A Formal Executable Specification of the GinMAC Protocol for Wireless Sensor Actuator Networks

Admar Ajith Kumar Somappa*†, Lars Michael Kristensen*, Knut Ovsthus*
*Faculty of Engineering, Bergen University College, Norway
{aaks, lmkr, kovs}@hib.no
†Faculty of Engineering and Science, University of Agder, Norway

***Abstract*—The inception of Wireless Sensor Actuator Networks from the existing Wireless Sensor Networks domain was mainly to satisfy the requirements of the automation applications. Automation applications can range in a broad domain from industrial automation, home automation to automation in body area networks and have specific real-time requirements differing from each other based on the criticality of the application. Given the complex nature of WSAN, efficient techniques are required to be employed to obtain error-free protocol designs. Formal specification languages like the Coloured Petri Nets assist designers to capture this complexity with its expressive specification language in combination with Standard ML. GinMAC is a Medium Access Control (MAC) protocol that aims at satisfying the real-time requirements laid down by the Wireless Sensor Actuator Networks applications. In this article, we provide a formal executable specification for the GinMAC protocol. With this formal executable we precisely capture the abstract features of this protocol and analyze it. This formal executable is platform independent and can be used as a basis for further analysis of the protocol or any protocol extensions that are made to this protocol. The platform independence allows us to convert this model to various other analysis tools including model checkers, network simulators and hardware emulators.**

I. Introduction

Wireless Sensor Actuator Network (WSAN) [1] has had significant effect on the process industry by greatly reducing the operating costs. It has found applications in various industrial scenarios, for monitoring and supervision, sometimes in closed loop control where the entire sensing to actuating activity is automated. The monitoring and supervision activity relates to the general application of WSN. The closed loop control particularly introduces real-time requirements and challenges to the protocol design for WSAN with tighter bounds on delay and requires high reliability. Apart from industrial scenarios, it is also used in applications [2] like home automation systems and in Body Area Networks. WSAN consists of heterogenous devices in the form of sensors and actuators connected via wireless links. These devices coordinate with each other to achieve sensing and actuation tasks for a given application. The sensors and actuators used are generally low cost, low power computation devices with radio modems. Actuators in some scenarios are resource rich and wire powered. The sensors monitor a given area, collect data and feeds it to the actuators in real-time, either directly or via a central entity known as a *Sink*. In a network topology containing a sink, the collected data may be analyzed and consolidated before being communicated to the actuators. Actuators then act on this collected data as required.

GINSENG [3] is an EU project aimed at designing performance controlled WSAN. GINSENG proposes a solution for WSAN with bounded delay and reliability requirements. It assumes pre-planned network deployment as a step towards building a sensor actuator network with reliable communication. As a part of the GINSENG project, an architecture was proposed including various software components, network protocols, operating system and a middleware to handle data communication and processing. One major element of the network protocols is the GinMAC [4] protocol that was proposed during this project. The authors claim to achieve 97% reliability results in terms of packet delivery under relevant conditions. The protocol was designed to satisfy specific delay bounds both for sensors and actuators. The main features of this protocol include *Off-line Dimensioning*, *Exclusive TDMA* slot allocation and *Delay Conform Reliability Control*. It provides end-to-end guarantee and the reliability control provided allows for *predictability* of the resulting network solution. Although GinMAC was proposed for a specific application in the GINSENG project, its important features could result in optimal solutions for various other scenarios. GinMAC is limited in terms of its network scalability since it was designed for very small number of nodes in the order of 25 [4]. This limitation can be overcome by extensions to the protocol thus making it applicable to wide deployment scenarios.

WSANs are complex concurrent real-time systems, thus analysis of these systems in itself is a complex procedure. Industries being clearly business oriented expect the implementation to realize the pre-set requirements efficiently. The possible difference between the specification and implementation can be minimized by using model-based verification and validation techniques. Also, the wireless channel is a pure non-deterministic entity in itself. Thus, to reduce the uncertainties in the implemented solution it is essential to start with precise design models. We use an expressive formal specification language like Coloured Petri Nets (CPN) to specify GinMAC thus providing a concise design to be analyzed. Modeling of the GinMAC protocol is done based on the article [4] and the example presented in it. Further, this platform independent design can be used as a common platform to be then transformed to other specification languages for further validation and verification. Sensor networks have been designed and validated in tools such as Uppaal in the past [5], [6]. With the added strength of probabilistic/stochastic modeling to powerful tools like Uppaal Statistical Model Checker [7] and PRISM [8], model checking could produce more valuable results. Inherently, the model checkers do not scale well to large numbers but are rather used for fine tuning the model to obtain optimum performance. Thus for *scalability* verification, network simulators are used. Based on the current user base and available facilities, OMNET++ based MiXiM [9] and Castalia [10] are among the few very important network simulators and can be used to assess *scalability*. After the current design is validated, analyzed and verified via simulation and model checking, we can proceed with the implementation to consolidate all the results obtained. Also, this implementation could be partly obtained from the CPN model resulting in a concise implementation.

The focus of this article is to provide a platform independent formal executable high level model specification for the GinMAC protocol. This modeling is done using CPN. The rest of the article is organized as follows: firstly we give a brief introduction of the GinMAC protocol and Coloured Petri Nets basics in section II. For extended knowledge on the CPN constructs one can refer to [11]. Modeling aspects for the GinMAC protocol in a WSAN context is described in detail in section III. In the modeling section, the constructs that are important to understand the model are explained in detail and the rest are omitted owing to limited space. We conclude along with a short discussion on the related work in this domain. This article assumes prior knowledge of WSN MAC protocol basics.

II. Background

A. GinMAC

GinMAC is a Timed Division Multiple Access (TDMA) MAC protocol, proposed for the GINSENG project with concrete requirements of reliable and timely delivery of data. Routing function in terms of tree topology forwarding is also embedded into this protocol via cross layering. The network topology is tree based. GinMAC also address energy efficiency issues via efficient duty cycling by introducing unused slots in the GinMAC frame. GinMAC provides three main features that is designed to satisfy the requirements.

Off-line Dimensioning. For GinMAC, authors assume that network deployment is known. Thus, scheduling computations and other required static protocol operations can be made apriori. A TDMA schedule is created with a given frame length of *F*. This frame is then divided into three types of slots: *basic*, *additional* and *unused*. The *basic* slots are for regular data transfer. The *additional* slots are used to increase reliability. Lastly, the *unused* slots are used to achieve low duty cycle. Delay requirement for the protocol is specified as D_S, for the maximum delay allowed for sending data from any sensor to the sink and D_A, for the maximum delay allowed for sending actuator data from the sink to the actuators. The offline dimensioning is done according to D_S and D_A in mind and requires to meet the delay requirement $F < min\{D_S, D_A\}$.

Exclusive TDMA. GinMAC assumes data transmission of maximum length and acknowledgement is accommodated in a single slot. The slots (including additional) used are exclusive, and cannot be re-used by other nodes in the network. A particular topology envelope is defined with maximum hop distance *H* of 3 and the maximum number of nodes that can be accommodated in a single network managed by one sink is 25. Basic slots are used for both types: sensor and actuator data, and sometimes for configuration commands if required. The sensor data is sent upstream from sensors to the sink and the actuator data is sent downstream from the sink to the actuators. Given the tree structure, the nodes having children need to have basic and additional slots for its children as well. Sensor data from the leaf nodes is sent upstream to their parents and each receiving node sends it to its parent node. This process continues until the data reaches the sink node. The actuator data is sent downstream from the sink to the actuator node by forwarding it to one of its children nodes, under which the actuator node exists. The process continues until the actuator data reaches the actuator node.

Delay Conform Reliability Control. Wireless channels are highly susceptible to interference. Just using basic slots does not satisfy the underlying requirements. Additional slots are used for this purpose to increase reliability. The number of additional slots required per basic slot depends on the channel conditions. The channel characteristics are captured by running some experiments in the deployment area and calculating worst-case link reliability. During the protocol operation links that have successful data exchange are marked as *good links*. GinMAC uses two methods to specify good links. The first one uses Packet Reception Rate (PRR) with good links defined as the ones that have PRR better than a given threshold. The second method calculated the burst errors occurred during the experiment to calculate worst case *link reliability*. A good link was then defined as a link with maximum of B_{max} consecutive transmission errors and minimum of B_{min} consecutive successful transmissions in between two burst errors. The rest of the slots in the frame apart from basic and additional slots are used as unused slots. Various topology control mechanisms are also proposed for new nodes to be able to join the network provided that the number of nodes does not exceed the maximum. We assume static network with no node joining procedure for simplicity.

The topology used in modeling example is shown in figure 1 and consists of 4 sensors, 1 actuator and 1 sink. Each sensor and actuator has a basic slot in each GinMAC frame. For a link with $B_{max} = 1$ and $B_{min} = 1$, GinMAC defines 1 additional slot per sensor/actuator. For a given delay requirement say $D_S = 1s$ and $D_A = 1s$ we can construct a GinMAC frame with ($F < min\{D_S, D_A\}$) = min{1,1}= 1s). One GinMAC frame accommodates slots required by each node in the network including sensor data, actuator data and sleep slots. All the data communications from every sensor to sink and sink to every actuator is carried out within this frame and is repeated for each frame. For a CC2420 transceiver with slot size of 10ms and with the frame required to fit within 1s,

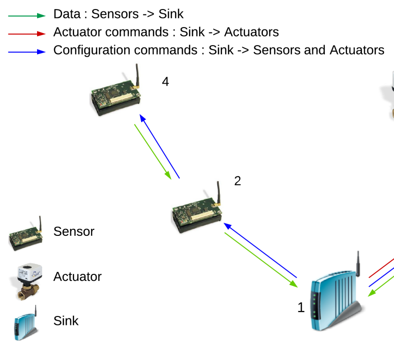

Data : Sensors -> Sink
Actuator commands : Sink -> Actuators
Configuration commands : Sink -> Sensors and Actuators
1
2
3
4
5
6
Sensor
Actuator
Sink

Fig. 1: GinMAC Topology

we have 100 slots per frame. We assume that actuators only receive data and do not send any data. Thus for upstream data (sensors to sink) we have 6 basic slots and and 6 additional slots. For downstream (sink to actuators) we have 2 basic slots and 2 additional slots, and one slot is reserved for configuration commands. Remaining 83 slots in the GinMAC frame are used as unused slots for energy efficient duty cycling. GinMAC frame for the current example can be seen in figure 2.

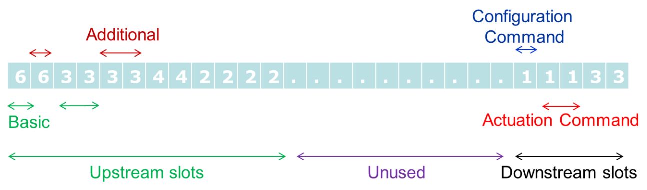
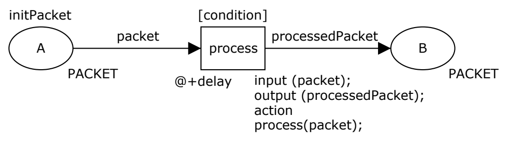


Fig. 2: GinMAC frame

B. Coloured Petri Nets

CPN is a discrete event based extension of Petri Nets combined with Standard ML, a functional programming language, thus making it more expressive. CPN is designed to model complex concurrent systems and has been used in various application domains [12]. CPN includes the notion of global time and allows tokens to be time stamped by defining the data types as timed. *CPN Tools* [12] is the software with its own graphical user interface to enable easy modeling. CPN Tools also includes other tools for state-space analysis, hierarchical model construction, and simulation-based performance analysis. The hierarchy allows for modular representation, breaking up complex parts into further smaller parts representing some specific function.



```
input (packet);
output (processedPacket);
action
process(packet);
```

Fig. 3: CPN example

CPN uses graph like constructs with prime components including places, transitions, arcs and tokens. An example is shown in figure 3, with two places labeled *A* and *B*. A place is of a defined data-type, *PACKET* in this example. A starting place in a net is initialized with certain data values called as tokens in this context. In our example *initPacket* defines the initial tokens of place labeled *A*. A transition labelled *process* connects these two places with arcs. The tokens are passed between places and transitions using these arcs. Transitions consume tokens from incoming edges and produces/creates tokens on output edges. Between the process of consuming and creating new tokens, we can perform certain actions on the input tokens and produce output tokens accordingly. In our example we process the input token using *processPacket* function and produce *processedPacket* as output. The execution of the transitions also known as firing can be decided using certain guard conditions which evaluate to either true or false, *condition* in this case. Guard conditions/functions are enclosed within square brackets. The firing of transitions can only take place if there are tokens on all input arcs and the guard condition evaluates to true (boolean). Given the use of global time and the fact that tokens are timed, this time can be incremented either on the arcs or on transitions. We add *delay* as an indication of the processing time taken by the *process* transition, *delay* here is an integer value.

III. GINMAC CPN MODEL

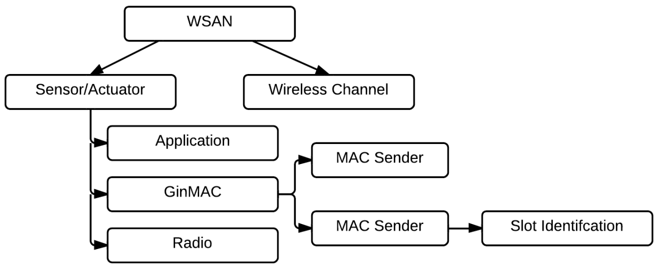


Fig. 4: Module Hierarchy of the Wireless Sensor Actuator Networks

Modeling approaches at the design level are high level abstracts of the actual implementation. In this article, the modeling in CPN focuses on certain details of the sensor/actuator nodes, with brief description of other modules. Detailed modeling constructs of CPN can be found in [11]. The modeling hierarchy is shown in figure 4 with WSAN at the top level with sensor/actuator and wireless channel. The sensor/actuator is further subdivided into Application, GinMAC, and Radio modules. The GinMAC function which is the focus of this paper is further divided into MAC receiver and MAC sender module which has a submodule slot identification and are responsible for all MAC functions. The CPN model for WSAN is shown in figure 5 with sensors and actuators being represented using a single substitution transition. A substitution transition is an abstract representation of a more detailed net, it represents the module hierarchy in CPN. Substitution transition in-turn can have further substitution transitions if required. We represent both sensors and actuators in a single module since the focus is on the communication part and both sensors and actuators have the same structure from this perspective. Separate channel for receive and send is more a representative model than actual to facilitate easy understanding of packet flow via simulation.

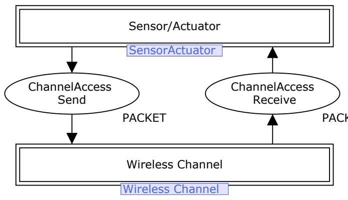


Fig. 5: Wireless Sensor Actuator Network

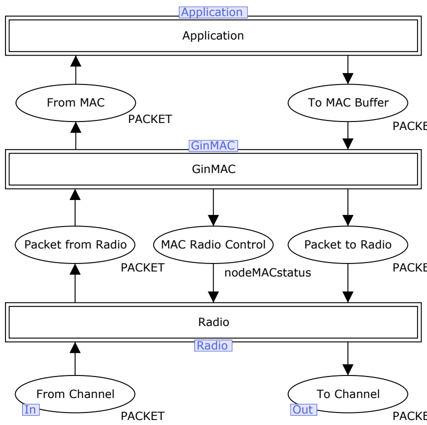


Fig. 6: Sensor/Actuator Module

Sensor/Actuator module. The Sensor/Actuator module is shown in figure 6. In the sensor/actuator module, the GinMAC module is described in sufficient detail and other functions like application, radio (physical layer) and wireless channel are mostly abstract. One of the nodes in the network is designated as a *sink*, which acts as the root node of the tree topology. The application layer for the sensor nodes generates sensor data and the sink node takes in all these sensor data and generates actuator data. Configuration data can be generated as well. Configuration data generated by the sink can be either periodic (recurring) or sporadic (occurring at random instances). We have constructed a periodic model, with configuration commands being sent every 10 frames. Alternatively, this can also be modeled sporadically. Note that the different sensors are differentiated via their node identifiers. The declaration of the node identifier (nodeID) along with the packet declaration in CPN is defined as follows and is explained in detail in the next paragraph.

```
colset nodeID = INT with 0..maxNodes;
colset source = nodeI;colset destination=nodeID;
colset sequenceNo = INT with 0..maxSequenceNo;
colset packetType = with DATA|ACK|CONFIG;
colset PACKET = product
source * destination * packetType * sequenceNo timed;
```

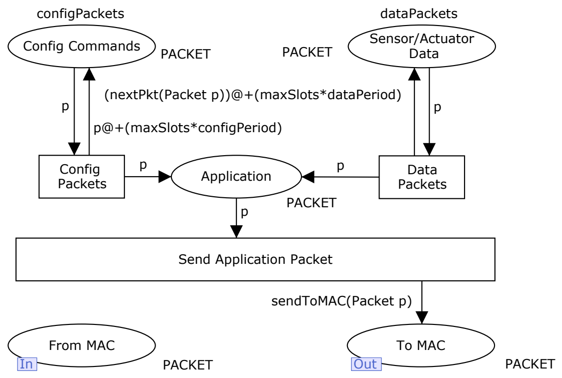


Fig. 7: Application module

Application module. The Application module is abstract and we skip network module since routing is defined by the GinMAC protocol via cross layering. The tree topology defined for the network is used as the routing specification where sensors route the data upwards to the sink via intermediate sensors/actuators and sink routes data to actuators via intermediate sensors/actuators. The CPN model for the Application module can be seen in figure 7. Time is an important entity in WSAN, thus we use time-stamps in the GinMAC module. Packets can be easily differentiated according to their time-stamps and sequence numbers if required. A packet is defined as a data-type with 4 fields: source, destination, packet type, and sequence number as shown in the previous code segment, and it is defined as timed in order to use the global time concept. Source, destination and sequence numbers are of integer (INT) type and packet type defines three possible types, DATA (sensor data), ACK (acknowledgement) and CONFIG (configuration packets). The product keyword in PACKET colour set defines all possible combinations of colour sets that follow. The values *maxNodes* and *maxSequenceNo* defines the maximum nodes in the network and maximum sequence number of packets allowed. In the GinMAC protocol, each sensor has to send DATA in their slots irrespective of whether there has been any sensor reading or not. Thus there is DATA sent in every frame. This periodicity of the packets is maintained by time-stamping the next packet (DATA and CONFIG) with the delay required to satisfy their periodicity. The variable p in figure 7 represents packet and the time-stamp $p@+(maxSlots*dataPeriod)$ represents restoring the token at the place "Sensor/Actuator Data" with a time-stamp that would enable it in the next frame. Similar time-stamp delay method is used on the CONFIG packets, $p@+(maxSlots*configPeriod)$. *maxSlots* represents maximum number of slots in each frame. The constant *dataPeriod* represents the period duration after which the data appears again, i.e in the next frame. The constant *configPeriod* represents the period duration after which configuration command is to be sent again.

GinMAC module. The GinMAC module is shown in figure 8. Packet arriving from the Application module is sent to the MAC sender module where packet (DATA and CONFIG)

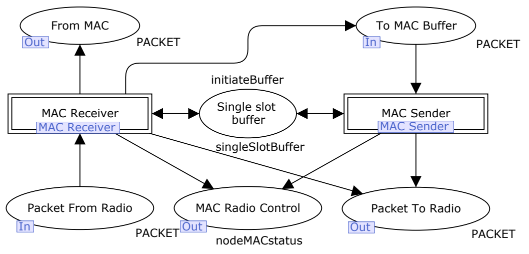


Fig. 8: GinMAC module

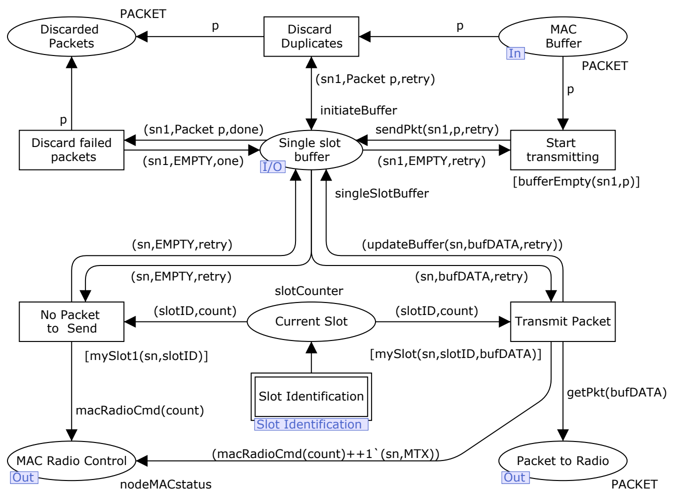


Fig. 9: MAC Sender Module

transmission is handled. Packets arriving from the radio module are serviced by the MAC receiver module, which also is responsible for sending acknowledgements (ACK) for all DATA packets received. Each packet transmission is accompanied by a radio control command from the MAC to switch the radio to transmission state. The GinMAC module includes a single slot buffer that is used by both of its submodules MAC Sender and MAC Receiver. This single slot buffer saves one packet for each node that is to be transmitted or is being transmitted. It is defined using the data-type with three fields, the node identifier, buffer, and the retransmission counter. Buffer data-type is a special type which is a union of a packet type and a value EMPTY representing empty buffer. Thus the buffer has either one packet in it or is empty.

MAC Sender module. The MAC Sender module is shown in figure 9. For simplicity some parts have been moved and may not be exactly in position with respect to figure 8. The incoming packets from the MAC buffer are transferred to a single slot buffer, where the packet is stored until it reaches the destination or maximum retransmission attempts are completed. After maximum retries the DATA packets are discarded and are recorded as discarded packets. When a packet is being sent over the channel, the MAC module issues a radio command to switch to transmission mode. Duplicate packets arriving at the MAC buffer due to failure of ACK packets are discarded by looking up the single slot buffer. Also, when a node has the current transmission slot and has no packets to send, a SLEEP radio command is sent to the Radio module indicating it to switch off radio parts via *macRadioCmd* function as shown in figure 8. The MAC Sender module contains a sub-page for Slot Identification which keeps count of the time and represents the current slot in the GinMAC frame.

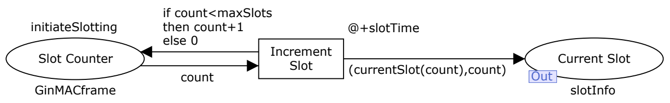


Fig. 10: Slot Identification Module

Slot Identification module. Given the *Offline Dimensioning* feature, the slot division is done statically and is stored on each of the nodes. Also all nodes are time synchronized since GinMAC follows a TDMA schedule. In the GinMAC model, slot division is stored statically and depending on the current time and their schedule the sensor/actuator nodes go to sleep or wakeup. This is represented by the Slot Identification module in the CPN model for the GinMAC protocol shown in figure 10. We start at time zero with the token initialized by the initiateSlotting value and the Increment Slot transition increments the slot counter after the given slotTime which is 10ms in our example. The nodeID of the node which has the current slot is calculated and sent over the arc to the Current Slot place along with the counter value (count). This counter value is used do decide the radio commands to be sent by the GinMAC modules of other nodes in the network. The counter value is reset when the GinMAC frame is completed, which is assessed using the value maxSlots in the conditional statement on the arc. maxSlots indicates maximum allowed slots in the GinMAC frame. The *Delay Conform Reliability Control* feature is represented by the additional slots given to each node in the network for transfer of DATA (sensor/actuator).

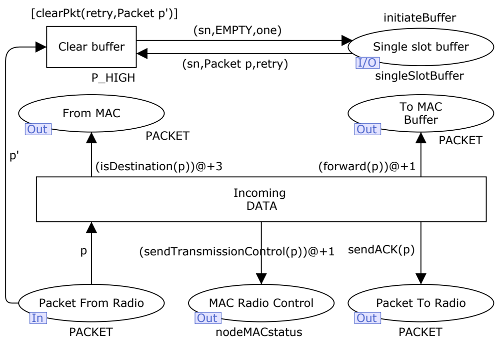


Fig. 11: MAC Receiver Module

MAC Receiver module. Incoming DATA is handled in this module (see figure 11), ACK is sent for every successfully received DATA and the DATA is then forwarded to the Application module by transferring it to the *from MAC* output port. For each ACK sent, the MAC Receiver module of the receiving node sends in a radio command to switch radio to transmission mode. DATA packets that have to be forwarded to other sensor nodes are passed on to the MAC buffer and are handled at the MAC Sender module and ACK is sent for the same from the MAC Receiver module. When an ACK is successfully received for a DATA packet, it is then cleared from the single slot buffer and this transition is given a higher than normal priority. This is to prevent the non-determinism when after maximum retries is completed and ACK is received for a packet, the delete packets transition and clear buffer transition are both enabled simultaneously.

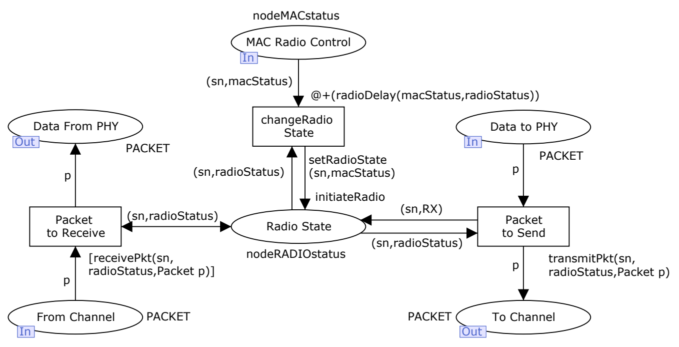


Fig. 12: Radio Module

Radio module. The Radio module is modeled in an abstract way and is shown in figure 12. The incoming MAC control messages (radioCommand) are used to change the state of the radio between three possible states, sleep (SLEEP), receive (RX) and transmit (TX). Two processing transitions are modeled to process sending and receiving of the packets from the wireless channel. Both sending and receiving processes need to have the proper radio state to go through their packet processing, which being TX for transmission and RX for receiving. Essentially, the Radio module here is clearly abstract but could be useful in modeling if certain techniques, like the Carrier Sense Multiple Access (CSMA), are to be used in the model or any other modifications that involves radio functionality being managed by the MAC protocol.

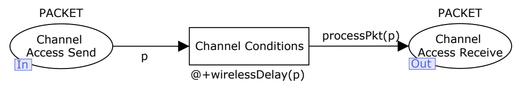


Fig. 13: Wireless Channel

Wireless Channel module. On the basic level the wireless channel receives a packet and sends it back to the Sensor/Actuator module as shown in figure 13. But between this sending and receiving various phenomena occur. The wireless signal is subjected to various effects, particularly path-loss, interference, shadowing, and multi-path fading. Thus modeling a wireless channel in itself is a very complex problem. Over the years, several proposals have been made to mathematically model the wireless channel, e.g. the channel model for 802.15.4 [13]. These mathematical models can be used as a basis to model the wireless channel model in CPN, we omit this part for simplicity. For simple simulation purposes Packet Error Rate (PER) can be used that can be calculated for the range of possible Bit Error Rate in industrial environment which is said to be 10^{-2} to 10^{-6} [14]. PER is the probability with which each packet in the network can fail during transmission. Similarly BER is the probability with which each transmitted bit can fail. These two terms are widely used in WSN.

IV. Related Work and Conclusion

Martínez [15] proposed a formal specification and design technique for WSAN. The article concentrates on the real-time scheduling of tasks on the CPU part, and uses CPN modeling formalisms to represent various architectural components and its working. Another work on using formal modeling tools is an article regarding modeling biomedical sensor networks [5] which concentrates on modeling the Chipcon CC2420 radio transceiver and uses constant bit error rate to model failure of packets. They also verify certain quality of service requirements for the network. In [16], authors show the capability of generic Petri Nets by designing a minimal model of the EQ-MAC protocol. The study also shows that Petri Nets is an efficient tool for modeling and to carry out performance analysis to a certain extent. Heidarian et al. [6] used Uppaal for model checking the clock synchronization processes in WSN using Uppaal and problems were identified for certain topologies.

In this article, we have modeled a formal executable specification of the GinMAC protocol. GinMAC is aimed at Wireless Sensor Actuator Networks with prime requirements including predictable performance, reliable and timely delivery in addition to energy efficiency. The offline scheduling and methods to alleviate packet losses and increase reliability in GinMAC supports *predictability*, an important requirement in industrial perspective. We have also verified the behaviour of the protocol model via simulation and the state-space analysis features of CPN. This formal executable can be used as a basis for further protocol extensions that can be proposed using GinMAC features and also for comparative analysis purposes. This executable model also serves as a common platform for further conversions to other analysis tools specialized in *predictability* analysis, *scalability* analysis and various other performance analysis as required by the application. Also, there have been some efforts towards automatic code generation from CPN models [17], [18], which assist in reducing the errors made when converting design to implementation and finalizing the model to precise implementation. Thus, we can also obtain automatic validation of the implementation code against the initial design model reducing the overall time requirement in building protocol extensions.

References

[1] I. F. Akyildiz and I. H. Kasimoglu, “Wireless sensor and actor networks: research challenges,” *Ad Hoc Networks*, vol. 2, no. 4, pp. 351–367, 2004.
[2] R. Verdone, D. Dardari, G. Mazzini, and A. Conti, *Wireless sensor and actuator networks: technologies, analysis and design*. Academic Press, 2010.
[3] “www.ict-ginseng.eu.” GINSENG Project.

[4] P. Suriyachai, J. Brown, and U. Roedig, “Time-critical data delivery in wireless sensor networks,” in *Proceedings of DCOSS*, vol. 6131, pp. 216–229, 2010.

[5] S. Tschirner, L. Xuedong, and W. Yi, “Model-based validation of QoS properties of biomedical sensor networks,” in *Proceedings of EMSOFT*, pp. 69–78, 2008.

[6] F. Heidarian, J. Schmaltz, and F. Vaandrager, “Analysis of a clock synchronization protocol for wireless sensor networks,” *Theoretical Computer Science*, vol. 413, no. 1, pp. 87–105, 2012.

[7] A. David, K. G. Larsen, A. Legay, M. Mikučionis, D. B. Poulsen, J. Van Vliet, and Z. Wang, “Statistical model checking for networks of priced timed automata,” in *Proceedings of FORMATS*, pp. 80–96, 2011.

[8] M. Kwiatkowska, G. Norman, and D. Parker, “Prism: probabilistic model checking for performance and reliability analysis,” *SIGMETRICS Performance Evaluation Review*, vol. 36, no. 4, pp. 40–45, 2009.

[9] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. Haneveld, T. Parker, O. Visser, H. S. Lichte, and S. Valentin, “Simulating wireless and mobile networks in omnet++ the mixim vision,” in *Proceedings of SIMUTools*, p. 71, 2008.

[10] A. Boulis, “Castalia, a simulator for wireless sensor networks and body area networks,” *National ICT Australia Ltd, Australia*, 2009.

[11] K. Jensen and L. M. Kristensen, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. 1st ed., 2009.

[12] K. Jensen, L. Kristensen, and L. Wells, “Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems,” *International Journal on STTT*, vol. 9, no. 3-4, 2007.

[13] A. F. Molisch, K. Balakrishnan, C.-C. Chong, S. Emami, A. Fort, J. Karedal, J. Kunisch, H. Schantz, U. Schuster, and K. Siwiak, “IEEE 802.15.4a channel model-final report,” *IEEE P802*, vol. 15, no. 04, p. 41, 2004.

[14] V. Gungor and G. Hancke, “Industrial wireless sensor networks: Challenges, design principles, and technical approaches,” *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 4258–4265, 2009.

[15] D. Martínez, A. González, F. Blanes, R. Aquino, J. Simo, and A. Crespo, “Formal specification and design techniques for wireless sensor and actuator networks,” *Sensors*, vol. 11, no. 1, pp. 1059–1077, 2011.

[16] J. Ben-Othman, S. Diagne, L. Mokdad, and B. Yahya, “Performance evaluation of a hybrid MAC protocol for wireless sensor networks,” in *Proceedings of MSWiM*, pp. 327–334, 2010.

[17] K. I. F. Simonsen, L. M. Kristensen, and E. Kindler, “Code generation for protocols from cpn models annotated with pragmatics,” in *24th Nordic Workshop on Programming Theory (NWPT 2012)*, 2013.

[18] V. Veiset and L. M. Kristensen, “Transforming platform independent cpn models into code for the tinyos platform: A case study of the RPL protocol,” in *Proceedings of PNSE*, June 2013. (to appear).