space statistics specifying how large the state space is. For the protocol shown in Fig. 12, there are 8753 nodes and 52497 arcs. The construction of the state space took 71 s (on a standard PC (Intel I3 , 1GB RAM)). Statistics for the SCC graph are also specified. It has 8753 nodes and 52497 arcs, and was calculated in 1 second. The fact that there are equal nodes in the SCC graph with the state space implies that there are non-trivial SCCs and hence there is no cycle in the state space of protocol. This means that an infinite occurrence sequence does not exist and that the protocol will terminate properly.



State

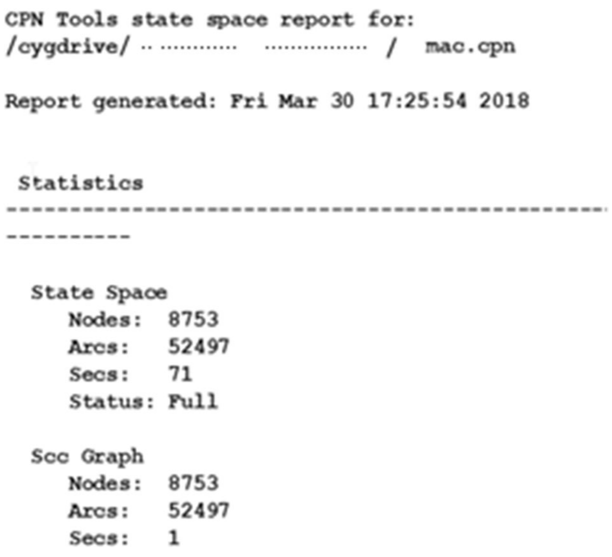
| Name | Value |
|---|---|
| MAC | |
| Bandwith | |
| DIFS | |
| Energie | |
| IDLE | |
| NAV | |
| Packet_Size | |
| SIFS | |
| Synchronization_time | |
| T_first_ACK | |
| T_first_CTS | |
| T_first_RTS | |
| T_first_data | |
| T_last_CTS | |
| T_last_RTS | |
| T_last_data | |
| Temps_fin_sim | |
| Temps_sleep | |
| Temps_trans | |
| reception | |
| temp_debut_sim | |
| temps_recep | |
| trans_delay | |
| transmettre | |
| type_trame | |
| type_trame_reçus | |
| Formulas | |
| invariants | |

no event errors detected

**Fig. 14** Illustration of Rodin generated results for MAC layer

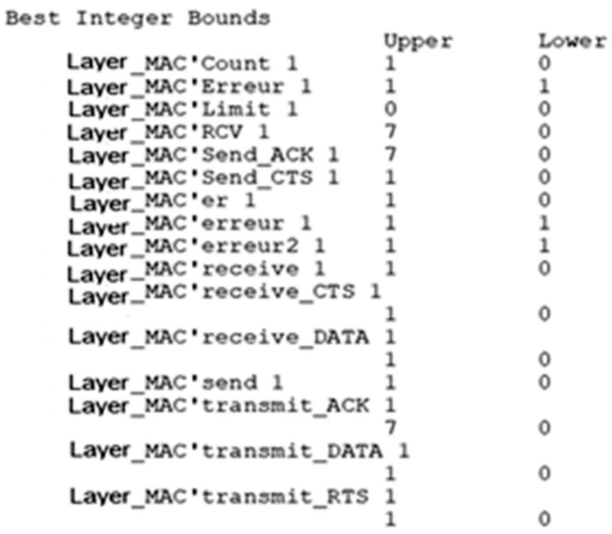
### 5.1.2 Boundedness Properties

The boundedness properties indicates how many and which tokens a place may hold, when all reachable markings are considered. Figure 16 specifies the best upper and lower integer bounds. The best upper integer bound of a place measure the maximal number of tokens that can reside on that place in any reachable marking. The best upper integer bound of the place MAC_Count is 1, which means that there is at most one token on the place and that there exists a reachable marking where there is one token on MAC_RECEIVE. It is what would be expected.



```
CPN Tools state space report for:
/cygdrive/ .. ............   ................ /  mac.cpn

Report generated: Fri Mar 30 17:25:54 2018


 Statistics
------------------------------------------------
----------

  State Space
     Nodes:  8753
     Arcs:   52497
     Secs:   71
     Status: Full

  Scc Graph
     Nodes:  8753
     Arcs:   52497
     Secs:   1
```

**Fig. 15** CPN tools statistics for MAC layer

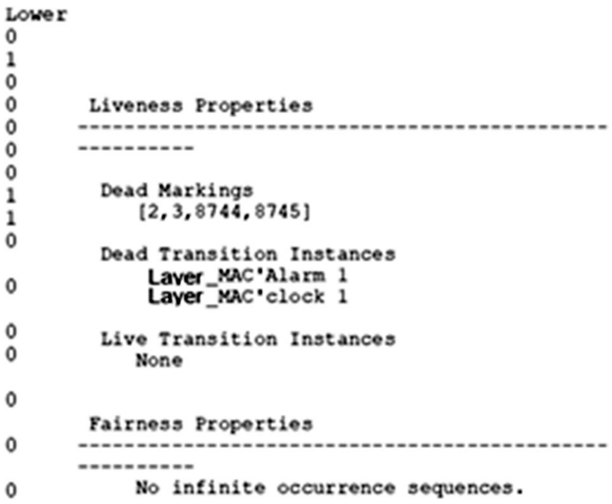


```
Best Integer Bounds
                                   Upper        Lower
     Layer_MAC'Count 1             1            0
     Layer_MAC'Erreur 1            1            1
     Layer_MAC'Limit 1             0            0
     Layer_MAC'RCV 1               7            0
     Layer_MAC'Send_ACK 1          7            0
     Layer_MAC'Send_CTS 1          1            0
     Layer_MAC'er 1                1            0
     Layer_MAC'erreur 1            1            1
     Layer_MAC'erreur2 1           1            1
     Layer_MAC'receive 1           1            0
     Layer_MAC'receive_CTS 1
                                   1            0
     Layer_MAC'receive_DATA 1
                                   1            0
     Layer_MAC'send 1              1            0
     Layer_MAC'transmit_ACK 1
                                   7            0
     Layer_MAC'transmit_DATA 1
                                   1            0
     Layer_MAC'transmit_RTS 1
                                   1            0
```

**Fig. 16** Boundedness properties for MAC CPN model

### 5.1.3 Liveness Properties

Figure 17 shows the state space report that indicates the liveness properties. The liveness properties inform us that there is a four dead marking, which have the nodes numbers 2, 3, 8744, 8754. A dead marking is a marking in which no binding elements are enabled. The fact that only 4 nodes are dead marking tells that the protocol as specified by the CPN model is partially correct, if execution terminates, the correct result will be obtained.



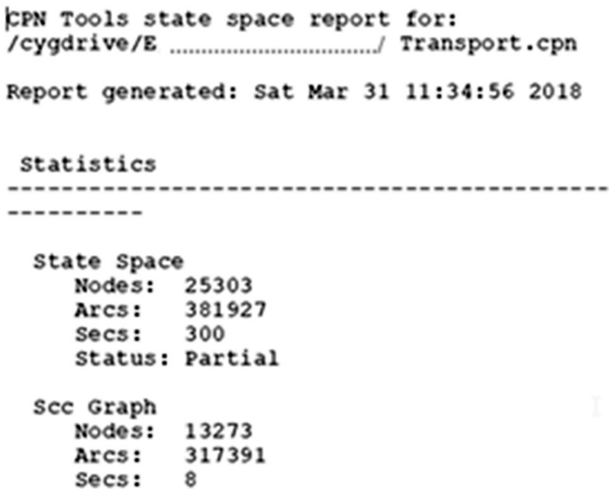
```
Lower
0
1
0
0       Liveness Properties
0      ------------------------------------------------
0      ----------
0
1         Dead Markings
1            [2,3,8744,8745]
0
          Dead Transition Instances
0             Layer_MAC'Alarm 1
              Layer_MAC'clock 1

0         Live Transition Instances
0            None

0
        Fairness Properties
0      ------------------------------------------------
       ----------
0            No infinite occurrence sequences.
```

Fig. 17 Liveness properties for MAC CPN model

## 5.2 Transport Layer

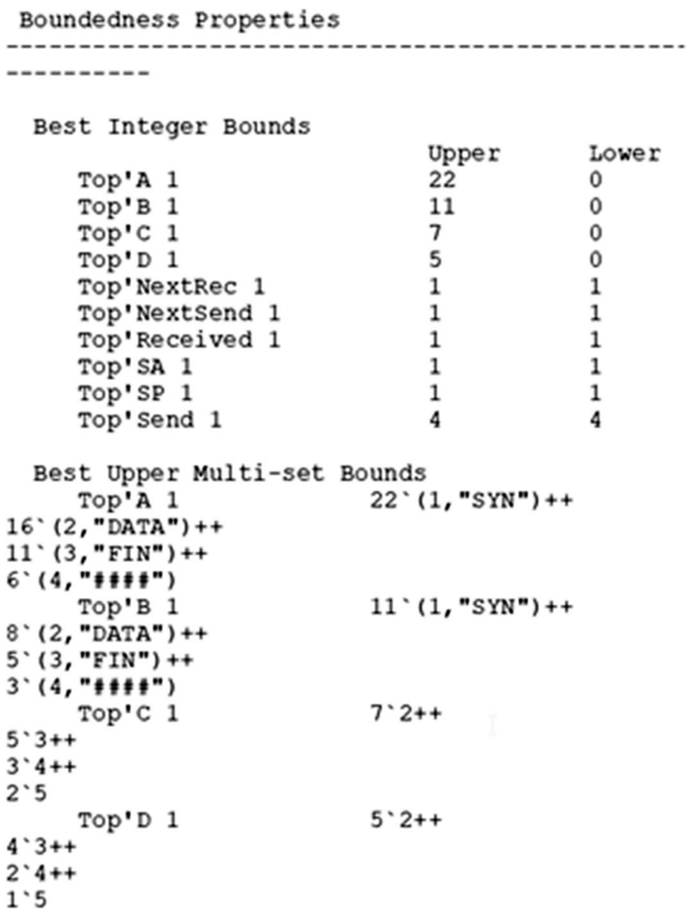
### 5.2.1 Behavioral Proprieties

In Fig. 18, the state space of the CPN model shown in Fig. 7 is reported. There are 25303 nodes and 381927 arcs. The construction of the state space took 300 seconds (on a standard PC (Intel I3, 1GB RAM)). Statistics for the SCC graph are also specified. It has 13273 nodes and 31391 arcs, and was calculated in 8 second. It's clear that there are less nodes in the SCC graph than in the state space, this implies that there are non-trivial SCCs and for this reason there are cycles in the state space of the protocol. This means that infinite occurrence sequences exist and that the protocol will not necessarily terminate.



```
CPN Tools state space report for:
/cygdrive/E ................................/ Transport.cpn

Report generated: Sat Mar 31 11:34:56 2018


 Statistics
---------------------------------------------
----------

  State Space
     Nodes:  25303
     Arcs:   381927
     Secs:   300
     Status: Partial

  Scc Graph
     Nodes:  13273
     Arcs:   317391
     Secs:   8
```

Fig. 18 CPN tools statistics for transport layer



```
 Boundedness Properties
----------------------------------------------------
----------

  Best Integer Bounds
                             Upper      Lower
     Top'A 1                 22         0
     Top'B 1                 11         0
     Top'C 1                 7          0
     Top'D 1                 5          0
     Top'NextRec 1           1          1
     Top'NextSend 1          1          1
     Top'Received 1          1          1
     Top'SA 1                1          1
     Top'SP 1                1          1
     Top'Send 1              4          4

  Best Upper Multi-set Bounds
     Top'A 1             22`(1,"SYN")++
16`(2,"DATA")++
11`(3,"FIN")++
6`(4,"####")
     Top'B 1             11`(1,"SYN")++
8`(2,"DATA")++
5`(3,"FIN")++
3`(4,"####")
     Top'C 1             7`2++
5`3++
3`4++
2`5
     Top'D 1             5`2++
4`3++
2`4++
1`5
```

**Fig. 19** Boundedness properties for transport CPN model

### 5.2.2 Boundedness Properties

Figure 19 illustrates the best upper and lower integer bounds. The best upper integer bound of a place measure the maximal number of tokens that can be on that place in any reachable marking. The best upper integer bound of the place TOP_A is 22, which means that there is at most 22 token on the place TOP_A and that there exists a reachable marking.

## 6 Conclusions

In this study, a formal method has been used and combined with Coloured Petri Network for WSN protocol stack verification. Event-B which is the formal method allows to model protocol layers and their proprieties, and Event-B invariants to check the protocol consistency. In addition, a coloured Petri network was developed for each layer. This last gives more rigorous verification and comprehension of a protocol stack.

It is clear that modeling and verifying liveness property of a protocol layers can add more guarantee regarding the validity of design. For future research in this field, it will be interesting to investigate modeling certain features of the protocol operation environment and their impact on its functionality.

## References

1. Adrien, B., Réjane, D., & Thierry, V. (2017). Enabling fast-prototyping of connected things using the WiNo* family. *Internet of Things ISTE Open Science*. https://doi.org/10.21494/ISTE.OP.2017.0138.
2. Tahar Abbes, M., Senouci, M., & Kechar, B. (2012). Impact of model mobility in ad hoc routing protocols. *IJCNIS*, *4*(10), 47–54.
3. Bechar, R., & Haffaf, H. (2014). Distributed monitoring for wireless sensor networks: A multi-agent approach. *International Journal of Computer Network and Information Security*. https://doi.org/10.5815/ijcnis.
4. Tahar Abbes, M., Belhirech, M., & Senouci, M. (2016). Adaptation of a routing algorithm in wireless video sensor network for disaster scenarios using JPEG 2000. *Wireless Networks*, *22*(2), 453–465.
5. Issariyakul, T., & Hossain, E. (2012). Introduction to network simulator 2 (NS2). In: *Introduction to network simulator NS2*. Boston: Springer. https://doi.org/10.1007/978-1-4614-1406-3_2.
6. OMNeT++ homepage. Retrieved July 7, 2017, from http://www.omnetpp.org.
7. Boulis, A., & Castalia, A. (2011). Simulator for wireless sensor networks and body area networks, version 3.2. User's manual, NICTA.
8. Network simulation tools project team. Retrieved July 7, 2017, from https://networksimulationtools.com/glomosim-simulator-projects/.
9. Sobeih, A., Hou, J. C., Lu-Chuan, K., Zhang Ning, H., Chen, W. P., Tyan, H. Y., et al. (2006). J-sim: A simulation and emulation environment for wireless sensor networks. *Wireless Communications*, *13*(4), 104–119.
10. Radak, J., Pavkovic, B., Mitton, N., Rousseau, F., & Stojmenovic, I. (2012). Emulation of large scale wireless sensor networks: From real neighbors to imaginary destination. In *Proceedings of ADHOC-NOW 2012* (pp. 459–471).
11. Xianda, C., Tae, K. K., & YongYoun, H. (2016). Integration of Markov random field with Markov chain for efficient event detection using wireless sensor network. *Computer Networks*. https://doi.org/10.1016/j.comnet.2016.07.004.
12. Obado, V., Djouani, K., & Hamam, Y. (2012). Hidden Markov model for shortest paths testing to detect a wormhole attack in a localized wireless sensor network. *Procedia Computer Science*, *10*, 1010–1017.
13. Vinutha, C. B., NAlini, N., & Nagaraja, M. (2017). Cluster-based adaptive power control protocol using hidden markov model for wireless sensor networks. *International Journal of Electronics*, *104*(6), 1–14.
14. Abrial, J. R. (2010). *Modeling in event-B: System and software engineering*. Cambridge: Cambridge University Press.
15. Hallerstede, S. (2009). Proving quick sort correct in event-B. *Electronic Notes in Theoretical Computer Science*, *259*, 47–65.
16. Robinson, K. (2012). *System modeling & design using event-B*. Sydney: School of Computer Science and Engineering, The University of New South Wales.
17. Butler, M., & Hallerstede, S. (2007). The Rodin formal modeling tool. In *Proceedings of BCS-FACS'07*.
18. Hoang, T. S., Hironobu, K., & Butler, M. (2016). Towards modular development in event-B. In *Proceedings of the 6th Rodin user and developer workshop* (pp. 9–16).
19. Smith, E. (2014). *Carl Adam Petri—Life and science*. Berlin: Springer.
20. Deseland, J., & Reisig, W. (1998). *Place/transition petri nets. Lectures on petri nets I: Basic models: Advances in petri nets* (pp. 122–173). Berlin: Springer.
21. Jensen, K., & Kristensen, L. M. (2009). *Coloured petri nets—Modeling and validation of concurrent systems*. New York: Springer.
22. Liang, C., & Zheng, W. (2012). Automated generation of test cases based on path optimization algorithm. In *Proceedings of the 2012 international conference on information technology and software engineering*. Lecture notes in electrical engineering (Vol. 212(2012)). Berlin: Springer.

23. CPN tools. Retrieved July 18, 2017, from http://cpntools.org/.
24. Boleslaw, M., & Abhishek, S. (2009). TransCPN—Software tool for transformation of colored petri nets. In *Proceedings of ITNG'09* (pp. 211–216).
25. Gawanmeh, A. (2011). Embedding and verification of ZigBee protocol stack in event-B. *Procedia Computer Science*, *5*(2011), 736–741.
26. Intana, A., Poppleton, M. R., & Merrett, G. V. (2014). A formal co-simulation approach for wireless sensor network development. *Electronic Communications of the EASST*, *70*(2014), 1–15.
27. Kamali, M., Liabnis, L., Petre, L., & Sere, K. (2014). Formal development of wireless sensor-actor networks. *Science of Computer Programming*, *80*(2014), 25–49.
28. Fakhfakh, F., Tounsi, M., Kacem, A. H., & Mosbah, M. (2016). Towards a formal model for dynamic networks through refinement and evolving graphs. In R. Lee (Ed.), *Software engineering, artificial intelligence, networking and parallel, distributed computing*. Studies in computational intelligence (pp. 227–243). Cham: Springer.
29. Nadeem, R. M., & Gill, A. (2017). A formal model for verification of ZigBee protocol for secure network authentication. *Indian Journal of Science and Technology*, *10*(20), 1–7.
30. Kamali, M., & Petre, L. (2016). Modeling link state routing in event-B. In *Proceedings of ICECCS'21* (pp. 207–210).
31. Fu, C., & Zheng, K. (2017). Formal modeling and analysis of ad hoc Zone Routing Protocol in event-B. *International Journal on Software Tools for Technology Transfer*. https://doi.org/10.1007/s10009-017-0463-4.
32. Azgomi, M. A., & Khalili, A. (2009). Performance evaluation of sensor medium access control protocol using coloured petri nets. *Electronic Notes in Theoretical Computer Science*. https://doi.org/10.1016/j.entcs.2009.06.021.
33. Martinez, D., Gonzalez, A., Blanes, F., Aquino, R., Simo, J., & Crespo, A. (2011). Formal specification and design techniques for wireless sensor and actuator networks. *Sensors*. https://doi.org/10.3390/s110101059.
34. Fu, X., Ma, Z., & Yu, Z. (2011). On wireless sensor networks formal modeling based on petri nets. In *Proceedings of WiCOM'7* (pp. 1–4). https://doi.org/10.1109/wicom.2011.6040356.
35. Ruiz, M. C., Mateo, J. A., Macia, H., Pardo, J. J., & Olivares, T. (2012). Formal modeling and performance evaluation of a novel role-based routing algorithm for wireless sensor networks. In *Proceedings of ADCOM'18* (pp. 4–11). https://doi.org/10.1109/ADCOM.2012.6563577.
36. Dâmaso, A., Freitas, D., Rosa, N., Silva, B., & Maciel, P. (2013). Evaluating the power consumption of wireless sensor network applications using models. *Sensors*. https://doi.org/10.3390/s130303473.
37. Zairi, S., Mezni, A., & Zouari, B. (2015). Formal approach for modeling, verification and performance analysis of wireless sensors network. In *Proceedings of international conference on wired/wireless internet communication* (pp. 381–395). Cham: Springer.
38. Le, K., Bui, T., Quan, T., Petrucci, L., & Andre, E. (2016). Congestion verification on abstracted wireless sensor networks with the WSN-PN tool. *Journal of Advances in Computer Networks*. https://doi.org/10.18178/JACN.2016.4.1.200.
39. Somappa, A. A. K., & Simonsen, K. I. F. (2016). Model-based development for MAC protocols in industrial wireless sensor networks. In *PNSE petri nets*.
40. Le, K., Cao, T., Le, P., Pham, B., Bui, T., & Quan, T. (2017). Probabilistic congestion of wireless sensor networks: A coloured petri net based approach. *Communications on Applied Electronics*, *7*(2), 1–7.
41. Hu, X., & Jiao, L. (2017). Efficient modeling and performance analysis for IEEE 802.15.4 with coloured petri nets. In *Proceedings of IWQoS'17* (pp. 1–6).
42. Elleuch, M., Hasan, O., Tahar, S., & Abid, M. (2014). Formal probabilistic analysis of detection properties in wireless sensor networks. *Formal Aspects of Computing*. https://doi.org/10.1007/s00165-014-0304-0.
43. Farah, Z., Ait-Ameur, Y., Ouederni, M., & Tari, A. (2017). A correct-by-construction model for asynchronously communicating systems. *International Journal on Software Tools for Technology Transfer*. https://doi.org/10.1007/s10009-016-0421-6.
44. Méry, D., & Poppleton, M. (2017). Towards an integrated formal method for verification of liveness properties in distributed systems: With application to population protocols. *Software & Systems Modeling*. https://doi.org/10.1007/s10270-015-0504-y.
45. Ait-Sadoune, I., & Ait-Ameur, Y. (2010). Stepwise design of BPEL web services compositions: An event B refinement based approach. In R. Lee, O. Ormandjieva, A. Abran, & C. Constantinides (Eds.), *Software engineering research, management and applications*. Studies in computational intelligence (Vol. 296). Berlin: Springer.
46. Reith, M., Carr, C., & Gunsch, G. (2002). An examination of digital forensic models. *International Journal of Digital Evidence*, *1*, 1–12.

**Rachid Bechar** obtained his engineer degree then his Magister in computer science from the University of Mostaganem (Algeria) in 2003 and 2010 respectively. He also obtained the PhD degree in computer science from the University of Oran1 (Algeria) in 2015. He worked at the University of Mostaganem as an engineer in software development from 2004 to 2010. In 2011 he joined the University of Chlef (Algeria) as an assistant lecturer then associate lecturer in computer science department where he is working now. His research interests include computer networks, wireless sensor networks, network modeling and evaluation and IoT and ML techniques.

**Mounir Tahar Abbes** received the MASTER degree in computer science from University of Poitiers (ENSMA, France) in 2003, and PHD (2012) in computer science from the university of Oran, Algeria and HDR from university of Oran in 2016. He is working Currently as professor in University of Chlef (Algeria) where he advises many projects in Computer science and telecommunication. His current research interests include mobile wireless sensor/actuator networks, Zigbee/IEEE 802.15.4 technologies deployment, hybrid cellular networks, Vehicular ad hoc networks, vehicular sensor networks and heterogeneous wireless networks, with special emphasis on radio resource management techniques, performance modeling, provisioning QoS and industrial applications. He is currently head of a research team on wireless sensor networks and their societal and industrial applications.

**Freha Mezzoudj** optained her PhD degree in computer science from University of science and technology of Oran (Algeria) in 2018. She is working as associate lecturer in department of Computer Science in University Hassiba Benbouali of Chlef (Algeria). His research interests include Machine Learning, Pattern Recognition, Neural Networks and Artificial Intelligence and Evolutionary Computation.

**Ladjel Bellatreche** is a Professor at National Engineering School for Mechanics and Aerotechnics (ENSMA), Poitiers (France) since 2010. He leads the Data and Model Engineering Team of Laboratory of Computer Science and Automatic Control for Systems (LIAS). He was a Visiting Professor of the Québec en Outaouais, Canada, a Visiting Researcher at Department of Computer Science, Purdue University, USA and Department of Computer Science of Hong Kong University of Science and Technology, China. He is also involved in Research Postgraduate Programmes in Computer Science of several Universities and Schools in Algeria, where he got his Engineer degree in Computer Science from University of SBA in 1992. His research area is semantic data Integration, ontology, database design, Big Data & Cloud and green computing.