Model-Checking Assisted Protocol Design for Ultra-Reliable Low-Latency Wireless Networks

Christian Dombrowski*, Sebastian Junges†, Joost-Pieter Katoen†, James Gross‡
*Communication and Distributed Systems, RWTH Aachen University, Germany
†Software Modeling and Verification, RWTH Aachen University, Germany
‡School of Electrical Engineering, KTH Royal Institute of Technology, Sweden

***Abstract*—Recently, the wireless networking community is getting more and more interested in novel protocol designs for safety-critical applications. These new applications come with unprecedented latency and reliability constraints which poses many open challenges. A particularly important one relates to the question how to develop such systems. Traditionally, development of wireless systems has mainly relied on simulations to identify viable architectures. However, in this case the drawbacks of simulations – in particular increasing run-times – rule out its application. Instead, in this paper we propose to use probabilistic model checking, a formal model-based verification technique, to evaluate different system variants during the design phase. Apart from allowing evaluations and therefore design iterations with much smaller periods, probabilistic model checking provides bounds on the reliability of the considered design choices. We demonstrate these salient features with respect to the novel EchoRing protocol, which is a token-based system designed for safety-critical industrial applications. Several mechanisms for dealing with a token loss are modeled and evaluated through probabilistic model checking, showing its potential as suitable evaluation tool for such novel wireless protocols. In particular, we show by probabilistic model checking that wireless token-passing systems can benefit tremendously from the considered fault-tolerant methods. The obtained performance guarantees for the different mechanisms even provide reasonable bounds for experimental results obtained from a real-world implementation.**

I. Introduction

With the increasing interest in critical machine-to-machine applications, the research community faces new challenges with respect to designing wireless networks. Traditionally, developing wireless systems has been driven mainly by voice, video, or data applications, characterized by soft latency and/or reliability requirements. However, demands of critical machine-to-machine applications are quite different as they necessitate very short latencies ($1 - 10\,\mathrm{ms}$) accompanied by guaranteed reliabilities of $1 - 1\text{E-}6$ and higher. Example applications of interest can be found in the area of industrial automation, the smart grid, and car-to-car communications. These new requirements open up many new challenges. On the one hand, it is currently much debated under which circumstances such requirements can be met, and which combination of physical layer and link layer mechanisms achieve optimal system performance [1]. These discussions are based mostly on communication-theoretic modeling and mathematical analysis to obtain principle insights.

On the other hand, a more practical challenge relates to development methodologies to be applied once theoretic guidelines are clarified. Two major problems arise in this context. As the wireless channel is a random communication medium, system design methodologies need to account for stochastic metrics. During the design phase of protocols, this has mainly been addressed so far through simulations. However, due to the extremely high reliability requirements, simulations become prohibitive with respect to their durations when evaluating systems rigorously at an appropriate level of confidence. A second issue relates to the parameterization of simulations. While simulations require the definition of a given set of parameters, a bigger interest is in providing worst-case bounds over a set of parameter ranges. In this case, non-determinism plays an important role, i. e., for certain parameters it is not possible or not useful to specify a probability of occurrence up front. With respect to simulations this leads to an even higher computational burden as a much wider parameter set has to be evaluated always with respect to the high reliability levels and statistical confidence requirements. Finally, studies [2] have shown a wide range of performance results for a given system model when evaluating with different simulation tools, questioning in general the credibility of simulation results.

In order to address these challenges, in this work we propose to apply probabilistic model checking [3], [4] during the development phase of ultra-reliable low-latency wireless networks. Given the importance of timing requirements in this case, we specifically employ Probabilistic Timed Automata (PTAs) [5], [6]. PTAs extend timed automata, the main formal modeling technique for real-time systems, with discrete probabilistic branching. We study the suitability of PTA modeling and analysis during the design phase of an ultra-reliable low-latency wireless protocol called EchoRing [7]. EchoRing is an evolved token-passing protocol featuring cooperative communication primitives to drastically increase reliability. Several fault-tolerant design alternatives exist with respect to dealing with token losses. Based on a PTA-model, we study the implications of these design alternatives on the reliability, demonstrating that probabilistic model checking serves as a viable evaluation tool for wireless system design, speeding-up the development phase in comparison to simulations. The novelty of our work relates to the application of probabilistic model checking during the design phase of wireless systems, and by validating results against a real-world implementation. Literature shows that PTAs have been applied to a-posteriori verification of various existing communication protocols [8]–[10] and other fields in natural science, but to the best of the authors' knowledge not to assist in an on-going design process. To evaluate these protocols significant pre-processing is carried out [8], [11] to reduce the complexity such that less involved formalisms like Markov chains can be utilized. This not only requires a deep protocol understanding, which typically is not given while developing a new protocol, but is also likely to abstract from critical timing aspects. However, for time-sensitive protocols exposed to a dynamic communication channel this may not be feasible. To circumvent some of the drawbacks of abstractions required in verification, hybrid approaches [12] are proposed that use channel models from simulation engines. Other work [13] incorporates other formal tools to alleviate the effects of state space explosion and abstraction errors. Further, the validity of the results in real-world environments is rarely addressed, but identified to be crucial [12].

This work is structured as follows. Sec. II briefly presents the wireless token ring protocol under study. Sec. III introduces PTAs, whereas Sec. IV gives details about the developed protocol model and the channel model. In Sec. V, we evaluate our PTA model to derive performance bounds, and investigate the accuracy of the predictions. Sec. VI concludes this work.

II. Overview of EchoRing and Problem Statement

In this section we first give a basic overview of the design of EchoRing. We then provide more details on possible fault-tolerant schemes to handle token losses.

A. Basic Protocol Functionality

The EchoRing Medium Access Control (MAC) protocol follows a token-passing approach, comparable to the Wireless Token Ring Protocol (WTRP) [14]. A token-passing MAC allows the network to be decentralized and self-configuring, two important properties that enable the network to react to wireless channel dynamics. A further important property of the wireless channel is its broadcast nature, i. e., stations in range of each other can overhear others' transmissions. This necessitates the stations to access the medium in a logical unambiguous ordering to avoid transmission collisions. The resulting logical order is a ring, in which each station has a well-defined predecessor and a successor. Such a logical ring is distinguished by an identification number (ID) provided by the first operating station.

The exclusive right to transmit in a ring is determined by holding a token. Every station holds it for a specified amount of time called Token Holding Time (THT), and forwards it at the end of the THT. Within this period, a station may transmit DATA packets including ACK and forwards the token to its successor. If no packets are to be transmitted, the station sends a NODATA packet and forwards its token. A station retransmits its token to prevent a token loss. Token retransmissions require the successor station to start its THT after the last token has been received. This avoids collisions with a potentially still ongoing token forwarding process. Hence, the first tokens are marked as INTTOKEN and only the last one is marked as TOKEN. Once the successor receives an intermediate token, it sends an ACK explicitly, except for the last one (the TOKEN) which is acknowledged implicitly by a DATA or NODATA packet. This is done to save transmission time and explains the need to send a NODATA packet if no real payload is to be sent. After a certain period called Token Rotation Time (TRT), a station receives the token again. The maximum length of this period is the Target Token Rotation Time (TTRT). TRT and TTRT depend on the actual system load and the number of stations in the ring. Exact timings are given in the following. Let $N \geq 0$ and $M \geq 0$ be the maximum number of token and payload packet retransmissions, respectively. Header Length (HL) and Payload Length (PL) represent the durations of control and payload packets, respectively. Considering the actual number of stations in the ring, n, the maximum number of supported stations, $n^\star$, and Payload Transmission Time (PTT) and Token Passing Time (TPT), these equations hold:

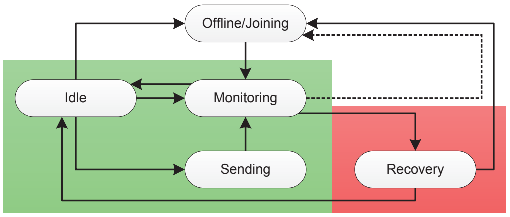


Fig. 1. Simplified Finite State Machine of the EchoRing protocol

$$\begin{aligned} \mathrm{PTT} &= (M+1)\cdot(2\cdot\mathrm{HL}+\mathrm{PL}) & \mathrm{TTRT} &= n^\star\cdot\mathrm{THT} \\ \mathrm{TPT} &= (N+1)\cdot(2\cdot\mathrm{HL}) & \mathrm{TRT} &= n\cdot\mathrm{THT} \\ \mathrm{THT} &= \mathrm{PTT}+\mathrm{TPT}-\mathrm{HL} & \mathrm{TRT} &\leq \mathrm{TTRT} \end{aligned}$$

Token passing provides three important features for real-time applications: It provides guaranteed medium access as the TRT is upper-bounded. The protocol works in a distributed fashion, eliminating the problem of a single point of failure of centralized schemes. Finally, it keeps its characteristics even under high utilization, in contrast to random access protocols.

B. Recovery Mechanisms

Token passing has two drawbacks. Firstly, exchanging the token imposes overhead that requires capacity even if a station has no payload. Yet, at least for control applications spectral efficiency is usually of minor concern as they favor reliable data exchange of fairly small packets. More problematic is a *token loss*. A token loss can cause a cascade of errors and therefore the Finite State Machine (FSM) becomes complex. Under normal conditions a station is in the IDLE, MONITORING, or SENDING mode (shown green in Fig. 1). If token forwarding to the respective successor is unsuccessful despite retransmissions, the station sets a flag to leave and rejoin the ring (at a different logical position) with potentially more favorable link conditions. This event is called *ring instability*. On leaving the ring, the station evolves to the OFFLINE mode in which it is not allowed to process packets. Typically, this jeopardizes real-time guarantees. Timers ensure that a token is not lost entirely.

In case retransmissions cannot prevent a token loss, a token creation is achieved by a timer expiry at the successor. If a station has not received a token after TTRT time units since the receipt of the previous token, it creates a new token and notifies the other stations by transmitting a CLAIMTOKEN packet. After transmitting the CLAIMTOKEN or on receiving a normal token from its predecessor, the station resets its timer for detecting a lost token.

To prevent frequent ring instabilities, EchoRing introduces an additional RECOVERY mode (highlighted red in Fig. 1). This extension consists of advanced packet inspection routines. The additional recovery strategies are aimed at enhancing the ring stability by dynamically interpreting the condition of the entire ring, based on partial information about the ring's status. The broadcast nature of the wireless channel allows to overhear other transmissions, analyze the incoming packets, and react to certain ones. This is exploited for the recovery mechanisms. Hence, a station may return to the IDLE mode instead of leaving and later rejoining the ring. Such an approach likely improves the stability since most of the time the wireless connection is degraded only temporarily due to fading. It can actually make a link appear broken, when in reality it will be stable for the next token rotation.

To successfully complete a token-forwarding process, at least one token has to reach the successor, and an ACK has to be received by the current token holder, be it explicit or implicit. As the forward and backward direction may exhibit different transmission characteristics, a token may be successfully received, while the ACK is not. Since the token holder is unaware of this, it leaves the ring according to the EchoRing protocol. Another token-forwarding error occurs if the successor does not receive at least one TOKEN. The default routine is to wait for a CLAIMTOKEN of the successor. Current conditions between both involved stations, however, may result into all packets being destroyed. The current token holder evolves to the OFFLINE in case of such a bi-directional error.

We treat RECOVERY mechanisms for two error conditions:

- **Recovery 1- Uni-directional transmission error:** Missing the ACK triggers this recovery mechanism. The station has to assume that the successor never received the packet, yielding a ring instability. If this had been the case, the successor would have sent a CLAIMTOKEN and changed the ring ID. If the ring ID however did not change, the station can conclude that its successor did receive the packet. Hence, checking for a change in the ring ID of any overheard packet constitutes the first mechanism.
- **Recovery 2 - Bi-directional transmission error:** Losing connectivity to its successor in both directions is another major transmission error. In this case, the successor station creates a new logical ring with a new ring ID. This is detected again by overhearing packets from other uninvolved stations and observing that the ring ID changed to that of the successor. Then, the previous token holding station assumes a short-term degradation and continues normal operation. After receiving the TOKEN again, the station notifies its successor of its ongoing presence.

Both mechanisms can be combined, yielding **Recovery 1+2**. Once one of these errors in the current rotation is resolved, the protocol mechanisms can repair another potential error in the next rotation. Problems arise in solving multiple errors per rotation, since the ring might easily get bisected as no station has global knowledge about the condition of the entire ring. This may result into a prolonging phase of packet collisions due to concurrently existing logical rings. To avoid faulty ring state interpretations, a station is not allowed to claim the token after it failed to forward the token in the last rotation.

III. FORMAL MODELING

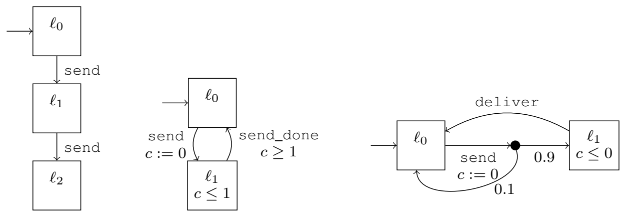
Treating token losses with dedicated recovery mechanisms yields a complex protocol state machine. This complicates checking the correctness of EchoRing, e. g., the absence of deadlocks. In addition, an important question is to what extent the recovery mechanisms improve the protocol's reliability. The features below are crucial for modeling EchoRing:

- Stochastic influence, e. g., behavior of wireless channels;
- Non-determinism, used to model packet queues and interleaving of distributed processes;
- Notion of real-time, as EchoRing is highly time-sensitive;
- Very precise reliability statements to be enforced.

Evidently, the analysis of a model to ensure that the design proposal is correct *and* can achieve the intended performance goals is a key step. As a by-product, such a model is helpful for fine-tuning and finding ambiguities in the informal protocol description. Unfortunately, the traditional model-based approaches for protocol evaluation, especially simulations, are not a good fit as the extreme reliability levels require immense simulation efforts. Analytical protocol evaluation techniques are not directly applicable either, due to the complexity of the EchoRing protocol. The various interdependencies and timing constraints make a Markov-chain based model hard to achieve. We therefore resort to more powerful models: *Probabilistic Timed Automata (PTAs)* [5], [6], an automata-based model that allows to account for probabilistic behavior, non-determinism, and real-time aspects. By analyzing them in an *automated manner* with the help of *model checking* [15], we can then check properties, e. g., if the probability of a station leaving the ring within t time units is at most 1E-5.

A. Model Checking

Model checking is a formal verification technique. Based on a model of the system at hand and a concise description of the desired properties, it automatically checks whether the model satisfies the properties. The latter are written in mathematical logic, yielding a rigorous and unambiguous description. In contrast to testing and simulation, model checking is not biased to certain scenarios, but instead relies on an exhaustive state space exploration. A model-checking result *true* means that the model satisfies the property, i. e., there is no possible behavior of the model violating it. Such hard guarantees cannot be given using simulation or testing and are the main reason why model checking is recommended by bodies such as National



(a) Station (b) Transmitter (c) Channel

Fig. 2. PTAs for the components

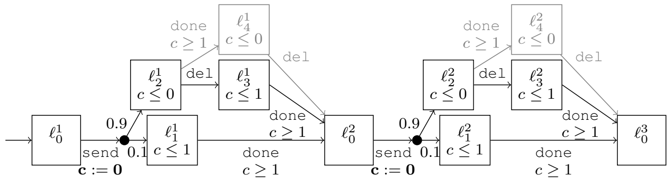


Fig. 3. PTA example of a communication process.

Aeronautics and Space Administration (NASA) and Federal Avionics Association (FAA).

As for any model-based technique, the reliability of verification results relies on the correctness of the system model, i. e., is the model adequately describing the EchoRing protocol. As models are by nature abstractions of the reality, inaccuracies are inevitable. We therefore developed the protocol models in close collaboration with the protocol engineers, used a modeling language that naturally reflects the protocol structure.

B. Probabilistic Timed Automata

PTAs extend *timed automata* [16], the prime formal modeling technique for real-time systems, by adding probabilistic transitions. Besides a transition system with transitions over locations (control states), a PTA contains a set of clocks, real-valued variables, that are used to measure the elapse of time. A PTA state, is thus, its current location together with the actual clock values. State changes happen in two ways: Either (i) time progresses, causing all clocks to increase by the amount of time passed while residing in its location, or (ii) a discrete action occurs, assumed to happen instantaneously. By executing an action, the PTA evolves to a given location and resets the specified clocks to zero, as governed by a specified distribution. Each location may be equipped with an *invariant* (upper bound on clocks). This restricts the progress of time. Time can only advance as long as the invariant is not violated. Likewise, actions can be equipped with *guards* (lower and / or upper bounds on clocks), restricting the states in which the action can be executed. Note that clocks can only be set to zero.

Example 1: Let us give a simple PTA example. We model a transmission attempt of a sender via a lossy channel. The sender sends a packet, which means it is put onto the channel. With probability 0.9, the transmission is successful and the packet is subsequently delivered; otherwise, the packet is lost (cf. Fig. 2c). There is one send action. On taking this action, the clock c is set to zero. With probability 0.1, the PTA returns to location ℓ_0. With probability 0.9, the PTA moves to ℓ_1. The deliver action can be taken at any time point to return to ℓ_0. The station attempts two transmissions (cf. Fig. 2a without time constraints). We assume that the transmitter is not usable for one time unit after sending a message. Once a message is sent, the clock is reset. Exactly one time unit is then spent in location ℓ_1 before issuing the action send_done, as indicated by the invariant $c \leq 1$ and the guard $c \geq 1$. Example states of the PTA TU are $(\ell_1, c{=}0)$, $(\ell_1, c{=}\frac{1}{2}\sqrt{2})$, and $(\ell_1, c{=}1)$, whereas $(\ell_1, c{=}3)$ is not a state as it violates the invariant of ℓ_1.

A monolithic PTA for this simple example is given by the dark part in Fig. 3. Starting left, either time passes or a first send action is taken. Now, either the transmission is successful (upper path, probability 0.9), and the message is directly delivered with no elapse of time. After that, we wait exactly one time-unit before we arrive at location ℓ_0^2. Otherwise, if the transmission is not successful, we wait a time unit before we can send another message.

A PTA is in fact a symbolic representation of a Markov Decision Process (MDP) which has an uncountable state space due to the real-valued clocks. Verifying PTAs is decidable thanks to an automated finite abstraction construction [6]. This comes with at the price of a high complexity. Checking reachability for PTAs with just two clocks is EXPTIME-complete [17]. To keep the complexity manageable, we exploit a so-called *digital clock semantics* [6] which give rise to coarser finite abstractions. This poses mild restrictions on the clocks and the PTA structure, e. g., by rounding timer values to integral numbers. For our model, these restrictions do not impose a restriction.

C. Compositional PTA Modeling

Whereas PTAs support the appropriate mechanisms—real-time and randomness—to model EchoRing, the sheer complexity of such protocols makes a monolithic approach infeasible. We therefore exploit the fact that PTAs can be *composed*, i. e., we can model a complex system or protocol by composing several simpler PTAs. These component PTAs run in parallel and synchronize on the passage of time (time evolves at the same pace in all PTAs), as well as on common actions. This keeps the model comprehensible and allows them to reflect the real system structure in a natural manner.

Example 2: The station PTA, TU, and channel synchronize on the common action send. This means that this action can only take place if all three PTAs are able to perform it. Once it is performed, all PTAs evolve synchronously to their respective location. A state of such a PTA network is given by all three PTA locations, together with the value of all clocks. Fig. 3 (including the gray part) depicts the product of these three component PTAs. The gray locations are in fact not reachable, and are removed in a preprocessing step. Note that formally we have two clocks (both named c), but as they are always synchronously reset, one has been eliminated (for the sake of readability, the result after elimination is depicted).

To enable the modeling of complex protocols such as EchoRing, we describe PTAs by the MODEST language [18]. This language supports the composition of PTAs, is conceptually close to programming languages, and yields comprehensive

```
Channel() {
  clock c;
  do {
  :: send {= c = 0 =}
     invariant(c <= 0) palt {
     :9: deliver
     :1: {==}
     }
  }
}
Transmitter() {
  clock c;
  do {
  :: send {= c = 0 =};
     invariant(c <= 1) when(c >= 1) send_done
  }
}
Station(i) {
  when(i > 0) send; Station(i-1)
}
par {
:: Channel()
:: Station(2)
:: Transmitter()
}
```

Listing 1. MoDeST description of the component PTAs of Fig. 3

and easily extensible models. The MoDeST tool [10] allows a direct translation to PTAs [19] (with variables). Using the digital clock semantics [6], such variable PTAs are mapped onto finite (but typically huge) MDPs, by encoding the values of clocks in a symbolic manner. The models are then amenable to model checking by software tools such as PRISM.

Example 3: The `Channel()` process of Listing 1 repeatedly synchronizes itself on the `send` action with a subsequent *probabilistic alternative (palt)* between the `deliver` action and doing nothing. Likewise, the `Transmitter()` process repeatedly synchronizes on the `send` action, and waits for one (ensured by *when*) and only one (ensured by *invariant*) time unit before it starts over. The `Station(i)` process makes i recursive calls. All processes start in parallel with $i := 2$.

IV. A PTA Model of EchoRing

We describe the model of a station's behavior, the recovery mechanisms, packet transmission and the channel model. A high-level view on our model is given in Fig. 4. Each station model is built on top of a module describing the transmission scheme, which decouples the modeling of the actual medium access and the control schemes. The modeling of the transmission is done on top of the (virtual) channels to other stations, and model the outcome of the transmission. Channels model the access to the medium, which is shared with other stations. The main life-cycle of a station is divided into two parts, depending on whether it currently holds the token or not. Both parts are then divided into token-related actions and others.

A. Station Model

For the station model, we distinguish the station to believe it either has the token (`hasToken`) or not. The code listings are for an arbitrary station.

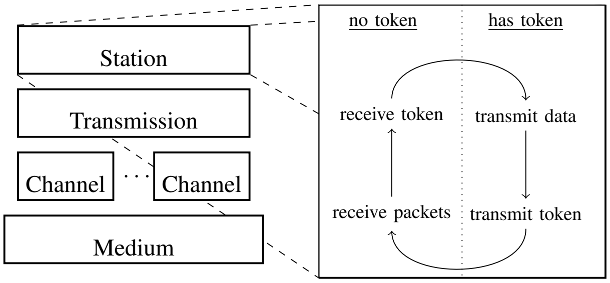


Fig. 4. A stacked model view on the modeling.

```
HasToken() {
  urgent palt { :99: SendData(STATION_C)
                :01: SendNoData() }
  invariant(stationTimer <= PTT)
  when(stationTimer == PTT) EnterTransmitToken()
}
```

Listing 2. HasToken representation

Station has Token: As described in Sec. II, on receiving a token, a station uses exactly PTT time units for data transmission. When no data is to be transmitted, a NoData packet is sent to serve as implicit acknowledgment. We assume that the first data packet is sent as soon as possible. After data transmission, the station starts the token forwarding process. Listing 2 shows the variant that data transmission is stochastic: In 1% of the cases no data is available. Variants of this scheme can be easily modeled. Note that the model requires `SendData` and `SendNoData` to return within PTT time units. `SendNoData` models the forwarding and delay of pushing the according packet to the Physical Layer (PHY). `SendData` waits for an additional known duration such that an acknowledgment (Ack) can be received. After this data transmission, we initiate the token transmission.

The counter `retries` keeps track of how many times a token has been retransmitted. IntToken is send first if `retries`$<N$, and an Ack is awaited for. It then starts listening for a given time period, measured by `auxClock`. If the station receives an Ack during this period, it sets the variable `rcvAck` to true. Once this period ends, we initialize the next token retransmission. The case `retries`$=N$ is handled similarly, except that Token is used.

Station has No Token: If the timer has not expired yet, a station waits to `Receive*` packets according to Listing 4. Dedicated processes handle packets depending on the content. On receiving the final Token, a station either normally continues or evolves to Offline. Once the IdleTimer expires (that maintains the TTRT), then depending on having had an error in the last token forwarding process or not, the station moves to either Offline or continues, possibly by claiming the Token. Claiming a token entails that a station informs all stations to switch to a ring with the id of the claimer. The station then switches to this ring assuming it has the token.

The IdleTimer deadline (`itd`) is set to TTRT − THT. During initialization, we have to set adapted deadlines to account for the fact that all clocks are initially set to zero.

```
EnterTransmitToken() { TransmitToken(0, false) }
TransmitToken(retries, bool rcvAck) {
  urgent {= stationTimer = 0 =};
  urgent alt {
  :: when(retries < N)
     urgent {= auxClock = 0 =}
     SendIntToken();
     WaitForAckIntToken();
     TransmitToken(retries+1, rcvAck)
  :: when(retries == N)
     urgent {= auxClock = 0 =}
     SendToken();
     WaitForAckToken();
     EnterIdle(rcvAck)
  }
}
WaitForAckIntToken() {
  invariant(auxClock <= 2*HL) alt {
  :: ReceiveIntAckToken()
  :: ReceiveUpdateRing(); WaitForAckIntToken()
  :: OverhearOther(); WaitForAckIntToken()
  :: when(auxClock >= 2 && (msg_type == 0 ||
    transClock != DEL_DELAY)) {==}
  }
}
```

Listing 3. TOKEN handling

```
EnterIdle(rcvAck) {
  Idle(false, rcvAck ? NO_ERR : TX_ERR)
}
Idle(rcvIntTok, errorMode) {
  invariant(stationClock <= itd) alt {
  :: MsgFromOtherRing(); Idle(rcvIntTok, errorMode)
  :: ReceiveUpdateRing(); Idle(rcvIntTok, errorMode)
  :: ReceiveToken();
     urgent alt {
     :: when(errorMode == NO_ERR) LeaveIdle(true)
     :: when(errorMode != NO_ERR) Offline()
     }
  :: ReceiveIntToken(); Idle(true, errorMode)
  :: ReceiveData(); Idle(rcvIntTok, errorMode)
  :: when(stationClock >= itd && (msg_type == 0 ||
    transClock != DEL_DELAY))
     urgent alt {
     :: when(errorMode == NO_ERR)
        LeaveIdle(!rcvIntToken)
     :: when(errorMode != NO_ERR)
        Offline()
     }
  }
}
ReceiveToken() {
  when(msg_type == TOKEN && msg_receiver == id &&
    msg_ring == ring) receive;
  ...
}
...
LeaveIdle(claimToken) {
  {= stationClock = 0 =};
  alt {
  :: when(claimToken) ClaimToken()
  :: when(!claimToken) {==}
  };
  HasToken()
}
```

Listing 4. Idle and packet reception handling in station PTA

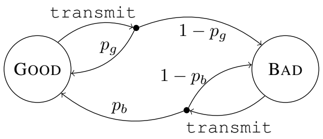


Fig. 5. The Gilbert-Elliot channel model

Recovery Mechanisms: **Recovery 1:** In all but the MsgFromOtherRing() process, the error mode is changed to NO_ERR. If now a station receives a packet on the ring it was on when entering the IDLE mode, it leaves the error mode. **Recovery 2:** In all but the MsgFromOtherRing() process, a station additionally listens on the ring whose id equals its successor. A station leaves the error mode if it receives a packet on the ring created by its successor. Process MsgFromOtherRing() ignores packets from rings other than the current ring and the ring created by its successor. **Recovery 1+2:** In addition, the error mode is left on receiving a packet on one of the two rings we are listening on.

B. Packet Transmission

In the EchoRing protocol all received packets are of interest, no matter to whom they were sent. Thus, when a Sta. A sends a packet to B, it may also arrive at Sta. C. The (virtual) channels $A \to B$ and $A \to C$ are independent, i. e., whether a packet reception succeeds at B and C is independent of each other. If a packet arrives at multiple stations, it arrives (roughly) synchronously in time. We furthermore assume that either the complete packet arrives or it does not arrive. For each packet transmission, the station involves a helper routine, which hides the low-level modeling of the transmission from the station-model. The subroutine assigns the content of the packet to medium, then starts a timer and waits until the sender can be informed that the send routine can finish. The subroutine waits a parameterized time until actually distributing the packet over the channels.

On the receiver side, we have to handle both, successful and unsuccessful receptions. Now, if the channel synchronizes via receive, the packet can be read, and the station synchronizes via arrive with all other stations. Here, it is important to prevent (i) leaking information to stations that do not receive the packet, and (ii) blocking behavior by stations that do not receive the packet. We omit details of the modeling for brevity.

C. More Realistic Channel Models

A proper link characterization is required to obtain reasonable PTA predictions. Since the protocol accesses the wireless channel on very short time scales, successive transmission attempts are not independent (as with the channel model used in Sec. III). Neglecting the correlation factor leads to an over-optimistic prediction that does not match reality. We use the Gilbert-Elliot channel model [20] to more accurately capture this effect. It is a Markov modulated process with a GOOD and a BAD state, shown to also hold for wireless links [20].

Our channel model features two actions: Packet transmission and changing (updating) the channel's state. Let p_g and p_b

```
Channel_good(pg, pb, i) {
  alt {
  :: send_via_this_channel {= c = 0 =};
     urgent palt {
     :pg:      invariant(c <= 0) receive_A;
               urgent Channel_good(pg, pb, i)
     :100-pg: urgent Channel_bad(pg, pb, i)
     }
  :: reset_Aout; palt {
    :i:      urgent Channel_good(pg, pb, i)
    :100-i: urgent Channel_bad(pg, pb, i)
    }
  }
}
```

Listing 5. Channel model inspired by Gilbert-Elliot

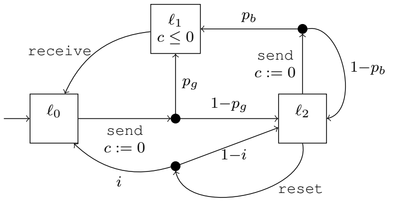


Fig. 6. PTA corresponding to our channel module

be the probability for successful packet transmission in the GOOD and BAD state, respectively (Fig. 5). To keep the channel model simple, we couple updating and transmission, i. e., in GOOD the transmission succeeds with probability p_g *and* the channel remains in the GOOD state. With probability $1-p_g$, the transmission fails and the channel switches to the BAD state. We display a PTA in Fig. 6. To account for short-time correlation, we use a `reset` action. This action can be used to reset the channel, being in the GOOD (BAD) state with probability i $(1-i)$ because after a long period, the channel can be assumed to be uncorrelated again, e. g., after a TRT. This `reset` action can be either triggered by a timer or from the outside. The GOOD channel process is displayed in Listing 5, the BAD one is analogous.

Using a timer is more general, but as the EchoRing protocol has exact timings, we can also use our knowledge about when the timer would expire, and manually reset the channels before the next transmission, e. g., before sending the ACK and possibly before the TOKEN transmissions. Employing the channel model above also means that the channel from Sta. A to Sta. B is independent of the channel in the other direction, which reflects real-world channels for which reciprocity is only realized under perfectly symmetric conditions. We emphasize that we can replace the channel model by any other PTA over actions `transmit` and `receive`. Moreover, we can employ different PTAs for different channels.

V. ANALYSIS RESULTS

Realizing a PTA model while developing EchoRing enabled deep insights into the protocol. Formal modeling revealed various inconsistencies and ambiguities. Yet, formal verification using PTAs provides more than a check for correctness. Two questions are of interest: (i) Which gains can be realized with the proposed recovery mechanisms, and (ii) how closely the predicted behavior resembles that of real systems. After describing the performance measure in Sec. V-A, Sec. V-B explains the general verification set-up and gives detailed insights into the performance gains for varying scenarios. Sec. V-C addresses the topic of how verification results match reality by comparing them to real-world measurements. This enables statements about the validity of applying PTAs to judge on the performance of real-world systems.

TABLE I
DEFINITION OF EVALUATION SCENARIOS ($M = 1, N = 1, Z = 1$)

	$p_g, 1 \to 2$	$p_b, 1 \to 2$	$p_g, 2 \to 1$	$p_b, 2 \to 1$
①	0.99	0.25	0.99	0.25
②	0.99	0.25	0	0
③	0	0	0	0

A. Evaluation Metric

The focus of our formal verification is on *ring stability*. This motivates to consider the probability of a station moving into the OFFLINE mode (cf. Fig. 1). Evidently, this probability increases with the time the station is operational in the ring.

This stems from the fact that the likelihood of an erroneous token forwarding increases when the number Z of token rotations increases, and the fact that in our model the OFFLINE mode cannot be left. Ring instability of Sta. A is thus

$$ri_A(Z) = \Pr\{\text{Sta. } A \text{ in OFFLINE within } Z \text{ token rotations}\}.$$

To check the correctness of our PTA model, we checked a.o. the probability (a) to reach a deadlock, (b) of two stations assuming they hold the token and (c) of a station sending anything beside ACKs when it has no token. For these checks, only NODATA packets are sent.

B. Formal Verification Results

Verification set-up: The model checker is executed on one CPU core of a MacBook Pro with a 2.7 GHz Intel Core i5 processor and 16 GB RAM. In the verified PTA model, the station of interest increments a counter `k` on every completed rotation. The logical formula

```
Pmax=? [ (k <= Z) U ("station_1=OFFLINE")]
```

inquires the maximum probability over all possible non-deterministic behaviors (e. g., interleavings of distributed processes) that a particular station reaches the OFFLINE mode within Z rotations.

The initial state of the model is set to mimic the behavior that the station of interest just joined the ring. Formal verifications of the PTA model focus on ring configurations with $n = 3$ through $n = 5$ stations. Here, we consider some simplified channel configurations. For all rings, we assume that channels which do not connect to Sta. 1 are perfect, i. e., packets arrive deterministically. The same holds for all outgoing channels from Sta. 1 to Sta. j, with $2 < j \leq n$, i. e., to stations which are not successor of Sta. 1. The corresponding incoming channels are parameterized with $(p_g, p_b) = (0.99, 0.25)$.

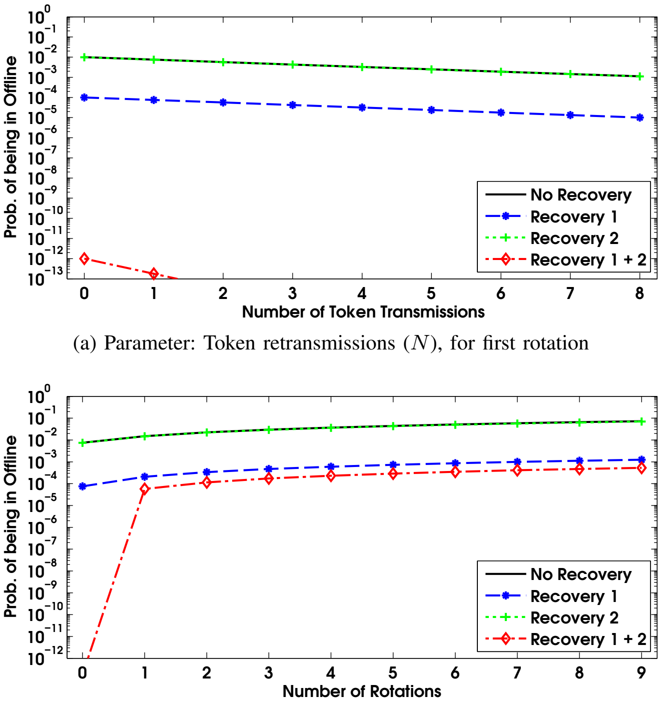
TABLE II
PTA-BASED ANALYSIS OF RING INSTABILITY FOR VARIOUS RING SIZES

①	No Recovery	Recovery 1	Recovery 2	Recovery 1&2
$n=3$	7.53E-3	7.50E-5	7.45E-3	3.16E-9
$n=4$	7.53E-3	7.50E-5	7.45E-3	1.78E-13
$n=5$	7.53E-3	7.50E-5	7.45E-3	1.78E-15
②	No Recovery	Recovery 1	Recovery 2	Recovery 1&2
$n=3$	1	7.50E-3	9.93E-1	5.63E-5
$n=4$	1	7.50E-3	9.93E-1	3.16E-9
$n=5$	1	7.50E-3	9.93E-1	3.16E-11
③	No Recovery	Recovery 1	Recovery 2	Recovery 1&2
$n=3$	1	1	5.62E-5	5.63E-5
$n=4$	1	1	3.16E-9	3.16E-9
$n=5$	1	1	3.16E-11	3.16E-11

First, we consider three scenarios particularly well-suited to give some basic insights into the performance of the recovery mechanisms. We construct the scenarios such that they are simple enough to understand manually and expose the situations we want to recover from. The first scenario is the baseline, the second scenario suffers from deterministic one-directional errors, the third from deterministic bi-directional errors. The scenarios are given in Table I, the first two columns describe parameters for the channel from Sta. 1 to Sta. 2, the second set of rows describes the parameters for the reverse direction. It is assumed that $i{=}1$ for all channels.

Model checking results: We analyze the probability of an EchoRing station to evolve into the OFFLINE mode within a single rotation ($Z{=}1$) under varying channel conditions. We consider four cases: no additional recovery mechanism, the recovery mechanisms for uni-directional and bi-directional transmission (separately), and the setting with both recovery mechanisms. Results are listed in Table II for $M{=}0, N{=}1$. State space sizes of the PTA model vary from 558 states for $n{=}3$ without recovery in Scenario ③, to about 580,000 states for $n{=}5$, with both recovery mechanisms in Scenario ①. Generating the underlying MDP from MoDeST for each model takes around 3 s. These MDPs contain between 1,000 and 80,000 transition-sets. The vast majority of the model-checking time is spent on the explicit state space construction ($15-1000$ s). Verifying the model is almost instantaneous, even for large models ($<1\%$). One eminent lesson is that the uni-directional error handling has a much larger influence than the bi-directional error handling, as the probability of observing a bi-directional error case is less likely. Moreover, we see that the potential of increased traffic in larger rings significantly improves the ring stability when both recovery mechanisms are used. If only one mechanism is in place, the unhandled failure case overshadows the effect of the other mechanism, and therefore, additional traffic is not significantly increasing the ring stability. Only in Scenario ③, there is no uni-directional error case, which implies that no unhandled failure case exists.

Verification results also allow to analyze some protocol trends; two examples are given in Fig. 7 (for $n{=}4$ and Scenario ①). Fig. 7a plots the probability of being offline versus the number of token transmissions (for one rotation). It shows that in this scenario the benefit of additional token transmissions is limited. This is mainly due to the correlation of multiple short-timed transmissions. Especially the recovery mechanisms additionally benefit from the increased amount of traffic, which helps the recovery. This becomes evident when both recovery strategies are used, as the majority of possible error schemes are then covered. Fig. 7b shows that for without combined recovery, the relative effects of the recovery can indeed be assessed using only one iteration. For the combined recovery mechanism, we observe a much lower failure probability within the first iteration. This can be explained as follows. At initialization all stations start in the same ring and the join of Sta. 1 is successful. Thus, Sta. 1 can safely claim a TOKEN in the first rotation, if it could recover from a transition failure at the start of the rotation. In later rotations, this is not always the case. Such scenarios rarely occur (roughly with probability 1E-6) which makes this initialization artifact hardly visible except when recovery strategies are combined (as then these kind of failures are the major source of failure).



(a) Parameter: Token retransmissions (N), for first rotation

(b) Parameter: Number of completed rotations Z, ($N{=}1$)

Fig. 7. Changing parameters in Scenario ①, $n{=}4$

C. Validating Verification Results by Measurements

Now we evaluate the accuracy of the verification model. Knowing whether verification results are accurate is crucial to derive valid bounds for the envisioned industrial applications.

Experimentation Set-up: The experimental study was performed by implementing EchoRing on the Wireless Open-Access Research Platform (WARP) [21]. An OFDM system that resembles IEEE 802.11A is used as the PHY layer. The

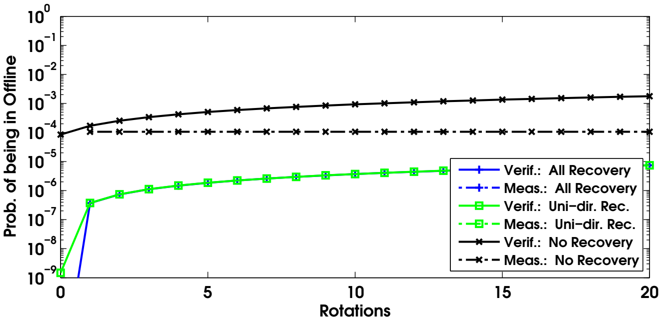


Fig. 8. Measurement with infrequent transmission errors ($N = 1$)

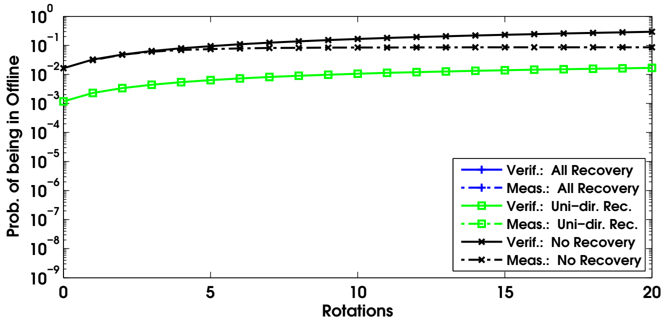


Fig. 9. Measurement with frequent transmission errors ($N = 1$)

network consists of $n = 3$ stations. Setting $N = 1$ allows for an additional retransmission of the Token. Payload packets were generated periodically and enqueued in measurement runs, each lasting for $10\,\mathrm{min}$. The environment in which the measurements were carried out is an office environment with distances of $3 - 15\,\mathrm{m}$ between stations. In order to see a reasonable number of error situations, we chose transmit powers that resulted in empirical error rates fluctuating between 1E-5 and 1E-1. Evaluating the accuracy of PTA predictions is based on parameters derived a-posteriori from these measurements. Stations record certain events depending on which station is predecessor and which is successor. A station further counted the inter-arrival times of erroneous situations, i. e., how many rotations it takes between joining the ring and having to leave again. Within each measurement run, all recovery mechanisms were enabled, but their application was tracked separately. This led to no transitions to Offline overall. However, to obtain information about the effectiveness of the single recovery mechanisms, certain events that were correctly resolved have been counted as a transition to Offline, e. g., to make statements about the first recovery mechanism, we counted the application of the second mechanism as an error event.

Channel Model Parameters: To properly parameterize the previously introduced channel model, the measurement results are analyzed afterwards. Stations record subsequently transmitted pairs of packets, e. g., an IntToken and a Token. By counting how often certain combinations of transmission errors of these packets occur allows to describe the temporal correlation. For this, we apply the Gilbert-Elliot model to the recorded events to obtain p_g and p_b empirically.

Experimental Results: In the following, the probability of a ring instability is plotted against an increasing number of rotations for the four protocol variants introduced in Sec. II-B. We show experimentally measured error events and the corresponding PTA predictions for up to $Z = 20$ rotations.

Discussion of Results: We start by examining a measurement scenario in Fig. 8 for a rather stable connection with an averaged empirical packet error rate of 6.19E-5 between the station of interest and its successor. In the observed measurement trace, only the first recovery mechanism found application. Furthermore, no event occurred for which neither recovery mechanism could resolve the transmission error, and hence, the green and blue dashed curves are missing. The model checker prediction is in accordance with this observation, i. e., the model checker predicts probabilities starting in the range of 1E-12. The obtained measurement trace is not long enough to make a statement about this level of precision. Hence, the dashed blue and green curves are missing in Fig. 8. However, in case of no additional recovery mechanisms, erroneous situations sporadically occur (dashed black curve). Model checking generates a curve close to that of the measurement (solid black curve). Given that a notable amount of observed packet errors came in bursts (6 incidents happened within 3 rotations, 4 incidents distributed across 100 to 100,000 rotations), the prediction is close to the actual measurement trace for small Z, but later on increases steadily. The channel model as well as its parameterization result into a rather pessimistic prediction as an error is expected for every rotation with the same likeliness. The actual trace however contained only several bursts, hinting towards the need for better long-term correlation models.

In addition to very good channel conditions, we present in Fig. 9 a situation with a significant packet error rate of 5.1%. This leads to a substantial number of situations in which a station has to move to Offline mode and rejoin the ring later on. In contrast to Fig. 8, the experimentally and theoretically derived curves show in general a worse ring stability, but are also closer to each other as the situation is of long-lasting nature and cannot be attributed towards an error burst. Due to the channel parameterization, verification presumes a lot more bi-directional transmission errors (which cause the predicted instability reflected by the green solid curve). Yet, the actual measurement trace does not contain a single event of applying the second recovery mechanism. To improve the accuracy, a more precise channel model needs to be added to the system that captures correlations between forward and backward direction. Drawing a conclusion based on these observations leads to an underestimation of the actual protocol performance. Since the link from the predecessor to the station of interest is perfect in the obtained measurement trace, the proposed recovery mechanisms are able to cope with all token forwarding errors. This allows the PTA to predict no instabilities at all, which was also observed in the trace.

Summarizing these observations, it can be seen that *verification results indeed accurately estimate the protocol behavior*. However, as is the case with every model-driven tool, important aspects of the environment need to be captured correctly.

TABLE III
MODEL CHECKING PRECISION VS. COMPUTATION TIME.

error margin	1E-3	1E-6	1E-9	1E-12
result	**0.001**401	**0.001401**215	**0.001401215**741	**0.001401215741**
time (sec)	11.1 ± 0.2	11.4 ± 0.2	11.7 ± 0.2	11.8 ± 0.2

D. Precision of the Results

For the high reliability that EchoRing and similar protocols aim for, any method employed to assess its reliability needs a high precision. Assessing reliability with a high precision is known to be intrinsically hard [22], [23] as a reasonable number of events of interest has to be captured. For simulations, a rule of thumb is to generate at least two orders of magnitude more events than the targeted statement of interest, e. g., generating at least 1E14 packets allows to make statements on an error floor of 1E-12 with reasonable confidence.

We briefly compare how model checking scales with respect to the required precision. Table III gives the time required to analyze the model with frequent transmission errors for eight rotations, for four different levels of precision. The emphasized results are forced to be correct by the precision level, we display the result up to the correct decimal. While a higher precision requires some more analysis time, it shows that for the current models the construction of the state space has the highest computational effort, and that therefore obtaining a higher precision in the analysis phase comes at no significant costs. In contrast, performing experiments using simulations or real-world measurement campaigns takes significantly longer. Considering a real-world campaign in which packets are generated every 5 ms, it would take roughly 42 d to reach an error rate of 1E-9, but 4,200 d to make statements with reasonable confidence. Even if simulations can speed up this execution time, the general rule of thumb still holds; the higher the reliability guarantees are, the longer the run-times will be.

VI. CONCLUSIONS

This paper presented the formal modeling and the resulting verification of the EchoRing protocol, a promising solution for hard real-time communication for low-latency wireless industrial networks. We presented a compositional modeling of the protocol that naturally reflects the protocol's functioning. The risk of making unnecessary abstraction errors is thus kept to a minimum. The formal modeling has revealed several ambiguities in the informal protocol description. Its compositional nature enables easy modifications and experimentation with several stability enhancing mechanisms. Together with formal verification, it has provided deep insights into the protocol behavior. One of the benefits of our approach is that it allows determining correctness as well as studying the performance under varying conditions. In addition, the protocol development process can be guided and streamlined by ruling out inferior alternatives in early design phases without the need of long-lasting measurement campaigns. It is shown how certain choices and procedures affect the protocol stability. We firmly believe that PTAs are the right means to model and analyze highly time-dependent protocols that are subject to random effects such as ultra-reliable low-latency wireless protocols.

ACKNOWLEDGMENT

This research is funded by the BMBF-IKT 2020 Project 16KIS0138 HODRIAN and the CDZ project CAP (GZ 1023).

REFERENCES

[1] P. Suriyachai, U. Roedig, and A. Scott, "A Survey of MAC Protocols for Mission-Critical Applications in Wireless Sensor Networks," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 2, pp. 240–264, 2012.

[2] D. Cavin, Y. Sasson, and A. Schiper, "On the Accuracy of MANET Simulators," in *Proc. of the 2nd ACM Int'l Workshop on Principles of Mobile Computing*, 2002, pp. 38–43.

[3] M. Kwiatkowska, "Model Checking for Probability And Time: From Theory to Practice," in *IEEE Symp. on Logic in Comp. Science (LICS)*, 2003, pp. 351–360.

[4] J.-P. Katoen, "Model Checking Meets Probability: A Gentle Introduction," in *Engineering Dependable Software Systems*, ser. NATO Science for Peace and Security Series, D. IOS Press, 2013, vol. 34, pp. 177–205.

[5] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston, "Automatic Verification of Real-Time Systems with Discrete Probability Distributions," *Theor. Comput. Science*, vol. 282, pp. 101–150, 2002.

[6] G. Norman, D. Parker, and J. Sproston, "Model Checking for Probabilistic Timed Automata," *Formal Methods in System Design*, vol. 43, no. 2, pp. 164–190, 2013.

[7] C. Dombrowski and J. Gross, "EchoRing: A Low-Latency, Reliable Token-Passing MAC Protocol for Wireless Industrial Networks," in *European Wireless Conference (EW 2015)*, Budapest, Hungary, 2015.

[8] C. Daws, M. Kwiatkowska, and G. Norman, "Automatic Verification of the IEEE 1394 Root Contention Protocol with KRONOS and PRISM," *STTT*, vol. 5, no. 2-3, pp. 221–236, 2004.

[9] W. Fokkink and J. Pang, "Simplifying Itai-Rodeh Leader Election for Anonymous Rings," *Electron. Notes Theor. Comput. Sci.*, vol. 128, no. 6, pp. 53–68, 2005.

[10] E. M. Hahn, A. Hartmanns, H. Hermanns, and J.-P. Katoen, "A Compositional Modelling and Analysis Framework for Stochastic Hybrid Systems," *Formal Methods in System Design*, vol. 43, no. 2, pp. 191–232, 2013.

[11] A. Remke and X. Wu, "WirelessHART Modeling and Performance Evaluation," in *IEEE/IFIP Int'l Conf. on Dependable Systems and Networks (DSN)*, Budapest, Hungary, June 2013, pp. 1–12.

[12] M. Fruth, "Formal Methods for the Analysis of Wireless Network Protocols," Ph.D. dissertation, Oxford University, 2011.

[13] A. Mouradian and I. A. Blum, "Formal Verification of Real-Time Wireless Sensor Networks Protocols: Scaling Up," in *26th Euromicro Conf. on Real-Time Systems*, 2014, pp. 41–50.

[14] M. Ergen, D. Lee, R. Sengupta, and P. Varaiya, "WTRP - Wireless Token Ring Protocol," *IEEE Trans. on Vehicular Tech.*, vol. 53, no. 6, pp. 1863–1881, 2004.

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

[16] R. Alur and D. L. Dill, "A Theory of Timed Automata," *Theor. Comput. Science*, vol. 126, no. 2, pp. 183–235, 1994.

[17] M. Jurdziński, F. Laroussinie, and J. Sproston, "Model Checking Probabilistic Timed Automata with One or Two Clocks," in *Proc. of TACAS*. Springer, 2007, pp. 170–184.

[18] H. C. Bohnenkamp, P. R. D'Argenio, H. Hermanns, and J.-P. Katoen, "Modest: A compositional modeling formalism for hard and softly timed systems," *IEEE Trans. on Software Eng.*, vol. 32, no. 10, pp. 812–830.

[19] A. Hartmanns and H. Hermanns, "A Modest Approach to Checking Probabilistic Timed Automata," in *6th Int'l Conf. on Quantitative Evaluation of Systems (QEST)*, 2009, pp. 187–196.

[20] F. Martelli, M. Elena Renda, G. Resta, and P. Santi, "A Measurement-Based Study of Beaconing Performance in IEEE 802.11p Vehicular Networks," in *IEEE INFOCOM*, 2012, pp. 1503–1511.

[21] "WARP Project." [Online]. Available: http://www.warpproject.org

[22] K. Pawlikowski, H.-D. Jeong, and J.-S. Lee, "On Credibility of Simulation Studies of Telecommunication Networks," *IEEE Communications Magazine*, vol. 40, no. 1, pp. 132–139, 2002.

[23] A. Lagnoux, "Rare event simulation," *Probability in the Engineering and Informational Sciences*, vol. 20, no. 01, pp. 45–66, 2006.