Verification of the Schedule Consistent Update Mechanisms of FTTRS with UPPAAL

Daniel Bujosa, Sergi Arguimbau, Patricia Arguimbau, Julián Proenza, Manuel Barranco
[daniel.bujosa, sergi.arguimbau, patricia.arguimbau, julian.proenza, manuel.barranco]@uib.es
Dept. de Matemàtiques i Informàtica, Universitat de les Illes Balears, Palma de Mallorca, Spain

ABSTRACT

Critical Adaptive Distributed Embedded Systems (ADESs) are nowadays the focus of many researchers. ADESs are real-time and dependable control systems that are expected to play a key role to appropriately interact with physical systems whose operational conditions change at runtime, even in unpredictable manners. To provide ADESs with an adequate communication infrastructure, our research group proposed the *Flexible-Time-Triggered-Replicated Star* (FTTRS) [1], which is the most reliable implementation of the *Flexible-Time-Triggered* (FTT) paradigm [2] on top of Ethernet.

FTT is a master multi-slave publisher-subscriber paradigm that supports adaptivity by means of *real-time flexibility* (holds periodic and aperiodic traffics with different real-time constraints) and *operational flexibility* (allows updating the traffic schedule at runtime). On the one hand, the so-called FTT master organizes the communication as a sequence of rounds called *Elementary Cycles* (ECs). The EC starts by the FTT master broadcasting what is called the *Trigger Message* (TM). The TM allows slaves to synchronize with the master and polls the periodic messages to be transmitted in that EC according to the current traffic schedule. The TM is followed first by the *Synchronous Window* (SW), in which the appropriate slaves transmit the polled periodic messages and, second, by the *Asynchronous Window* (AW), where slaves transmit aperiodic messages. On the other hand, a slave can request the FTT master for an update in the schedule. When so the master subjects the update to admission control; notifies the slaves about its decision; and if it has accepted the update, it also reflects the new schedule in the TMs from then on.

FTTRS basically consists of a duplicated full-duplex Ethernet star in which each switch embeds an FTT master. Each slave connects to each switch by means of a link composed of a separated uplink and downlink. Both switches are interconnected by means of several full duplex interlinks, through which they exchange the TMs and all the traffic they receive from the non-faulty slaves.

FTTRS deals with permanent and temporary *non-malicious operational hardware faults* [3]. First, thanks to its redundant paths, FTTRS tolerates permanent faults affecting the switches and the links. Second, in FTTRS each critical message can be pro-actively retransmitted to timely tolerate temporary faults affecting the links. In particular, each FTTRS master always proactively retransmits several replicas of the TM to guarantee its successful broadcast.

Faults can manifest arbitrarily, however each switch/master is internally duplicated to exhibit crash failure semantics. Also, each switch/master includes a Port Guardian (PG) per port to contain errors. In this way, faults affecting a slave or its link are perceived by the rest of the system as that slave omitting or sending messages with an incorrect (application) payload.

To appropriately provide fault tolerance, both masters/switches of FTTRS must act as if they were a single one, i.e. they must be *replica determinate* [4]. FTTRS includes several mechanisms to enforce this replica determinism [1]. For instance, both masters isochronously broadcast several replicas of the same TM in lockstep.

Of all the FTTRS's replica determinism mechanisms, in this poster we are interested in the ones that FTTRS includes to consistently update the traffic schedule at runtime. This is so because these later mechanisms are fundamental for FTTRS to provide high reliability while keeping the most distinguishing advantage of FTT, i.e its real-time operational flexibility. In this sense note that [1] argues for the correctness of these FTTRS's consistent schedule update mechanisms. However, since these mechanisms are quite complex, the use of formal methods provides a more reliable way to check their correctness. Thus, in this poster we model and formally verify these complex mechanisms by using a model checker called UPPAAL [5], which is specially suited for real-time systems.

First, the presented poster explains the FTTRS's schedule consistent update mechanisms. Afterwards it provides a brief introduction to the UPPAAL model checker that, then, allows showing how we modeled the master/switches, the slaves and the update mechanisms. Finally, it shows how we verified the correctness of these mechanisms by checking some properties with UPPAAL and, then, draws some conclusions.

ACKNOWLEDGEMENTS

This work is supported in part by the Spanish Agencia Estatal de Investigación (AEI) and in part by FEDER funding through grant TEC2015-70313-R (AEI/FEDER, UE).

REFERENCES

[1] D. Gessner, J. Proenza, M. A. Barranco. "Adding Fault Tolerance To a Flexible Real-Time Ethernet Network for Embedded Systems". *PhD thesis*, *University of the Balearic Islands,* 2017.

[2] L. Almeida, P. Pedreiras, J.A. Fonseca. "The FTT-CAN Protocol: Why and How". *In: IEEE Transactions on Industrial Electronics* 49.6 (2002), pp. 1189–1201.

[3] Algirdas Avižienis et al. "Basic Concepts and Taxonomy of Dependable and Secure Computing". *In: IEEE Transactions on Dependable and Secure Computing* 1.1 (2004), pp. 11–33.

[4] S. Poledna. "Fault-Tolerant Real-Time Systems. The Problem of Replica Determinism". *The Springer International Series in Engineering and Computer Science*, Springer US, 1996.

[5] G. Behrmann, A. David, K.G. Larsen. "A Tutorial on Uppaal". *In: Formal Methods for the Design of Real-Time Systems. Lecture Notes in Computer Science*, vol 3185. Springer, Berlin, Heidelberg, 2004.