Benchmark for Verification of Fault-Tolerant Clock Synchronization Algorithms

(Benchmark Proposal)

Sergiy Bogomolov[‡], Christian Herrera[⋆,1], and Wilfried Steiner[†]

[⋆] Albert-Ludwigs-Universität Freiburg, Freiburg, Germany
[†] TTTech Computertechnik AG, Chip IP Design, Vienna, Austria
[‡] IST Austria, Vienna, Austria

Abstract

In this paper, we propose a benchmark for verification of properties of fault-tolerant clock synchronization algorithms, namely, a benchmark of a TTEthernet network, where properties of the clock synchronization algorithm as implemented in a TTEthernet network can be verified, and optimization techniques for verification purposes can be applied. Our benchmark, which assumes non-faulty components, aims to be a basis for verifying configurations which include faulty components, information consistency mechanisms, and for verifying other clock synchronization algorithms.

1 Introduction

Distributed real-time systems are present in many commercial hardware and software products, e.g. the avionics of the *Orion Space Program* [1]. These systems require a generic architecture that complies with even the most demanding safety-critical real-time requirements.
TTEthernet is an implementation of the traditional *Ethernet* standard which complies with time-critical, deterministic and safety-critical real-time requirements [2]. Safety critical systems using the TTEthernet standard rely on the availability of a global time base for tolerating faulty behavior of these systems. This global time base is a common perception of time for the distributed components of these systems, i.e. any two logical clocks of two distributed components must read the same values at any time. In the TTEthernet standard this common perception of time can be established by the periodic synchronization of the physical local clock of each component. This synchronization is performed by an internal clock synchronization algorithm [2] which compensates for the physical imperfections of these clocks. The maximal difference between the values of two logical clocks of two components is the *precision* achieved by the algorithm.

A typical TTEthernet *network* consists of switches and end systems connected by a communication channel, i.e. a bidirectional point-to-point link. A *standard configuration* consists of end systems connected to a single switch, while a *fault-tolerant configuration* consists of end systems connected to two independent switches [2]. Each switch belongs to one and only one communication channel. In this paper, we present a benchmark inspired by the fault-tolerant configuration of a TTEthernet network described in [3], and we use the framework of hybrid automata [4, 5, 6] to analyze and model the exhibiting complex continuous behavior of a network using this configuration. Our benchmark aims to be a simplified model where properties of the TTEthernet network, e.g. the precision of the synchronization algorithm, can be verified, and optimization techniques for verification purposes, e.g. detection and reduction of *quasi-dependent variables* [7], can be applied.

[1] CONACYT (Mexico) and DAAD (Germany) sponsor the work of this author.

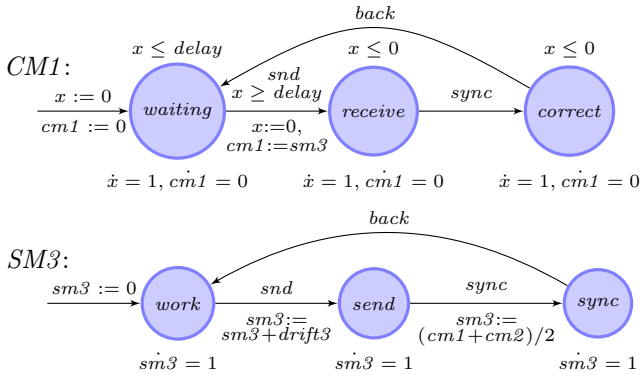

Figure 1: Components *CM1* and *SM3* of a network with two CMs and five SMs.

The paper is organized as follows. Section 2 presents an overview of the clock synchronization algorithm. Section 3 describes our proposed benchmark. Section 4 presents the results of our experiments. We conclude in Section 5.

2 Clock Synchronization Overview

In a network with a fault-tolerant configuration, switches and end systems assume the roles of *Compression Master* (CM), and *Synchronization Master* (SM), respectively. Furthermore, each SM is connected to each CM by one and only one communication channel. In the clock synchronization algorithm which we use, SMs and CMs send information to each other, e.g. the current value of the local clock of a given SM, by using *Protocol Control Frames* (PCF).

The clock synchronization algorithm which we use in our benchmark consists of two steps. Firstly, each SM sends a PCF to each linked CM. Then each CM extracts from the arrival point in time of the sent PCF the current value of the local clock of a given SM, with this information then the CM executes a first compression function to obtain the median from the received clock values of each SM. Secondly, each CM sends in a new PCF the result of the first compression function to all SMs, then each SM executes a second compression function in order to obtain the median of the values received from the CMs. The result of this second function is used to correct the value of the clock of each SM.
The algorithm from above describes the steps for synchronizing clocks in a network with non-faulty components, however, the interested reader can find in [3] a more detailed description of the clock synchronization algorithm in networks with faulty components.

In the next section, we present our benchmark as a network of hybrid automata, discuss optimizations for verification purposes, and briefly mention some possible extensions.

3 A Benchmark of a Fault-Tolerant TTEthernet Network

In the following, we use the definitions of hybrid automata as described in [7]. For simplicity we propose a typical industrial network of hybrid automata using a fault-tolerant configuration with two CMs, namely, *CM1* and *CM2*, and five SMs from *SM1* to *SM5*. In Figure 1 automaton *CM1* consists of the real variables x (the local clock with *rate* 1), and *cm1* which stores the result of the first compression function, and the locations *waiting* (initial), *receive* and *correct*,

while automaton *SM3* consists of the real variables *sm3* (the local clock with rate 1); *drift3* which ranges from $-maxdrift$ to *maxdrift*, where *maxdrift* describes the absolute value of the maximum drift offset that a SM's clock can achieve before the execution of the synchronization algorithm, and the locations *work* (initial), *send* and *sync*. The exchange of PCFs between CMs and SMs is realized in our benchmark by using edges in SMs and CMs labeled with *snd* and *sync*. The rest of the CMs and SMs follow a similar structure.

At the start of the system each automaton delays exactly *delay* time units, with $delay > 0$, at the unique location where a delay greater than 0 time units is possible, namely, at its initial location. Then after this delay the CMs and SMs respectively transit simultaneously to locations *receive* and *send* by taking the edges labeled with *snd*. By performing this transition each SM sends the drifted value of their clocks to both CMs. Then *CM1* and *CM2* store in their variables *cm1* and *cm2*, respectively, the result of the first compression function, which in Figure 1 we assume is the drifted value of the clock *sm3*. The second compression function, performed to correct the value of the clock of each SM, is executed when CMs and SMs transit simultaneously to locations *correct* and *sync*, respectively, by taking the edges labeled with *sync*. Finally, all automata return to their initial location by simultaneously taking the edges labeled with *back*.

It is important to point out that we assume non-faulty components, hence, consistency mechanisms for detecting PCFs from faulty components are not included in our benchmark.

3.1 Optimization for Verification Purposes

Our benchmark is a candidate for the optimization techniques that we have presented in [7], namely, detection and reduction of quasi-dependent variables. Quasi-dependency of variables is a generalization of *quasi-equality* of clocks [8, 9, 10]. Intuitively, between two variables x and y of a hybrid automaton $\mathcal{H}$, there exists a quasi-dependency, namely, x quasi-depends on y via function f, if and only if at all runs of $\mathcal{H}$ and at all points in time, the value of x is the value of f applied to the value of y, except when the value of y is determined by an update action.

The technique in [7] allows us to detect in our TTEthernet network two sets of quasi-dependent variables, namely, the set consisting of clocks from each SM, and the set consisting of clocks from each CM. This detection allows us as well to implement a reduction of the detected quasi-dependent variables together with a syntactical transformation of the original network, in order to produce a transformed network where the original complexity is reduced, and where properties of the original network are reflected, that is, a forbidden configuration is reachable in the transformed network if and only if it is reachable in the original network. The most remarkable result of this transformation is a dramatic performance improvement of the verification time of properties of the transformed network. We refer the interested reader to [7] for more details on the detection and reduction of quasi-dependent variables.

3.2 Possible Extensions of the Benchmark

In the following, we mention some possible extensions and uses for our benchmark:

1. it can be used as a basis for modeling TTEthernet networks where we can verify the precision of the synchronization algorithm under failures of a single SM, a single CM, and under concurrent SM and CM failures as studied in [3],
2. it can be also used as a basis for verifying clock synchronization algorithms like the *interactive convergence algorithm*[11] and the *byzantine clock synchronization*[11],

3. it can be useful for studying and implementing the elimination of the remaining syntactical assumptions for networks of hybrid automata, similar to the work for networks of timed automata [12] with quasi-equal clocks as described in [10], and
4. it can be used for scalability analysis by introducing additional CMs and SMs.

3.3 Open Problems of the Benchmark

Note that in our benchmark we assume that the rate of each clock is 1. However, in practice this assumption may not always hold due to the imperfection of the physical clocks, for instance, the clock of a SM may have rate 1 for at most n time units before dropping below 1, that is, that clock will tick slower than rate 1 after n time units. In this case a rate correction algorithm as in [13] will correct the rate of that clock. A more realistic benchmark would consider scenarios where the clock of a given SM has several rates before the execution of the synchronization algorithm. However, remains unclear how to detect and reduce quasi-dependent variables in benchmarks with the mentioned scenarios, since that detection and reduction assumes that the rate of the variables in a model is constant at all points in time. Therefore, further studies wrt. quasi-dependent variables with different rates for the same variables are required.

4 Experiments

In the following section, we present the results of our experiments, where our aim is to show that in our benchmark we have: (a) verified that the precision of the synchronization algorithm holds and, (b) applied the techniques for detecting and reducing quasi-dependent variables. We have verified the precision of the synchronization algorithm in settings of our benchmark with 2 CMs and from 5 to 30 SMs. We recall that our benchmark is a fault-tolerant TTEthernet network with non-faulty components. In this benchmark, we verify that the maximal difference between the values of any two logical clocks of two SMs is bounded by $2 * maxdrift$, as reported in [3], i.e. $\forall\, i \neq j \in \mathbb{N} \bullet\ sm_i > sm_j \implies sm_i - sm_j \leq 2 * maxdrift$. In addition, in order to enable efficient handling of large scale benchmark instances, we have applied a model transformation based on quasi-dependent variables [7]. The results for both, original and transformed networks, are reported in Table 1. In this table, we compare the analysis runtime needed by the model checker SpaceEx [14, 15, 16] for the original and the transformed networks (the latter denoted in the table by the suffix K). Note that transformed models use only one clock for the CMs and one clock for the SMs. We observe that the transformation leads to a drastic performance improvement due to the reduction of quasi-dependent variables.

Network	C	M	$t(s)$
TT-5	7	3.0	31.32
TT-5K	2	2.9	11.96
TT-20	22	3.0	124.08
TT-20K	2	3.0	12.11
TT-30	32	3.1	201.21
TT-30K	2	3.0	12.57

Experimental environment: Intel i3, 2.3 GHz, 3 GB, Ubuntu 11.04, SpaceEx server VM (VMX) v0.9.8b / PHAVer scenario.

Table 1: Row X-$N(K)$ gives the figures for case study X with N components, the suffix 'K' denotes the models after the quasi-dependent variables transformation, 'C' gives the number of clocks in the model, 'M' memory usage in MB for the analysis, and '$t(s)$' verification time in seconds. Detection of clocks does not contribute to the verification time.

5 Conclusion

We have presented a benchmark inspired by the fault-tolerant configuration of a TTEthernet network, and we have used the framework of hybrid automata to analyze and model the exhibiting complex continuous behavior of that network. Our benchmark is a simplified model where properties of the TTEthernet network can be verified, and optimization techniques for verification purposes can be applied. The benchmark can incorporate new CMs and SMs and be used in scalability analysis. Furthermore, our benchmark can be extended and used for verifying synchronization algorithms under failures of its components. Interesting open problems wrt. this benchmark require further studies, e.g. clocks in components with multiples rates for some time units and their implications wrt. detection and reduction of quasi-dependent variables.

References

[1] C. E. Howard. *Orion avionics employ COTS technologies*. *Avionics Intelligence*, 2009.

[2] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer. *The Time-Triggered Ethernet (TTE) Design*. In *ISORC*, pages 22–33. IEEE Computer Society, 2005.

[3] W. Steiner and B. Dutertre. *Automated Formal Verification of the TTEthernet Synchronization Quality*. In *NASA Formal Methods*, volume 6617 of *LNCS*, pages 375–390. Springer, 2011.

[4] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. *The Algorithmic Analysis of Hybrid Systems*. *TCS*, 138(1):3–34, 1995.

[5] S. Bogomolov, A. Donzé, G. Frehse, R. Grosu, T. Johnson, H. Ladan, A. Podelski, and M. Wehrle. *Abstraction-Based Guided Search for Hybrid Systems*. In *SPIN*, LNCS. Springer, 2013.

[6] S. Bogomolov, G. Frehse, R. Grosu, H. Ladan, A. Podelski, and M. Wehrle. *A Box-based Distance between Regions for Guiding the Reachability Analysis of SpaceEx*. In *CAV*, volume 7358 of *LNCS*, pages 479–494. Springer, 2012.

[7] S. Bogomolov, C. Herrera, M. Muñiz, B. Westphal, and A. Podelski. *Quasi-dependent variables in hybrid automata*. In *HSCC*, pages 93–102. ACM, 2014.

[8] C. Herrera, B. Westphal, et al. *Reducing Quasi-Equal Clocks in Networks of Timed Automata*. In *FORMATS*, volume 7595 of *LNCS*, pages 155–170. Springer, 2012.

[9] C. Herrera, B. Westphal, and A. Podelski. *Quasi-Equal Clock Reduction: More Networks, More Queries*. In *TACAS*, volume 8413 of *LNCS*, pages 295–309. Springer, 2014.

[10] C. Herrera and B. Westphal. *Quasi-equal Clock Reduction: Eliminating Assumptions on Networks*. In *HVC*, volume 9434 of *LNCS*, pages 173–189. Springer, 2015.

[11] L. Lamport and P. M. Melliar-Smith. *Byzantine Clock Synchronization*. In *PDC*, pages 68–74. ACM, 1984.

[12] R. Alur and D. Dill. *A Theory of Timed Automata*. *TCS*, 126(2):183–235, 1994.

[13] W. Steiner and B. Dutertre. *Layered Diagnosis and Clock-Rate Correction for the TTEthernet Clock Synchronization Protocol*. In *PRDC*, pages 244–253. IEEE Computer Society, 2011.

[14] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O.Maler. *SpaceEx: Scalable Verification of Hybrid Systems*. In *CAV*, volume 6806 of *LNCS*, pages 379–395. Springer, 2011.

[15] S. Bogomolov, A. Donzé, G. Frehse, R. Grosu, T. Johnson, H. Ladan, A. Podelski, and M. Wehrle. *Guided search for hybrid systems based on coarse-grained space abstractions*. *JSTTT*, pages 1–19, 2015.

[16] S. Bogomolov, G. Frehse, M. Greitschus, R. Grosu, C. S. Pasareanu, A. Podelski, and T. Strump. *Assume-Guarantee Abstraction Refinement Meets Hybrid Systems*. In *HVC*, volume 8855 of *LNCS*, pages 116–131. Springer, 2014.