Reliable Model Checking for WSNs

Christian Appold
Chair of Computer Science V
University of Würzburg
Am Hubland, 97074 Würzburg
Email: appold@informatik.uni-wuerzburg.de

***Abstract*—The capabilities of wireless sensor networks are promising a great future for them. Nevertheless it's necessary to have methods to verify their correct operation before deployment, for widening their range of application and enable their usage in e.g. safety critical environments. One method to examine if systems behave as desired is temporal logic model checking [1], which is a formal verification technique. When verifying wireless sensor networks, some special aspects like the correct modeling of the wireless communication and possibly other components of the sensor nodes are essential. In this paper we report about the verification of a traffic light synchronization protocol at 4-way intersections, where the traffic lights communicate wireless. We present some needful abstraction techniques and discuss particularities in the verification of wireless sensor networks.**

I. Introduction

In this paper we report about the verification of a traffic light synchronization protocol at 4-way intersections and the verification of wireless sensor networks (WSNs) in general. WSNs can consist of a large number of sensor nodes. Because verification of distributed systems is hard, and even a single sensor node and its software could be very complex, verification of sensor networks is a highly non-trivial task. But if they should become deployable in e.g. safety critical environments or areas where they can't be reprogrammed, it's unavoidable to verify that they fulfill their requirements. Otherwise implementation failures can be very costly and cause accidents, which in the worst case could lead to the death of humans. Common approaches to verify the functionality of WSNs are e.g. the use of simulators like TOSSIM [2] or live testing by using testbeds. As a drawback of these methods, they don't verify the desired properties for all possible computations of the sensor network. It is known that especially hard bugs in distributed systems often appear only in a few corner cases. Therefore such complex bugs cannot be detected reliably by these methods. An approach for early stage sensor network verification is the use of model checking. Because model checking isn't easy to apply correctly without some verification experience, it currently isn't widely used in the area of WSNs. Hence we outline in our paper guidelines and abstractions for improving the verifiability of WSNs. This should help to achieve fast and correct verification and make formal verification amenable for the WSN domain.

The paper is organized as follows. In Section II we discuss related work on the field of verification. Section III gives a short introduction in model checking and the symbolic model checker NuSMV [3], which we used for our verification experiments. The verified traffic light synchronization protocol is described in Section IV and Section V shows pitfalls when modeling wireless communication. In Section VI we present useful techniques to model the protocol in the input language of NuSMV. The paper closes with concluding remarks and an outlook to further investigations.

II. Related Work

WSNs often contain stochastic elements (e.g. backoff procedures of communication protocols or elements of the environment). To allow reliable verification of them, these have to be modeled as accurate as possible. PRISM [4] is a probabilistic model checker for analyzing quantitative properties of systems which exhibit stochastic behavior. But though the possibility to model probabilistic elements accurately, its input language is very restricted and probabilistic models are typically more complex, which decreases the limit what can be analyzed. Therefore we have chosen for our work the symbolic model checker NuSMV, which doesn't directly support the specification of probabilistic elements.

In [5] the authors verified the IEEE 802.3 Ethernet CSMA/CD protocol using the symbolic model checker SMV [6]. This protocol is a wired protocol, so they hadn't to deal with the special characteristics of wireless communication. Fehnker et. al [7] verified the LMAC protocol, a medium access control protocol for WSNs using Uppaal [8], a model checker for timed automata. Their property of main interest was detecting and resolving collisions, which they verified for different topologies. They showed that the truth of properties may depend on the network topology. The focus of both papers mentioned above was to verify a communication protocol, whereas our work aims towards verifying networks of sensor nodes considering not only communication, but also implemented functionality. We show that it's often necessary to model communication or possibly other important system components to verify functionality.

III. Model Checking and NuSMV

Model checking is an automatic formal verification technique for verifying properties of finite state systems. A model checker is a tool which, given as input a model of a system and a property of interest formulated in a temporal logic, automatically decides whether the property is valid for all possible computations of the model. To decide if a property is valid, the model checker has to explore all possible system

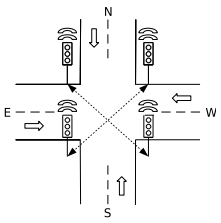

Fig. 1. 4-way road intersection with traffic lights

states exhaustively. As a consequence, the main problem of model checking is the state explosion problem. This problem especially appears in the verification of distributed systems, where the number of possible system states grows exponentially in the number of components. One method to reduce the state explosion problem is to use symbolic model checking [9], which uses BDDs for representing sets of states and the transition relation symbolically, instead of representing them explicitly.

Because symbolic model checking allows the handling of systems with very large state spaces, we used the symbolic model checker NuSMV [3] for our work. NuSMV is a reimplementation and extension of the symbolic model checker SMV. NuSMV permits the description of synchronous and asynchronous systems and has its own input language. For property formulation, NuSMV supports the temporal logics LTL and CTL [1], which extend propositional logic with temporal operators.

IV. The Traffic Light Synchronization Protocol

To show particularities of WSN verification and the usefulness of our abstractions, we developed a simple traffic light synchronization protocol for 4-way intersections. Figure 1 shows a 4-way road intersection with one traffic light for each incoming road. The purpose of the protocol is to synchronize traffic lights which communicate wireless. Thereby one of the main targets is to ensure that only diagonally arranged traffic lights are allowed to show green at the same time, to prevent accidents. A simplified state diagram of the control flow of the protocol for a single traffic light can be seen in Figure 2. The conditions for the feasibility of transitions and also transitions without state changes have been omitted for clarity. These conditions consist of combinations of values of local state variables and types of incoming messages from other traffic lights.

In the control states *red* (initial state), *yellow* and *green*, the protocol triggers its lights to get the corresponding color. If a traffic light is in the state *red* or *green* and gets a message to change its light color (e.g. from a traffic measurement sensor at the road), the protocol sends a command to transmit a light change request to the underlying MAC protocol. When the MAC protocol confirms the sending of the message, the transmitting traffic light changes its control state to an acknowledgement receiving state. If no communication errors occured, the diagonally arranged traffic light at the

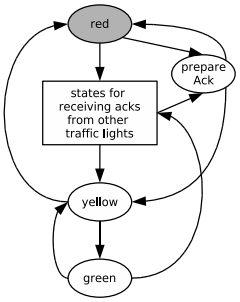

Fig. 2. Control flow state diagram of the traffic light synchronization protocol for a single traffic light

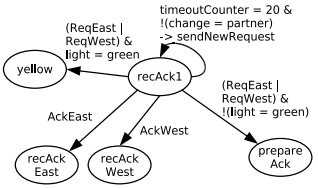

Fig. 3. Control state *recAck1* with transitions

intersection subsequently changes its control state also to an acknowledgement receive state. There are four such states in the protocol, but for clarity they are summarized in Figure 2 into one state. The other two traffic lights change their state to *prepareAck*, if their lights are red at the moment. When their lights show green, they first change their light color to red and then go to this state. In the state *prepareAck* they prepare and send an acknowledgement transmit request to the MAC layer. They leave this state, when they get the confirmation that the request has been sent.

Figure 3 shows exemplary transitions and next states of the receiving state *recAck1*. A traffic light changes to this control state from the states *red* and *green*, when it receives the confirmation that a light change request has been sent from the MAC layer. The Figure is drawn with identifiers for the traffic lights at the incoming roads from north and south, whose behavior is symmetric in this state. If the traffic light receives an acknowledgement from the traffic light at east or west and there hadn't been a communication error, it changes its state to *recAckEast*, and *recAckWest* respectively. Because it could be possible, that two light change requests from traffic lights collide, or a traffic light hasn't received a send request from another traffic light, this had to be considered in the protocol. Therefore the protocol uses a timeout counter, which is incremented every step when the traffic light is in state *recAck1* until the timeout limit is reached. Also the protocol uses a variable *change*, which has the value *partner*, when the traffic light received a change request from the diagonally arranged traffic light at the intersection.

By executing the transition with the variables *timeoutCounter* and *change* in Figure 3, a command to transmit a traffic light change request is send to the MAC layer. The state changes from *recAck1* to *yellow* or *prepareAck* are needed,

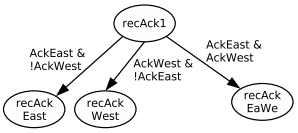

Fig. 4. Receiving of acknowledgements in *recAck1* without collisions

because a traffic light could be in the state *recAck1* while a change request from another traffic light could have been sent. This behavior could appear because of collisions or transmission errors, when sending a light change request. If the traffic light shows green, it first changes its state to *yellow* and then *red*, before going to state *prepareAck*. Otherwise it changes its state to *prepareAck* to initiate the transmission of an acknowledgement. Two different types of acknowledgements can be sent by a traffic light. One type to acknowledge a request of the diagonally arranged traffic light and another type to acknowledge traffic light change requests from the other two traffic lights. After reception of all necessary acknowledgements without timeout, the initiating traffic light sends a sendComplete message to inform the other traffic lights about successful light change and changes its light color accordingly.

V. Communication Modeling Pitfalls

In this section we show how neglecting wireless communication characteristics can circumvent the detection of design errors through model checking.

A. Nonobservance of collisions and impossibility to listen during sending

Here we show how disregarding collisions together with nonobservance of impossibility to listen during sending can prohibit the detection of design errors. Figure 4 shows an example model of the outgoing transitions of state *recAck1* where acknowledgements are received for the traffic light at north, without considering collisions. In a real world deployment acknowledgements from the traffic lights at east and west cannot arrive at the same time, because there would be a collision in wireless communication. When verifying the property that no deadlock exists for this model of the protocol, it could be verified by the model checker as correct even without using a timeout and request resend mechanism in the acknowledgement receiving states. If the acknowledgements from the traffic lights at east and west collide, without a timeout and request resend mechanism all traffic lights are stuck in their states. When ignoring that listening in wireless communication usually isn't possible during sending, the other parts of the protocol could be implemented for the model checker in a way, that these deadlock doesn't appear on any computation path of the model. As a consequence the model checker can't find the deadlock.

B. Nonobservance of variations in radio wave propagation

Variations in radio wave propagation, e.g. through changing environmental conditions or obstacles, can cause situations where a message sent by a traffic light could be received only by a subset of the desired receivers. In an early version of the protocol we used only one type of acknowledgements. During verification runs considering collisions but without variations of radio wave propagation, we couldn't find a counterexample for the property that only diagonally arranged traffic lights are allowed to be green at the same time. When we inserted variations of radio wave propagation in our verification model, we could find computation paths where three traffic lights could show green at the same time.

This behavior could appear, if all traffic lights showed red and the south traffic light changed its control state from *recAck1* to *prepareAck*, because of a light change request from the west traffic light. The light at north didn't receive this request. Subsequently the light at north did send a light change request and the south traffic light approved this by sending an acknowledgement, whereas its state change to *prepareAck* has been caused by a change request from the west traffic light. As a consequence the south traffic light switched to control state red and the north traffic light to state green. After that the north traffic light changed its state to *red* and then to *prepareAck*, because the traffic light at east transmitted a light change request to green. Then the traffic light at south, which didn't receive this request, transmitted a light change request which was received by north. In the old protocol the traffic light from north was able to change its variable *change* (see Section IV and Figure 3) to the value *partner* in state *prepareAck*. Therewith it could execute the transition from state *prepareAck* to state *yellow*. Subsequently the west traffic light send a request to change its lights color to green and the north traffic light acknowledged this and also changed its color to green.

VI. Modeling suggestions for reliable verification

In this section we describe suggestions to model the characteristics of wireless communication for the model checker NuSMV. We present suitable abstractions for modeling variations in the radio range, transmission errors, the possibility of packet collisions and the circumstance that collisions normally cannot be detected by the sending nodes. Their use allows the reliable verification of WSNs.

Verification models for NuSMV consist of several processes, which can be executed completely synchronous or completely asynchronous. For our verification runs we chose synchronous execution, because of the lower complexity of the verification model and the lower verification effort for the model checker. To model the wireless communication channel, we used DEFINE statements from the input language of NuSMV. These work like macros. We therewith specified for each traffic light defines for channel free, collision and one for each message of any other traffic light. The value of these defines is determined by the current control states of the traffic lights and the values of input variables. For this purpose we inserted the control states *sendReq*, *sendAck*, *sendAckPartner* and *sendComplete* in our verification model.

```
DEFINE
  free := !(sendReqEast | sendAckEast | sendAckPEast | sendCompleteEast | ...);
  collision := ((sendReqEast | sendAckEast | sendAckPEast | sendCompleteEast) &
                ((sendReqWest | sendAckWest | sendAckPWest | sendCompleteWest) | ...));
  sendReqEast := (east.cState = sendReq) & inputEast;
  sendCompleteEast := (east.cState = sendComplete) & inputEast;
```

Fig. 5. Example DEFINE commands for channel modeling

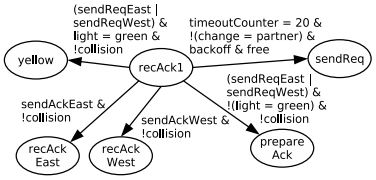

Fig. 6. Control state *recAck1* with transitions and abstractions

A traffic light in our model transmits, if its control state currently equals one of these new states. Figure 5 shows examples of DEFINE commands for channel free, collision occured and some incoming messages for the traffic light from north. A big advantage of this modeling, beneath its correctness and easy implementation is, that no new state variables are needed for it, because defines work like macros. In contrast the channel model of [7] introduces new state variables, which in large networks can affect the verifiability. The domain of the defines we formulated is Boolean. For *free*, the define holds the logical value true if no other traffic light at the intersection currently sends a message which north receives, otherwise false. To detect collisions, the define *collision* takes the value true if two or three other traffic lights send simultaneously messages which north receives. The last two defines in Figure 5 indicate if the traffic light for the incoming road from east sends a sendRequest or sendComplete message, which north receives correctly. Their logical value depends on the current control state of the traffic light at east and the value of the Boolean input variable *inputEast*. Input variables in NuSMV get their value from the verification environment through the model checker, which assigns all possible values to them in every state. They are used in the defines *sendReqEast* and *sendCompleteEast* for modeling variations in radio range and transmission errors.

Figure 6 shows the outgoing transitions and successor states of state *recAck1*, as in Figure 3, with our modelings and abstractions for wireless communication. To include collisions in our model, we added the condition *!collision* to transitions where messages have to be received for their feasibility. With our experiments we intended to verify the protocol for wireless communication and a MAC protocol with carrier sense and a randomized backoff procedure. Thus, for reliable verification we needed a suitable model for it, which preserves all possible computations and keeps the state space small. We developed an abstraction using Boolean input variables. Through adding conditions about a certain value of an input variable, we restricted the feasibility of transitions which lead to a state which models the sending of a message. In the transition from state *recAck1* to *sendReq* in Figure 6 this is the input variable *backoff*. Additionally we added the condition that the define *free* of the communication channel model also has to be true for feasibility of the transition. In the transition *free* is used to model the carrier sense mechanism and the input variable is responsible for modeling all possible behaviors of the backoff procedure.

VII. Conclusion and Outlook

In this paper we reported about verification of WSNs. We developed a traffic light synchronization protocol for 4-way intersections and showed some particularities in verifying WSNs. One conclusion is, that often system components, like e.g. synchronization protocols, cannot be verified isolated in WSNs. Frequently, a model of the communication protocol and models of other sensor node components, like e.g. timers or even parts of operating systems, are also necessary. A big challenge is to find suitable models which don't affect the verifiability (by leading to the state explosion problem) but describe the intended behavior correctly. Therefore especially for non-verification experts, suitable and faultless abstraction techniques should be available. In this paper we presented a way to model a communication channel with collisions, transmission errors and variations of radio wave propagation for the model checker NuSMV. Additionally we presented a model of a backoff procedure.

For future work we want to develop abstractions for several other sensor network components. Additionally we will examine the impact of different and dynamic topologies together with varying radio ranges on verification results.

References

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

[2] P. Levis, N. Lee, M. Welsh and D. Culler, *TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications*, In Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys 2003).

[3] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani and A. Tacchella, *NuSMV 2: An OpenSource Tool for Symbolic Model Checking*, In Proceedings of the 14th International Conference on Computer Aided Verification (CAV 2002). Springer, 2002.

[4] A. Hinton, M. Kwiatkowska, G. Norman and D. Parker, *PRISM: A Tool for Automatic Verification of Probabilistic Systems*, In Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2006). Springer, 2006.

[5] V. G. Naik and A. P. Sistla, *Modeling and Verification of a Real Life Protocol Using Symbolic Model Checking*, In Proceedings of the 6th International Conference on Computer Aided Verification (CAV 1994). Springer, 1994.

[6] K. L. McMillan, *Getting Started with SMV*, User's Manual, Cadence Berkeley Laboratories, USA, 1998.

[7] A. Fehnker, L. van Hoesel and A. Mader, *Modelling and Verification of the LMAC Protocol for Wireless Sensor Networks*, In Proceedings of the 6th International Conference on Integrated Formal Methods (IFM 2007). Springer, 2007.

[8] G. Behrmann, A. David, K. G. Larsen, J. Hakansson, P. Pettersson, W. Yi and M. Hendriks, *Uppaal 4.0*, In Quantitative Evaluation of Systems (QEST 2006). IEEE Computer Society, 2006.

[9] K. L. McMillan, *Symbolic Model Checking: An approach to the state explosion problem*, Ph.D. thesis, Carnegie Mellon University, 1992.