Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 5 (2011) 736–741

Procedia Computer Science

International Symposium on Frontiers in Ambient and Mobile Systems

Embedding and Verification of ZigBee Protocol Stack in Event-B

Amjad Gawanmeh[1]

Khalifa University of Science, Technology, and Research Sharjah Campus PO.Box 573 Sharjah, UAE

Abstract

ZigBee is a specification that enhances the IEEE 802.15.4 standard by adding network and security layers and an application framework for high level communication in Wireless Sensor Networks (WSN). Since ZigBee is essential in the operation of WSN; it is imperative to verify the correctness of its design. Formal methods can be used efficiently to verify a wide range of systems, including ZigBee protocol stack specification. In this paper we use Event-B formal verification method to model and verify ZigBee protocol stack by providing embedding of the protocol primitives in Event-B. Our approach takes advantage of the Event-B method capabilities to model designs at different levels of abstraction which fits the layered nature of the protocol.

Keywords: ZigBee, Formal Verification, Event-B, First Order Logic, Theorem Proving;

1. Introduction

WSNs has become an important research field that is rapidly growing due to the development of new technologies in inexpensive sensors. As many interesting and diverse applications of Wireless Sensor Networks (WSNs) are explored, there has become an urgent need to develop a communication and management protocol that is tailored for the special environment and requirements of WSN operations. ZigBee [6] is a specification that enhances the IEEE 802.15.4 [4] standard by adding network and security layers and an application framework, it provides a suite of high level communication using small, low-power digital radios for Low-Rate Wireless Sensor Networks. The technology defined by the ZigBee specification is intended to be simple and less expensive than other wireless protocols.

Simulation based verification is widely used to verify systems. In simulation a test-bench is built to functionally verify the design performs to specification by providing meaningful input scenarios. However, it is infeasible to simulate all possible scenarios of the system under test, in addition, certain corner case scenarios are highly likely to be missed during simulation. Therefore, formal methods can be used in order to verify the correctness of the system under design, along with simulation based techniques. Formal verification [8] uses mathematical reasoning to verify that a design specification comprehends certain design requirements, and it has been used for the analysis, modeling and verification of a variety of hardware and software systems. Using formal reasoning for the development and analyses of a formal model for the given system will increase the chances for finding errors in the system under test. The usage of wireless systems in safety critical applications and their complexity are good motivations for using formal methods in this domain [9].

[1] Email: amjad.gawanmeh@kustar.ac.ae

Selection and/or peer-review under responsibility of Prof. Elhadi Shakshuki and Prof. Muhammad Younas.
doi:10.1016/j.procs.2011.07.097

In this paper, we use formal verification in order to model and verify ZigBee protocol stack. We verify certain design requirements for the protocol stack by modeling its functional operations, and then we verify properties related to its specifications using the Event-B method. For the embedding of ZigBee protocol layers in Event-B [15] first-order logic, the protocol semantics should be well defined and understood to be semantically embedded in the Event-B language [12]. This embedding allows the verification of properties related to the protocol specifications while preserving the semantics of the protocol in the Event-B model. ZigBee is a layered protocol, therefore it can be modeled in Event-B at different levels of abstraction.

2. Related work

Formal methods have been used in the analysis and verification of similar wireless systems, for instance, in the PRISM model checker has been used to analyze a sub protocol of the IEEE 802.11 standard for wireless local area networks [10]. Ballarini and Miller [1] also used the same approach to verify the Medium Access Control protocol SMAC. In another approach, the ETMCC model checking has been used for the analysis of a variant of the central access protocol of the IEEE 802.11 standard [11].

Other work on the verification of ZigBee includes the work in [13] that uses OPNET in order to simulate the IEEE 802.15.4/ZigBee. Fruth [7] used PRISM model checker for probabilistic verification of the IEEE 802.15.4 networking standard. Yang et al. [14] used OMNeT++ discrete event simulation for Maritime Surveillance Sensor Networks(MSSNs) verification, ZigBee protocol was simulated in the data link layer. Duflot et al. [3] present a formal analysis of the discovery phase of the Bluetooth wireless communication protocol using the DTMC model. In [2] Chunqing and Jiancheng studied the data security protection of ZigBee technology.

The use of probabilistic model checking is limited to systems that can only be expressed as probabilistic finite state machines or Markov chains. Another major limitation of the probabilistic model checking approach is state space explosion. The work in [9] used Higher Order Logic (HOL) theorem proving for analysis of the probabilistic properties in wireless systems. The method is highly interactive, while using first-order logic can reduce interaction with the prover.

In our approach, we model and verify the functional behaviors of the ZigBee protocol stack considering the transaction level functionality of the protocol, while most of the state of the art methods focus on probabilistic and low level related properties. The use of Event-B provides a rich expressive modeling language, and on the other hand, it is less interactive, compared to HOL theorem proving approach.

3. Verification Methodology

In order to embed the protocol specifications in Event-B [15], a semantical map between these specifications and Event-B language should be established. Fig. 1 below defines the relation between primitive protocol components and Event-B constructs. Once we have the protocol stack embedded in Event-B, its correctness is established by proof obligations for the invariants, where each event, including the initialization event, should preserve these invariants. Event-B guards are used to define preconditions that should hold before the event can be executed. The guard and the action of an event define a relation between variables before the event holds and after. Proof obligations are produced from events in order to state that the invariant condition is preserved. These proof obligations need to be verified in order to proof the correctness of the invariants. ZigBee devices and nodes are defined as constants, functions and operations are defined as events, similarly, layers interfaces are modeled as events. Once an event is executed, the set of connected nodes that forms a network is updated accordingly. Each layer of ZigBee protocol stack is modeled in Event-B machine along with its attributes, then, interfaces between layers is modeled using events. Finally, properties related to the correct operation of the protocol are modeled as Event-B invariants. Rodin framework [5] is used to check the consistency of these properties with regards to the Event-B model as shown in Fig. 1.

To create the Event-B model for the protocol, we use the same layered approach, where machines are used to model layers and interfaces between these layers are defined with events. The abstract model of the protocol defines its basic components and its initial states. Proof obligations for invariants are generated from the abstract model in the initial state, in addition, certain properties about the correctness of the protocol are defined for the concrete model as invariants, and another set of proof obligations are generated for these invariants from the concrete model.

Event-B abstract model can be refined with events or variables. We use events to model different layers primitives. In the refined model, we use similar events to model the updates on the abstract one. The correctness of the protocol properties, modeled by the invariants with regards to the event-B concrete model is achieved through the correctness of the gluing invariant over the two models. In the following section, we show how ZigBee protocol stack components and its properties are embedded in Event-B.

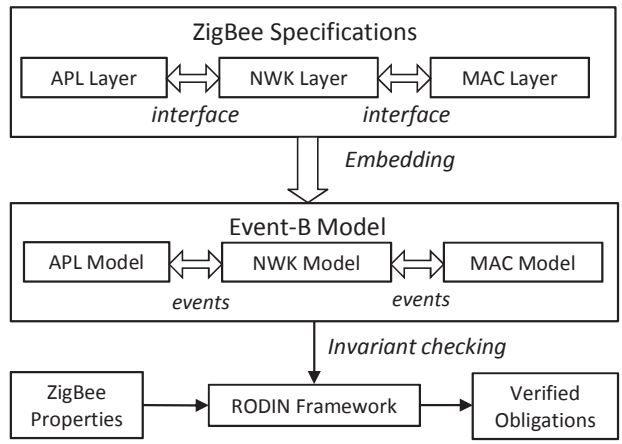

Fig. 1. Event-B Embedding and Verification of ZigBee Protocol Layers

4. Embedding and Verification of ZigBee Protocol Stack in Event-B

The ZigBee stack architecture is made up of a set of layers that performs services for the layer above. Layers have a data entity that provides a data transmission service and a management entity that provides all other services. Each service entity exposes an interface to the upper layer through a service access point (SAP), and each SAP supports a number of service primitives to achieve the required functionality [6].

4.1. ZigBee Protocol Stack

The ZigBee stack architecture includes three layered components: Medium Access Control (MAC) layer, Physical layer, and the ZigBee Network layer. Each component provides an application with its own set of services and capabilities. The services are provided by two entities: the data entity and the management. Each provides its services through its own service access point. The network layer includes two service entities that provide the necessary functionality to interface with the application layer. These entities are the network layer data entity (NLDE) and the network layer management entity (NLME). Each provides data services via its associated SAP [6].

4.2. Data and Management Entities

We model the network layer data entity primitives Event-B by providing en embedding in Rodin [5] platform. NLDE-DATA.request primitive requests the transfer of a data PDU from the local APS sub-layer entity to a single or multiple peer APS sub-layer entities. NLDE-DATA.confirm reports the results of a request to transfer a data PDU from a local APS sub-layer entity to a single peer APS sub-layer entity. Finally, NLDE-DATA.indication indicates the transfer of a data PDU from the network layer to the local APS sub-layer entity.

ZigBee devices can perform the following functions: join a network, leave a network, rejoin a network. Both ZigBee coordinators and routers can permit devices to join or leave the network, assign logical network addresses, and maintain a list of neighboring devices. Only ZigBee coordinators can establish a new network.

CONTEXT C0
SETS
NLDE — Network layer data entity primitives type
NLDE_STATUS_NLDE — Request status message used in con_rm
DEVICES — ZigBee Devices
CONSTANTS
NLDE_DATA_REQ, NLDE_DATA_CONF, NLDE_DATA_IND
INVALID_REQUEST, MAX_FRM_COUNTER, NO_KEY
NODES, ROUTERS, COORDINATORS
AXIOMS
axm1 : NLDE_DATA_REQ ∈ NLDE ∧ NLDE_DATA_CONF ∈ NLDE
axm2 : NLDE_DATA_IND ∈ NLDE ∧ INVALID_REQUEST ∈ NLDE_STATUS
axm3 : MAX_FRM_COUNTER ∈ NLDE STATUS
axm11 : NODES ⊆ DEVICES ∧ ROUTERS ⊂ DEVICES ∧ COORDINATORS ⊂ DEVICES
axm15 : NODES ∩ (ROUTERS ∪ COORDINATORS) = ∅ ROUTERS ∧ COORDINATORS = ∅
END

Data entity is modeled above, where the first part defines three sets, the *NLDE* primitive type to identify the commands used in the interface within the protocol, status messages, *NLDE_STATUS* and finally, a set of possible devices, which can be regular nodes, routers, coordinators, etc. In the second part, all possible NLDE primitives, status messages, and devices are defined. These components are bound to their sets through a number of axioms, the last two axioms are used to model the restrictions on the above sets. Similarly, network layer management entity and primitives are modeled in Event-B:

```
CONTEXT C1 EXTENDS C0
SETS
    NLME                NLME_STATUS
CONSTANTS
    NLME_NWK_FORM_REQ   NLME_NWK_FORM_CONF   NLME_JOIN_REQ          SUCCESS
    NLME_JOIN_IND       NLME_JOIN_CONF       INVALID_REQ STAR_UP   FAILURE
AXIOMS
    axm02 : NLME_NWK_FORM_REQ ∈ NLME ∧ NLME_NWK_FORM_CONF ∈ NLME
    axm10 : NLME_JOIN_REQ ∈ NLME ∧ NLME_JOIN_IND ∈ NLME
    axm32 : FAILURE ∈ NLME_STATUS ∧ SUCCESS ∈ NLME_STATUS ∧ INVALID_REQ ∈ NLME_STATUS
END
CONTEXT C2 EXTENDS C1
SETS
    MLME
CONSTANTS
    MLME_SCAN_REQ   MLME_SCAN_CONF  MLME_SET_CONF   MLME_SET_REQ
    MLME_START_REQ  MLME_START_CONF        node1    node2    coord    router
AXIOMS
    axm01 : MLME_SCAN_REQ ∈ MLME ∧ MLME_SCAN_CONF ∈ MLME
    axm04 : node1 ∈ NODES ∧ node2 ∈ NODES ∧ router ∈ ROUTERS ∧ coord ∈ COORDINATORS
END
```

The functional requirements are modeled in Event-B by providing a set of events, where each event checks for the validity of the service for the initiating device.

4.3. Layers Services and Interfaces

We define the basic machine including the initialization event, and then we define the services provided by the layers interfaces in a refined machine. The following ZigBee operations are embedded in Event-B:

Initialization event. In the initialization event, attributes for various devices are set to their default values as described in the protocol specifications. The network is assumed to be empty, which means it is not formed yet, coordinators are the only devices that are granted permission to establish a network. We define a function *nwkCapable* that maps a device into a *boolean* value that represents its capability of creating network. This value might change, for instance when a coordinator creates a network. In addition, another function called Parent is defined below, this function is used to pair nodes with routers or coordinators so that nodes can join the network through them. The initialization event is defined below:

```
MACHINE M0
SEES C2
VARIABLES
    nwkCapable      NETWORK      NLMEcmd      MLMEcmd      NLME_status      Parent
INVARIANTS
    inv11 : NETWORK ⊆ DEVICES ∧ NLMEcmd ∈ NLME ∧ MLMEcmd ∈ MLME ∧ nwkCapable ∈ DEVICES → BOOL
    inv14 : NLME_status ∈ NLME STATUS ∧ Parent ∈ {COORDINATORS ∪ ROUTERS − NODES}
EVENTS
Initialisation
begin
    act1 : node1nwkCapable ≔ FALSE
    act4 : coordnwkCapable ≔ TRUE
    act5 : routernwkCapable ≔ FALSE
    act6 : NETWORK ≔ ∅
    act7 : Parent ≔ {coord ↦ node1} ∪ {router ↦ node2}
    act8 : nwkCapable ≔ {node1 ↦ FALSE; node2 ↦ FAL_SE; coord ↦ TRUE; router ↦ FALSE}
end
END
```

Establishing a new network. The procedure to successfully start a new network is illustrated in the message sequence chart shown in Fig 2 [6] as given in ZigBee specifications. When a request to establish a new network is initiated through use of the *NLME-NWK-FORMATION.request* primitive, the Event-B confirms that the device has the appropriate *nwkCapable* value, and not currently joined to a network. This operation is defined using the following guarded event:

Event FormReq
any dev
when
grd5 : dev ∈ COORDINATORS ∧ nwkCapable(dev) = TRUE
grd1 : NLMEcmd = NLME_NWK_FORM_REQ
grd4 : NETWORK = ∅ ∧ MLME status = PERMITED
then
act1 : MLMEcmd ≔ MLME_SCAN_REQ
act2 : NLME_status ≔ SUCCESS
act3 : NETWORK ≔ NETWORK ∪ {coord}
act11 : NLMEcmd ≔ NLME_NWK_FORM_CONF
act12 : nwkCapable(dev) ≔ FALSE
end
END

Joining a Network. The procedure for permitting devices to join a network is initiated through the *NLMEPERMIT-JOINING.request primitive*. Only devices that are either the ZigBee coordinator or a ZigBee router shall attempt to permit devices to join the network. We consider joining a network through association case, other cases are modeled similarly. In addition other events such as leaving a network, resetting devices, and network discovery are also defined similarly. This operation is defined using the following guarded event:

Event Join
any
dev status devP
where
grd1 : dev NODES
grd2 : devP ROUTERS devP COORDINATORS
grd4 : devP NETWORK dev NETWORK
grd6 : status = PERMITTED ∧ Parent(devP) = dev
then
act1 : NETWORK ≔ NETWORK ∪ {dev}
act2 : NLME_status ≔ SUCCESS
end
END

4.4. 4.4. Protocol Properties

The three kinds of nodes in a ZigBee network are coordinators, routers and end device which are the sensor nodes. When ZigBee creates the network, there are some basic rules that should apply in order to maintain the operation of the network. These rules are modeled as Event-B invariants and must hold for the protocol to be correct. Each generates a number of proof obligations in Rodin, these proof obligations are proven one by one, some are automatically discharged using the proof system of the tool, and some must be proven interactively by providing certain rewrite rules to simplify the obligation.

In the following, we state four properties for illustration purposes. These properties are defined for the machine *M0*, and verified for the events of this machine. The first property states that the end devices should connect through a router or a coordinator, this property is modeled in Event-B using the following invariant:

Prop. 1 $\forall d, n \cdot d \in (ROUTERS \cup CORDINATORS) \wedge n \in NODES \wedge n \in NETWORK \Rightarrow Parent(d) = n$

The next property states that a coordinator cannot establish a network if it has already established one, and is modeled in Event-B using the following invariant:

Prop. 2 $\forall d \cdot d \in (CORDINATORS\ d \quad NETWORK) \Rightarrow nwkCapable(d) = FALSE$

A similar property states that routers and end devices cannot establish network and is modeled using the following invariant:

Prop. 3 $\forall d \cdot d \in DEVICES \wedge d \in (ROUTERS \cup NODES) \Rightarrow nwkCapable(d) = FALSE$

The last property states that any node cannot join an empty network. It is included in machine M0 above that executes the join event. It is modeled as follows:

Prop. 4 $\forall n \cdot n \in (NODES \wedge NETWORK) \Rightarrow$ NLME_status = FAILURE

These properties are all defined as invariants in Rodin platform, the tool generates proof obligations which were successfully discharged using Event-B proof control. The results achieved here are important because our method allows modeling the protocol structure at different levels of abstraction. This model can be further refined in order to

include more details about the protocol implementation, while preserving the correctness of the invariants. In addition, the verification of any extension of the protocol will be straight forward by a refinement of this model. This is useful for future work on verification of the protocol while operating within the WSN environment.

5. Conclusion

In this paper we provide a formal verification method for ZigBee protocol stack. We use Event-B to model the protocol layers and their interfaces, and Event-B invariants to model and verify certain properties for the protocol. For this purpose, a formal link between the semantics of the protocol model and Event-B was established. Even though the ZigBee protocol stack has received considerable attention of analysis and testing using simulation, we believe that formal methods can provide certain level of assurance about the correctness of the protocol that simulation based methods cannot. The approach we present here is based on first-order theorem proving, and therefore, it overcomes the limitations of current methods. In addition it provides functional verification of the ZigBee protocol stack at different levels of abstraction.

The rich expressive language of the first-order logic allows us to formally verify complex properties about the protocol. As future work, an extension of the current layered model to handle a more refined description of the protocol will allow the verification of a concrete implementation of the protocol. In addition, modeling and verifying liveness properties about the ZigBee protocol will add more assurance about its correct design. It will also be interesting to investigate modeling certain features of the protocol operation environment and how it affects its functionality.

References

1. P. Ballarini and A. Miller. Model Checking Medium Access Control for Sensor Networks. In Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, pages 255–262. IEEE Computer Society Press, November 2006.

2. L. Chunqing and Z. Jiancheng. Research of ZigBee's Data Security and Protection. In IEEE International Forum on Computer Science-Technology and Applications, pages 298–302. IEEE Computer Society Press, December 2009.

3. M. Duflot, M. Kwiatkowska, G. Norman, and D. Parker. A Formal Analysis of Bluetooth Device Discovery. International Journal on Software Tools for Technology Transfer, 8(6):621–632, 2006.

4. IEEE. IEEE-SA Standards Board, IEEE 802.15.4-2006 Standard for WPAN, http://standards.ieee.org/findstds/standard/802.15.4-2006.html, 2006.

5. Rodin Platform. http://www.event-b.org, 2010.

6. ZigBee Alliance. WPAN Industry Group, http://www.zigbee.org, 2010.

7. M. Fruth. Probabilistic Model Checking of Contention Resolution in the IEEE 802.15.4 Low-Rate Wireless Personal Area Network Protocol. In Symposium on Leveraging Applications of Formal Methods, Verification and Validation, pages 290–297. IEEE Computer Society Press, November 2006.

8. A. Gupta. Formal hardware verification methods: A survey. Formal Methods in System Design, 1(2-3):151–238, 1992.

9. O. Hasan and S. Tahar. Probabilistic Analysis of Wireless Systems Using Theorem Proving. Electronic Notes in Theoretical Computer Science, 242(2):43–58, 2009. Workshop on Formal Methods for Wireless Systems.

10. M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of the ieee 802.11 wireless local area network protocol. In Process Algebra and Probabilistic Methods, Performance Modeling and Verification, V. 2399 of LNCS, pp. 169–187. Springer-Verlag, 2002.

11. M. Massink, D. Latella, and J-P. Katoen. Model Checking Dependability Attributes of Wireless Group Communication. In Dependable Systems and Networks, pages 711–720. IEEE Computer Society Press, July 2004.

12 C. Metayer, J. Abrial, and L. Voisin. RODIN Deliverable 3.2: Event-B Language. Technical Report Project IST-511599, School of Computing Science, University of Newcastle, UK, 2005.

13. U. Pešoveć. J. Mohoroko, and Ž. Čučej. Upgraded OPEN-ZB 802.15.4 Simulation Model. In ELMAR International Symposium, pages 281–284. IEEE Computer Society Press, September 2010.

14. X. Yang, K. Liu, Y. Cui, and J. Zhang. Modeling and Simulation for Maritime Surveillance Sensor Networks. In IEEE International Symposium on Communications and Information Technologies, pages 215–219. IEEE Computer Society Press, October 2010.

15 J. Abrial, Modelling in Event-B: System and Software Engineering, Cambbridge University Press, 2009.