Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 19 (2013) 273 – 280

Procedia Computer Science

The 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013)

Symmetry Reduction of Time-Triggered Ethernet Protocol

Marwan Ammar[a], Samir Ouchani[a,*], Otmane Aït Mohamed[a]

[a]*Hardware Verification Group (HVG), Concordia University, Montreal, Quebec, Canada H3G 1M8*

Abstract

Time-Triggered Ethernet is a relatively new protocol for safety-critical Local Area Network environments, known for its reliability and efficiency in providing seamless fail-safe communication. This paper presents a formal verification of the TTEthernet protocol, based on an interpretation of the TTEthernet specification document, applied to two formal network scenarios that represent a real-time, safety-critical setting. The conceptual model of both scenarios are modeled using the probabilistic timed automata, then, encoded into PRISM code. Also, a set of precise functional requirements is derived from the TTEthernet specification document and expressed using PCTL. Using this approach, the TTEthernet protocol was verified as applied to real-life network environments and one faulty state was identified within one of the settings, thus revealing a possible weak point in TTEthernet protocol.

Keywords: Probabilistic Verification, Probabilistic Timed Automata, PCTL, PRISM; Time-Triggered Ethernet Protocol.

1. Introduction

Time-Triggered Ethernet (TTE) [1, 2] is a Local Area Network that builds on top of Ethernet protocol, adding to it a deterministic time and event layer, based on the functional principles of the Time-Triggered Protocol (TTP). These two protocols arent to be confused for they are not the same thing, as TTE is applied to high-level computer systems while TTP is used with networks of simpler electronic components. The infrastructure of TTE is designed for mixed criticality systems, which means the protocol can handle seamless dataflow in applications bearing different criticality levels, such as aerospace, health-care or data-backup networks. For the purpose of this study, the safety-critical aspects of TTE will be considered. The TTE bases its functionality by assigning a time frame for each node, in which that node will be allowed to send data. This functionality is known as Time-Triggered communication and it is implemented by synchronizing the clocks of all active elements that are part of the network.

Despite of its high level of sophistication and the use of formal methods in the development of some of its control algorithms, TTE protocol was never verified as a whole real-world network scenario, where all

*Corresponding author
Email addresses: m_amma@ece.concordia.ca (Marwan Ammar), s_oucha@encs.concordia.ca (Samir Ouchani), ait@ece.concordia.ca (Otmane Aït Mohamed)

doi:10.1016/j.procs.2013.06.039

components are subject to fail in the most diverse ways, and, as stated in the specification document [1], the protocol may contain faults. Thus, implementation on safety-critical environments, as of today, relies heavily on simulated test runs and network redundancy, to ensure fault-free functionality at all times.

Our objective is to model a state-driven closed-world scenario encompassing all the elements of a network, making use of the TTE standard. Then, verify the TTE-based network formally by using the PRISM model checker[3]. The use of PRISM was motivated by its statistical capabilities; it is capable of running comparative analysis on sets of result values. In order to avoid verification process limitations, each scenario is verified with an adapted verification technique. Both network designs are largely based on the TTE functional descriptions available in [2, 4, 5] and [6] is especially valuable for modeling the clock synchronization function. To our knowledge, this is the first time TTE protocol has been verified as a complete network scenario at a high-level of abstraction, where the focus is the analysis of the network behaviour as a system and the verification of the relations between the networks multiple components, in detriment of a low level study of how each component performs its function. This paper is organized as follows. Section 2 presents the verification approach. The verification results are given in section 3. Section 4 discusses the related work and concludes this paper.

2. The Verification Approach

This section brings an approach for the functional requirements and the performance analysis of TTE protocol designed as Probabilistic Timed Automata (PTA). The proposed approach is illustrated in Figure 1; it consists in modeling each scenario to a corresponding PTA. The Combined PTA of the resulting model is encoded into PRISM input language and the functional requirements are expressed in Probabilistic Computational Tree Logic (PCTL) in order to evaluate the functionality requirements and the probabilities for best/worst cases.

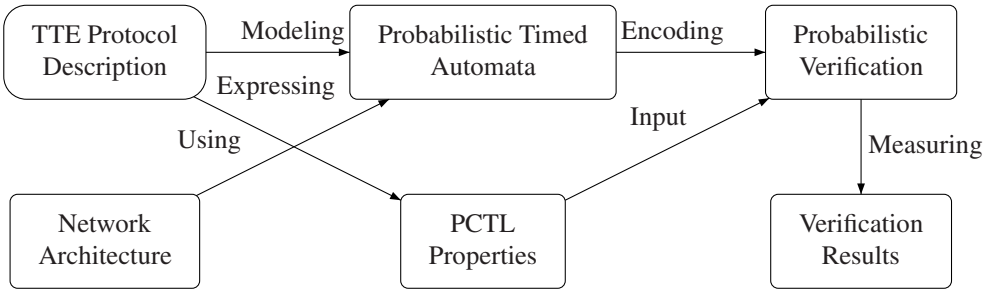


Fig. 1: Time-Triggered Ethernet Verification Approach.

2.1. TTE Model Description

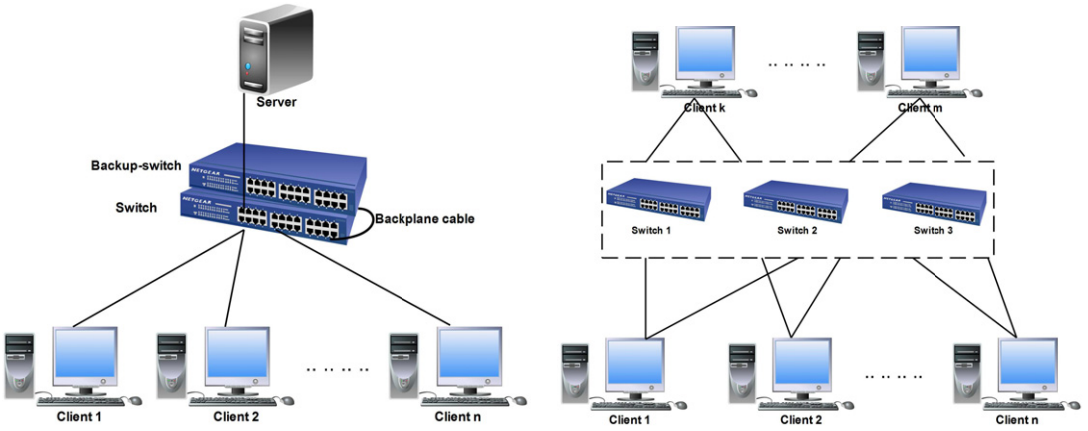
Here, we propose two scenarios describing the possible TTE architecture.

Scenario 1. The first scenario describes a standard TTE configuration, which consists of a number N of terminals connected through a TTE switch. Standard TTE provides deterministic time-triggered communication between nodes without safety-critical protection. As depicted in Figure 2(a), Scenario 1 is composed by a number of computer nodes connected to a server through a TTE switch. A computer node is formed by a host computer system and a standard TTE communication controller. A standard TTE switch is composed by a TTE switch and a standard TTE controller. For the sake of this scenario, a few constraints were incorporated to the network:

- The server is just a data repository and has no effect on the network,

- Only clients are able to send data and the only destination is the server, with the exception of the server, hardware failures can be introduced to any element of the network,
- Only one hardware failure may happen at each run,
- Theres a backup shadow switch that will become active,
- All cases of client failure are treated as non-silent failure, where the client will start to continuously flood the network with garbage data.

Scenario 2. The second scenario describes a Fault-Tolerant TTE Configuration, which consists of a number N of terminals connected through two or more Fault-Tolerant TTE switches. This setting is used for safety-critical applications, such as life support equipment. As depicted in Figure 2(b), Scenario 2 is composed by a number of computer nodes connected through two Fault-Tolerant TTE switches, in a star topology. A computer node is formed by a host computer system and a Fault-Tolerant TTE communication controller. A Fault-Tolerant TTE switch is composed by a TTE switch and a Fault-Tolerant TTE controller.



(a) Scenario 1. (b) Scenario 2.

Fig. 2: TTE Topologies.

The switches in Fault-Tolerant and in Standard TTE are very similar; the main difference being the TTE Guardian Function present in the Fault-Tolerant TTE switches. In this work, the guardian function implements a Leaky Bucket flow control algorithm, as stated previously. The same constraints used in scenario 1 are valid for scenario 2, with only a few differences: Theres no server in this scenario, Clients will send data to one-another, and any two hardware failures can occur at each run.

2.2. TTE Formal Model

From TTE description, we consider TTE as a set of objects communicating via ordered messages. We propose to design our scenarios around basic constructions of clients and switches (Figure 4). These models show the main functionality of each component in the network. These components are combined to form scenarios 1 and 2. Formally, we define TTE protocol in Definition 1.

Definition 1 (TTE Protocol). A TTE Protocol is a tuple $TTE = (\mathbf{Obj}, \mathbf{Int}, \mathbf{Order})$, where:

- **Obj** is the set of objects that define the actors such as server S, clients C and switches W,
- **Int** $= m_1, \ldots, m_n$ is a sequence of messages of size n. Each message is a tuple $m_i = (\mathbf{S_i}, \mathbf{N_i}, \mathbf{G_i}, \mathbf{T_i})$ where $\mathbf{S_i}$ is the source, $\mathbf{N_i}$ is the message content, $\mathbf{G_i}$ is a guard and $\mathbf{T_i}$ is the message target,
- **Order** : **Int** $\rightarrow \mathbb{N} \times \mathbb{N}$, is a function assigning to each message its order in the source and the target objects.

To verify scenarios 1 and 2, we use PCTL to express their related specifications. Formally, PCTL syntax is given as follows:

$\phi \ ::= \ \top \mid ap \mid \phi \wedge \phi \mid \neg\phi \mid \mathrm{P}_{\bowtie p}[\psi]$ and $\psi \ ::= \mathrm{X}\phi \mid \phi \mathrm{U}^{\leq k}\phi \mid \phi \mathrm{U}\phi$.

Where "ap" is an atomic proposition, $k \in \mathbb{N}$, $p \in [0,1]$, and $\bowtie \in \{<, \leq, >, \geq\}$. "$\wedge$" represents the *conjunction* operator and "$\neg$" is the *negation* operator. "X", "$\mathrm{U}^{\leq k}$" and "U are the *next*, the *bounded until* and the *until* temporal logic operators, respectively.

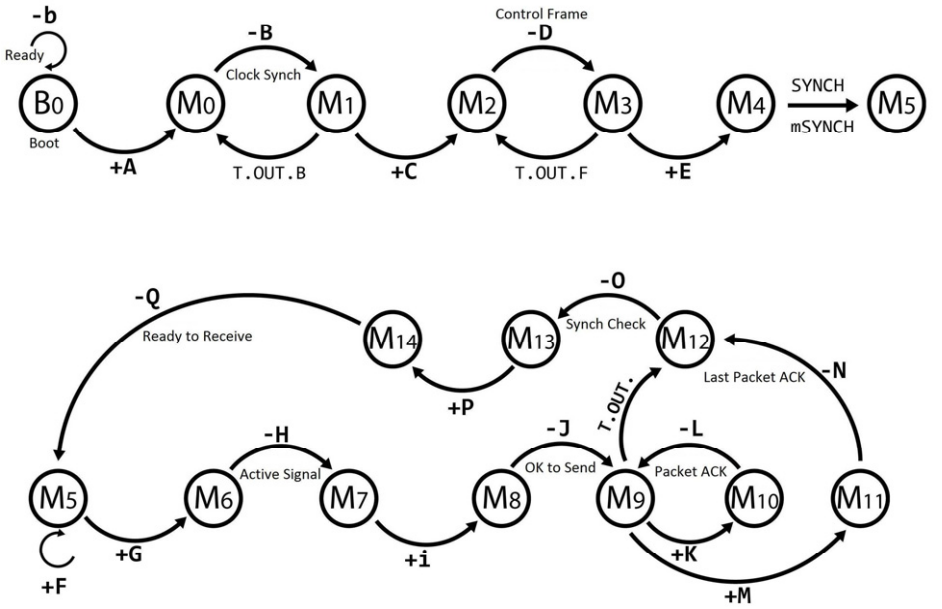


Fig. 3: Probabilistic Timed Automata of TTE Switch.

Scenario 1. This scenario uses a simple Master/Slave one-step synchronization protocol where the single master will calculate the synchronization clock, based on its internal clock, and broadcast it to the rest of the network, thus we ignore the TTE compression function.

Definition 2 (Scenario 1). The first scenario is a special instantiation of TTE-uple such as:

- **Obj** is a collection of servers, switches, and clients,
- **Int** is a set of messages,
- **Order** : **Int** $\rightarrow \mathbb{N} \times \mathbb{N}$ is an order function.

For *Scenario 1*, we prove in Proposition 1 that if a PCTL property ϕ_i proper to a node C_i is satisfied then its similar one for node j is satisfied too.

Proposition 1 (Scenario 1). *For a given node C_i that communicates with a server S via a switch W as in the form of Scenario*1*, we have:*

1. $S\|W\|C_i \models \phi_i \Rightarrow \forall\, j > 0 : S\|W\|C_i\|C_j \models \phi_i$ *where ϕ_i propositions exclude N_j variables.*
2. $S\|W\|C_i \models \phi_i \Rightarrow \forall\, j > 0 : S\|W\|C_j \models \phi_j$ *where ϕ_j is the symmetric PCTL expression of ϕ_i with to C_j variables.*

PROOF. Before discussing the proof of each case, note that the composition operator "$\|$" is commutative and associative. Those two properties help to generalize both case of this proposition to cover n nodes.

1. The first proof is based on the cone of influence for the node C_j. This is done for each term of PCTL syntax.
2. The second proof is based on the structure of C_i and C_j. By definition, we consider that $C_i \equiv C_j$. The satisfaction of PCTL properties follows a structural induction on PCTL syntax.

Similarly, it is true for more than one Switch $W_{i:i>0}$ or Server $S_{j:j>0}$.

Scenario 2. This scenario simulates a safety-critical network with double-failure hypothesis and a multi-master two-step clock synchronization, where the existence of more than one synchronization master requires an extra step for generating the synchronization clock. The multiple synchronization masters will send a control frame each to a compression master that will calculate an average value based on the latencies of those control frames as they arrive. Then, the compression master sends a new control frame that will be used to synchronize the local clocks in the synchronization masters, as described in [4]. This scenario is defined in Definition 3.

Definition 3 (Scenario 2). The second scenario is a TT implementation where:

- **Obj** is a collection of switches and clients,
- **Int** is a set of messages,
- **Order** : **Int** $\rightarrow \mathbb{N} \times \mathbb{N}$ is a function that assign for each message its order.

The configuration consists of five nodes (clients) and two switches/ guardians on a star network configuration. The guardian uses a Leaky Bucket function. Every connection is redundant. There is a backup shadow-switch in case of catastrophic hardware failure of one of the original switches. In this scenario any two faults can happen at any given time with the only exception being that only one switch may fail per run since the scenario has only one backup and the safety-critical protocol requires at least two functioning switches.

For *Scenario 2*, we derive the symmetry property of the communicated nodes as given in Proposition 2.

Proposition 2 (Scenario 2). *Let C_i and C_j two nodes communicate as in* Scenario 2*, we have:*

1. $C_i \parallel W_k \parallel C_j \models \phi_i \Leftrightarrow C_i \parallel W_k \parallel C_j \models \phi_j$ *where ϕ_j is symmetric to ϕ_i with respect to the propositions of C_i and C_j.*
2. $G_k \parallel C_j \models \phi_i \Leftrightarrow G_k \parallel C_j \models \phi_j$ *where ϕ_j expression is equivalent to that of ϕ_i but ϕ_j propositions are proper to C_j.*

PROOF. The proof is similar to the previous proposition.

1. The first is based on the structural equivalence of both properties ϕ_i and ϕ_j. By assuming that ϕ_i is symmetric to ϕ_j for the propositions in C_j. And, we have by definition $C_i \equiv C_j$ and $\|$ composition operator preserves the satisfaction of PCTL properties under adversaries. We found that the first part of proposition holds.
2. The second part is deduced by applying the cone of influence on the first one.

For the generalization case, we have the commutativity and the associativity of the composition operator "$\|$". Then, Proposition 2 holds for more than one node or one switch.

2.3. TTE PRISM Representation

The network scenarios are coded in PRISM as interdependent modules. The diagrams were coded to handle TTE properties, such as Predictable and Deterministic message transfer, strong fault isolation, scalability, seamless communication and no single point of failure. Due to the space limitations, we present only the Channel PRISM code and we describe the switch and the client codes.

The Channel Process: The network models have one instance of "Transmission Channel" for every connection between a client node and a switch. The channel has two state variables, *ch* and *failure*. The first is used to keep track of the transmission state of the channel and the latter is used to implement a physical failure logic. As long as the channel is fault-free (*failure* = 0) it is susceptible to a 0.05 percent chance of having a physical failure. Listing 1 shows a simplified code for the channel module.

Listing 1: Channel PRISM Code Fragment.

```
Module Channel_A

ch1:[0..3] init 0;
fail1: [0..1] init 0;
[Msnd] (ch1=0)&(fail1=0) -> (ch1'=1);
[rcvC] (ch1=1)&(fail1=0) -> (ch1'=2);
[Csnd] (ch1=2)&(fail1=0) -> (ch1'=3);
[rcv] (ch1=3)&(fail1=0) -> (ch1'=0);

[Csnd] (ch1=0)&(fail1=1) -> (ch1'=1);
[Csnd] (ch1=1)&(fail1=1) -> (ch1'=2);
[Csnd] (ch1=2)&(fail1=1) -> (ch1'=3);
[Csnd] (ch1=3)&(fail1=1) -> (ch1'=0);

[](fail1=0) -> 0.05:(fail1'=0)+0.95:(fail1'=1);
endmodule
```

The Switch Process: In the switch module, every transmission is guarded by the state of the timer module and the transmission must be synchronized with the state of Channel module. This structure ensures Deterministic Communication, where no two messages can be sent at the same time. This measure is part of the Guardian Function of TTE. In PRISM, fault tolerance was modeled by restraining every "send" and "receive" action to a rendezvous connection between Client, Channel and Switch modules.

The Client Process: The Client module is constructed around the principles and boundaries of the Channel module. Every client has a unique ID number which the network will use to identify which client is currently active. This allows every message to be traced back to its origin, ensuring that the network meets the TTE property "Consistent Diagnosis".

3. TTE Verification

To guarantee that the proposed scenarios fall under the TTE specifications, a set of eight properties is proposed that, if held true, will ensure the models conformance with TTE. The actual verification is performed by placing a set of state-triggered Boolean flags into the PRISM code. Whenever a relevant state is reached, a specific flag will be turned to mark that event (eg. Entering or exiting transmission mode). To circumvent verification limits, verification is performed over two symmetrical nodes in the network. Then the results are formally generalized in order to cover N nodes, by applying the proved propositions. The properties are expressed in PCTL for a generic node "A" and expressed as a set of assertions that should always hold true, about the aforementioned flags.

1. The transmission only starts after successful synchronization of the node;
   Pmax=? [G !(!"Synch_A" & "Tr_A")]
   Where "Synch" represents that the node has successfully synchronized with the network and "Tr" shows that a transmission is in process.
2. A transmission only occurs if all nodes are synchronized;
   Pmax=? [G !("Tr_A" &(!"Synch_A" | !"Synch_B"|...| !"Synch_n")]

3. At any given point in time there's only one node sending a message;
   Pmax=? [G ("Tr_A" ⇒ (!"Tr_B" & !"Tr_C" & . . . & !"Tr_n")]
4. A node only sends on its time partition;
   Pmax=? [G ("Tr_A" ⇒ "Time_A")]
   Where "Time_A" represents the time partition assigned to "A".
5. Every node has its own unique time partition;
   Pmax=? [G ("Tp_A" & "Tp_B" & . . . &"Tp_N")]
   Where: "Tp_A=(∀ B≠ A":Time_A≠ Time_B).
6. A sent packet will be received by the destination;
   Pmax=? [G ("Snd_A" ⇒ F "Rcv_B")]
   Where "Snd" represents that a packet has been sent and "Rcv" represents that a packet has been received.
7. To every message sent "Snd_A" there would be an acknowledgement"Ack_A";
   Pmax=? [G ("Snd_A" ⇒ F "Ack_A")].
8. Every transmission (Tr_*A*) will eventually end (Tend_*A*).
   Pmax=? [G ("Tr_*A*"⇒ F "Tend_*A*")].

In a control verification, assuming no hardware failure would even occur, both scenarios were successfully verified for all properties, validating them within TTE specification. Then a second verification for each scenario was performed, this time having a set probability for each hardware component to fail. In this second run, a state was found within the second scenario in which two of the properties did not hold true.

Table 1 displays the verification times and results for all eight properties of the second scenario. The faulty state violates the third and the fourth properties and it may be reached when one of the switches and one of the clients fail at the exact same time. Whenever a switch fails, communication will be frozen and the backup switch will boot and the synchronization function will synchronize the clocks of both switches and then broadcast the new clock-synch to the entire network, allowing communication to resume. In this particular case where the client fails at the same time as the switch, there is a chance that the random data from the clients non-silent failure will collide with the broadcast message from the switch, making the failure to pass unnoticed by the guardians. After this, there is a chance that the second garbage message from the failing client will collide with the new clock synch coming from the switches and, after this, theres a chance that another node will be assigned to the same time partition that is being used by the failing client. When this state is reached, well have two nodes assigned to the same time partition, plus one failing node which is not yet known to the networks guardians.

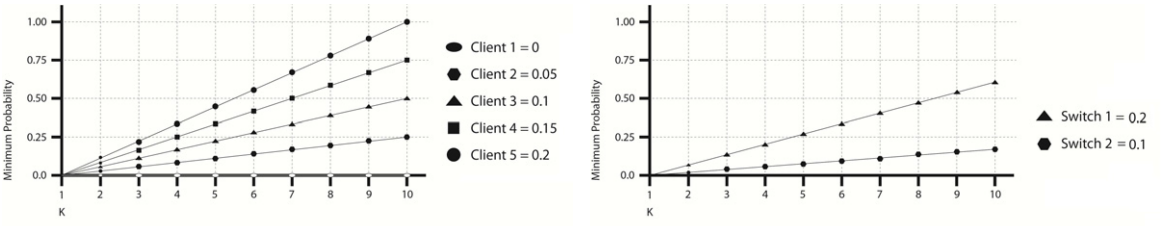
Table 1: Verification Results

Property #	1	2	3	4	5	6	7	8
Time	0.33	0.33	0.31	0.29	0.30	0.25	0.35	0.28
Result	1	1	0	0	1	1	1	1

Figure 4(a) and 4(b) show the plotted results of failure rates pertaining Clients and Switches in network scenario 2, assuming components with different failure probabilities. Each components likeness to fail is given in function of transmission round (K). In this test, "Client 1" is the control node with zero chance of failure. The remaining clients probability to fail increase at a constant rate of 0.05 (per cent) -a value arbitrarily chosen just for the sake of this test. The progression of the failure rate is linear for all clients and all switches, due to the deterministic nature of TTE.

4. Related Work and Conclusions

Most related work focus on design and implementation of TTE protocol. [5] develops a TTE controller for real-time applications, built on top of a 100Mbit Ethernet controller. It covers TTE architecture and



(a) Client. (b) Switch.

Fig. 4: Failure Rates.

functionality. [6] presents how TTE switch handles Time and Event-triggered messages and discussing its general behavior at low-level and how the switch handles predictable real-time communication. [2] have a presentation of the elements that are present in a TTE setting including its main functions and services, its basic definitions, and the protocol's requirements. [4] discuss an integration of TTE with SpaceWire networks for aerospace applications. This work covers the dataflow and synchronization in TTE. [7] focuses on the Compression Function of TTE, which is a clock synchronization function that runs inside the TTE switches collecting clock data from the connected systems.

[8] implements the specifications of a synchronous model on a distributed real-time platform while maintaining the semantic equivalence between the model and the implementation. [9] discourses about synchronous fault-tolerant systems by presenting how synchronous systems can be implemented as time-triggered (TT) systems. [10] extends [9] by generalizing the theory to accommodate time-triggered properties, like event-trigger, communication delays, reception windows, non-static clock and pipelined rounds.

Compared to the related work, we used probabilistic timed automata to model two real-time network scenarios using TTE protocol and verified them by using PRISM Model checker, to build a high-level software implementation. Our models are the most accurate form to represent a TTE network on the PRISM tool. In addition, we have proposed two techniques to accelerate further the checking of the models. We were able to identify one circumstance where TTE's properties will not be respected, likely resulting in catastrophic network failure.

References

[1] T. Group, TTE Specification, http://www.ttagroup.org., last visited: July 2012.
[2] H. Kopetz, A. Ademaj, P. Grillinger, K. Steinhammer, The Time-Triggered Ethernet (TTE) Design, in: Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, ISORC '05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 22–33.
[3] P. Team, PRISM - Probabilistic Symbolic Model Checker, http://www.prismmodelchecker.org, last visited: June 2011.
[4] D. J. W. Steiner, G. Bauer, Ethernet for Space Applications, in: S. T. Centre (Ed.), International SpaceWire Conference,Nara, 2008.
[5] P. Grillinger, A. Ademaj, K. Steinhammer, H. Kopetz, Software implementation of a time-triggered ethernet controller, in: IEEE International Workshop on Factory Communication Systems, 2006.
[6] K. Steinhammer, P. Grillinger, A. Ademaj, H. Kopetz, A Time-Triggered Ethernet (TTE) Switch, in: Proceedings of the Conference on Design, Automation and Test in Europe, DATE '06, 2006.
[7] W. Steiner, B. Dutertre, SMT-Based Formal Verification of a TTEthernet Synchronization Function, in: Proceedings of the 15th international conference on Formal methods for industrial critical systems, FMICS'10, 2010, pp. 148–163.
[8] S. Tripakis, C. Pinello, A. Benveniste, A. L. Sangiovanni-Vincentelli, P. Caspi, M. D. Natale, Implementing Synchronous Models on Loosely Time Triggered Architectures, IEEE Trans. Computers 57 (10).
[9] J. Rushby, Systematic Formal Verification for Fault-Tolerant Time-Triggered Algorithms, IEEE Transactions on Software Engineering 25 (5) (1999) 651–660.
[10] L. Pike, Modeling Time-Triggered Protocols and Verifying Their Real-Time Schedules, in: Proceedings of the Formal Methods in Computer Aided Design, FMCAD '07, 2007, pp. 231–238.