# Simulation of the Proactive Transmission of Replicated Frames Mechanism over TSN

Inés Álvarez*, Drago Čavka†, Julián Proenza*, Manuel Barranco*
*Departament de Matemàtiques i Informàtica, Universitat de les Illes Balears, Spain,
{ines.alvarez, julian.proenza, manuel.barranco}@uib.es
†Faculty of Electrical Engineering, University of Banja Luka, Bosnia and Herzegovina,
drago.cavka@etf.unibl.org

***Abstract*—The Time-Sensitive Networking (TSN) Task Group (TG) is providing Ethernet with timing guarantees, reconfiguration services and fault tolerance mechanisms. Some of TSN's targeted applications are real-time critical applications, which must provide a correct service continuously. To support these applications the TSN TG standardised a spatial redundancy mechanism. Even though spatial redundancy can tolerate permanent and temporary faults, it is not cost-effective. Instead, temporary faults can be tolerated using time redundancy. We proposed the Proactive Transmission of Replicated Frames (PTRF) mechanism to tolerate temporary faults in the links. In this work we present a new PTRF approach, a PTRF simulation model and a comparison of the approaches using exhaustive fault injection.**

## I. Introduction

The Time-Sensitive Networking (TSN) Task Group (TG) is working to provide Ethernet with hard and soft real-time guarantees, network configuration capabilities and fault tolerance mechanisms. To do so the TSN TG proposed a series of standards that operate at the layer 2 of the network architecture, commonly referred to as TSN standards. Some of TSN's targeted applications interact with the environment in which they operate and, thus, they must provide their services in *real time*, i.e., they must produce their results within a bounded time. Furthermore, some of these applications are considered to be critical, as their failures can have catastrophic consequences. Thus, these applications must provide a correct service continuously, i.e., they must be highly reliable.

To increase reliability, the TSN TG standardised a *spatial redundancy* mechanism [1] [2], which consists in transmitting several copies of the same frame in parallel, each copy through a different path. Spatial redundancy is suited to tolerate *permanent faults*, but the need for additional hardware makes it expensive and increases the size and energy consumption of the system. Furthermore, when spatial redundancy is no longer available due to permanent faults, it is no longer possible to tolerate *temporary faults*. Thus, spatial redundancy is not the best choice to tolerate temporary faults. Instead, temporary faults can be tolerated using *time redundancy*, which is more cost-effective than spatial redundancy.

Nevertheless, the TSN TG does not propose any time redundancy mechanism in this level of the architecture designed to tolerate temporary faults in the channel. This is particularly important as temporary faults in the links are more likely to happen than permanent ones. Thus, tolerating both permanent and temporary faults in the channel using solely spatial redundancy may lead to a significant increase in the hardware of the system. In fact, the hardware increases with the number of faults that must be tolerated simultaneously.

TSN networks are compatible with higher layer mechanisms to tolerate temporary faults, such as those based on Automatic Repeat Request (ARQ) techniques. ARQ-based solutions rely on the transmission of *acknowledgement* (ACK) or *negative ACK* (NACK) messages and timeouts to trigger the retransmission of frames when they are lost. ARQ-based solutions are non-deterministic in terms of the bandwidth and the time required to carry out the successful transmission of frames in the presence of faults. Moreover, the jitter introduced by these solutions is high, as the end-to-end delay when using retransmissions is significantly higher than when retransmissions are not needed. Thus, ARQ-based solutions are not the best choice for real-time systems. Furthermore, ACKs and NACKs can also be affected by temporary faults, introducing new and more complex fault scenarios that need to be tolerated.

For these reasons, we propose to use proactive frame replication to tolerate temporary faults in the links of TSN-based networks. Specifically, we designed a mechanism to transmit several copies of each frame through the same link in a preventive manner. This way we aim at ensuring that at least one copy reaches the destination even in the presence of temporary faults. This technique is a better choice for real-time systems as it is deterministic in the resource consumption, it reduces jitter and in the worst-case scenario it requires less time and bandwidth than solutions based on ARQ.

The mechanism we designed is called *Proactive Transmission of Replicated Frames* (PTRF) and its main ideas were already presented in [3]. Specifically, in said work we presented two different design approaches of the PTRF mechanism. In this work we present a new approach of the mechanism, which is a first step towards dynamic fault tolerance. We also carry out a simulation on top of OMNeT++ [4] in order to check the feasibility and compare the three approaches. Specifically, we compare the approaches in terms of the number of fault scenarios they can tolerate using exhaustive fault injection.

## II. Related Work

The retransmission of frames is a common technique to tolerate temporary faults in the links and, in particular, the

proactive frame replication technique is commonly used to tolerate temporary faults in real-time systems. As has been mentioned, this technique is suitable for real-time systems as it is deterministic in the resource consumption and bounds the jitter introduced by retransmissions. Examples of systems that use this technique are the Time-Triggered Protocol [5] and the Flexible Time-Triggered Replicated Star for Ethernet [6].

As we already said, the TSN TG proposes the use of spatial redundancy to increase the reliability of Ethernet networks. More precisely, the IEEE Std 802.1Qca amendment for Path Control and Reservation [1] allows to create multiple paths between end-stations willing to communicate; whereas the IEEE Std 802.1CB standard for Frame Replication and Elimination for Reliability [2] defines how to replicate streams to transmit each replica through one of the paths created by IEEE 802.1Qca, and how to eliminate surplus replicas upon reception. In this way TSN can tolerate permanent faults in the network in a seamless manner, i.e., without introducing a failover time. The number of faults that can be tolerated depends on the number of independent paths available.

To the best of the authors' knowledge, the TSN TG has no plans for the standardisation of the proactive retransmission of frames in the layer 2 of the network architecture. Thus, we proposed the PTRF mechanism to work over TSN-based networks. We next describe said mechanism.

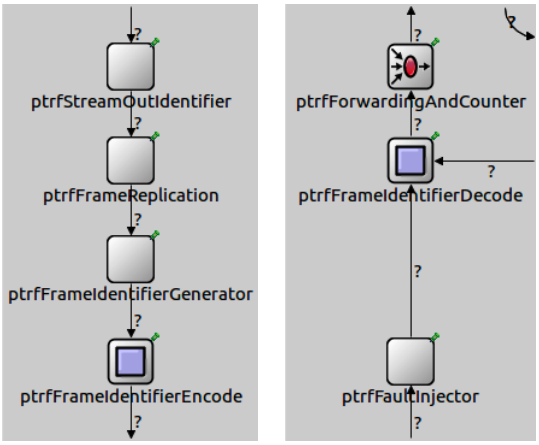
## III. Description of the PTRF mechanism

As mentioned, the PTRF mechanism is designed to tolerate temporary faults in the links. Thus, it must be used together with TSN's spatial redundancy if tolerating permanent faults is also needed. PTRF consists in transmitting several copies of each frame in a preventive manner to ensure that at least one copy reaches the destination. Each one of the copies created by the mechanism is called a replica. PTRF transmits all the replicas of a frame through the same path.

More concretely, PTRF is designed to replicate time-triggered (scheduled) frames, as these usually convey information that is critical for the correct operation of the system. Replicas are modified to carry special information which is used by PTRF to detect and eliminate surplus replicas upon reception. This way PTRF makes replication transparent for the application.

It is important to note that this mechanism works under the assumption that links exhibit omission failure semantics, i.e., we assume that a fault in the link causes the omission of a frame or replica. This is a reasonable assumption as Ethernet frames convey a CRC code used to detect errors in frames upon reception. Erroneous frames are identified and then dropped with a high probability, thereby manifesting as omissions.

We designed three different approaches of the PTRF mechanism. The first two approaches were presented in [3], whereas the third one is presented in this paper for the first time. These approaches differ in the devices that carry out replication and in the way replicas are handled. We next provide an overview of the three approaches.




(a) PTRF OMNeT++ module for transmission. (b) PTRF OMNeT++ module for reception.

Fig. 1: PTRF OMNeT++ modules.

- *End-to-end estimation and replication of frames.* In this approach only end-stations replicate frames during transmission and eliminate surplus replicas upon reception. On the other hand, it uses COTS bridges that simply forward all frames they receive. The number of replicas $k$ that end-stations must transmit is based on an end-to-end worst-case estimation of loss probability. As we use COTS bridges, if a frame is lost in a link that connects two bridges, the second bridge will only forward $k-1$ frames. We will refer to this approach as *approach A*.
- *End-to-end estimation, link-based replication of frames.* In this approach both, end-stations and bridges, replicate frames during transmission and eliminate surplus replicas upon reception. The number of replicas $k'$ sent by all devices is the same and is based on an end-to-end worst-case estimation. In this case, if a frame is lost in the link that connects two bridges, the second bridge will eliminate all replicas, except for one, and will transmit $k'$ replicas again. We will refer to this approach as *approach B*.
- *Link-based estimation and replication of frames.* In this approach, again, end-stations and bridges replicate frames during transmission and eliminate replicas upon reception. Nonetheless, now the number of replicas transmitted by any link $m$ of any device can vary depending on the loss probability of the forwarding link. Thus, each device may transmit a different number of replicas $k''_m$ through each link. We will refer to this approach as *approach C*.

We next describe the modules we developed in OMNeT++ to implement the PTRF mechanism, which are the same for the three approaches.

## IV. Simulation Model

We developed a simulation model of the PTRF mechanism. We used this model to check the feasibility of the PTRF mechanism and to compare the three approaches in terms of number of fault scenarios that they can tolerate. We

developed our model using OMNeT++ [4], a modular event-based simulation framework to model distributed systems and networks in C++. Furthermore, OMNeT++ counts with the INET library [7], which provides models for a series of wired, wireless and cellular network protocols, including Ethernet.

As a starting point, we had an already existing preliminary TSN simulation model called TSimNet [8], developed in the University of Siegen. TSimNet is built on top of INET and provides a subset of the TSN services, namely *stream identification*, *per-stream filtering and policing* and *frame replication and elimination for reliability* (spatial redundancy). Further details on these mechanisms can be found in [8]. Note that in this work we did not use spatial redundancy, as we only focus on our PTRF. A first approximation of the integration of spatial and time redundancy is presented in [9] and further work is left for the future.

Figure 1 shows the different PTRF modules. The left-hand side of the figure shows the modules for transmission; whereas the right-hand side shows the modules for reception and for fault injection. We next describe the different modules, but first we need to note that we modified the TSN frame provided by TSimNet to convey PTRF information. This information is used by receivers to identify and eliminate surplus replicas.

Figure 1a shows the modules for transmission. The *ptrfStreamOutIdentifier* module identifies scheduled frames to replicate them; the *ptrfFrameReplication* module generates the replicas; the *ptrfFrameIdentifierGenerator* module creates a unique identifier for each set of replicas of the same frame to differentiate them from replicas of other frames and, finally, the *ptrfFrameIdentifierEncode* module encapsulates the information in a PTRF message and sends it out.

Figure 1b shows the modules for the identification and elimination of replicas and the module for fault injection. Starting with PTRF, the *ptrfFrameIdentifierDecode* module identifies replicas in order to further process them and the *ptrfForwardingAndCounter* module decides whether the replica must be forwarded or not, depending on whether it is the first replica of a given frame to be received or not. Finally the *ptrfFaultInjector* drops frames to emulate the behaviour of a device that detects an erroneous frame using the frame's CRC. The ptrfFaultInjector of all receiving devices can cooperate in order to generate any fault scenario.

## V. Fault Scenarios Analyses

In this work we evaluate how the different PTRF approaches behave in front of temporary faults. Specifically, we study how many fault scenarios each approach can tolerate before losing information. We do it using the model, by means of exhaustive fault injection, i.e., we inject all combinations of faults that can affect the replicas of a frame and count which ones are tolerated by each approach. To assess the correctness of the results obtained using simulation, we did an analysis to count the combinations of fault scenarios tolerated by each approach.

Note that, in order to tolerate a fault scenario at least one replica must reach its destination. Thus, in the following analyses we consider all the possible scenarios where at least one replica reaches the destination. We next describe the analysis done for each approach. For the sake of clarity, we will start describing the simplest analysis, which corresponds to approach B, as the other two analyses can be more easily derived from this one.

- *Approach B*. In approach B, end-stations and bridges send $k'$ replicas. That is, $k'$ replicas are transmitted through each link as long as 1 replica reaches the bridge. Thus, approach B can tolerate all fault scenarios where up to $k'-1$ replicas are lost in each link. Equation 1 shows the number of tolerated scenarios,

$$\left(\sum_{e'=0}^{k'-1}\binom{k'}{e'}\right)^{l} \tag{1}$$

  where $k'$ is the number of replicas transmitted by all components, $e'$ is the number of faults that happen in each link and $l$ denotes the number of links in the path.
- *Approach C*. In approach C, end-stations and bridges may send a different number of replicas $k''_m$ through each link. Receiving 1 replica is enough for a bridge to generate $k''_m$ replicas again. Thus, approach C can tolerate all fault scenarios where up to $k''_m-1$ replicas are lost in each link. Equation 2 shows the number of tolerated scenarios,

$$\prod_{m=1}^{l}\sum_{e''=0}^{k''_m-1}\binom{k''_m}{e''} \tag{2}$$

  where $k''_m$ is the number of replicas transmitted through link $m$, $e''$ is the number of faults in said link and $l$ is the number of links in the path.
- *Approach A*. In approach A end-stations replicate frames and bridges simply forward the frames they receive. Thus, as bridges do not generate new replicas, approach A can only tolerate fault scenarios where up to $k-1$ replicas are lost in the whole path. The combinations of fault scenarios in each link must account for the faults occurred in other links. Equation 3 shows the number of tolerated scenarios,

$$\sum_{e_1=0}^{k-1}\cdots\sum_{e_l=0}^{(k-e_1-\cdots e_{l-1})-1}\left(\prod_{m=1}^{l}\binom{k-\sum_{i=1}^{m-1}e_i}{e_m}\right) \tag{3}$$

  where $k$ is the number of replicas sent by the end-station, $e_m$ is the number of faults in link $m$ and $l$ is the number of links. The term $k-\sum_{i=1}^{m-1}e_i$ limits the replicas transmitted in the current link according to the faults occurred in all previous links. Note that in the first link $m$ is 1, so we have that $\sum_{i=1}^{m-1}e_i=\sum_{i=1}^{0}e_i=0$ faults.

## VI. Results

We used the model described in Section IV to check the feasibility of the design of the approaches and to compare their behaviour in front of faults. To do it, we used exhaustive fault injection, i.e., we injected all possible combinations of fault scenarios that can affect the replicas of a frame and we checked which ones are tolerated by each approach. We also used the analyses presented in the previous Section to validate

TABLE I: Network parameters and results in fault injection experiments in a 7-hop network.

| Approach | Links | Replicas | Tolerated scenarios |
|---|---|---|---|
| A | 7 | 3 | 169 |
| B | 7 | 3 | 823543 |
| C | 7 | 2,2,2,3,3,4,4 | 297675 |

the results obtained with the model. We used a 7-hop network, as it is the maximum number of hops for which TSN ensures timing guarantees [10], even if in practical implementations the number of hops can be higher. Moreover, we used a line topology as TSN relies in specific protocols to eliminate loops.

Table I shows the parameters we used to carry out the simulations of the approaches and the results obtained. Regarding the number of replicas, we decided to use 3 replicas for approaches A and B, as it is sufficiently high to show the difference between the approaches, but it is still a realistic number of replicas. On the other hand, we used a variable number of replicas for approach C, which goes from 2 to 4. This is because approach C behaves as approach B when using the same number of replicas for all links and we want to highlight the differences among the approaches.

The results obtained show that the number of fault scenarios tolerated by each approach during the simulations corresponds to the number obtained using the analyses presented in the previous Section. Thus, we can conclude that it is feasible to build the approaches and that the design behaves as intended. An implementation on a real prototype is currently being developed to further study the approaches.

Regarding the number of scenarios, it is important to note that the actual reliability that is obtained with an approach is not directly proportional to the number of scenarios it tolerates. This means that, even though tolerating a higher number of fault scenarios in this case is likely to improve reliability, the actual impact on the reliability also depends on the probability of each scenario. Looking at the results we can see that approach A can tolerate a significant lower number of scenarios than approaches B and C, and we also see that reducing the number of replicas in approach C also impacts the number of scenarios. Nevertheless, the real impact on reliability requires a reliability analysis, which is left for future work.

## VII. Conclusions

The TSN TG is working to provide Ethernet with hard and soft real-time guarantees, network configuration capabilities and fault tolerance mechanisms. Specifically, the TSN TG proposed two standards to support spatial redundancy to tolerate faults in Ethernet networks. Nevertheless, even though spatial redundancy is suited to tolerate permanent faults, it is not the best choice for temporary faults, as it is expensive and implies an increase in the size and energy consumption of the system.

Instead, temporary faults can be tolerated using time redundancy. TSN standards can be used together with higher-layer techniques, such as those based in ARQ. Nevertheless, these solutions are not the best choice for real-time systems as they rely on timeouts and special messages to trigger retransmissions; which introduces a high jitter, reduced efficiency when the maximum number of faults occur and new fault scenarios that must be tolerated.

We proposed the PTRF mechanism to tolerate temporary faults using proactive frame replication. PTRF consists in transmitting several copies of each frame in a preventive manner to ensure that at least one copy reaches the destination even in the presence of temporary faults. We proposed three approaches of this technique, of which the third one was presented in this paper. We developed a simulation model and used it to inject all combinations of fault scenarios to see which ones are tolerated by each approach. Furthermore, we made a fault combination analysis to validate the results obtained with the simulation.

The results obtained with the simulation and the analysis were the same. This results allowed us to assess the feasibility of the three approaches. We saw that approach A can tolerate a lower number of fault scenarios than approaches B and C, and that the reduction in the number of replicas significantly impacts the number of tolerated scenarios. Quantifying the impact that this has on the reliability is left as future work.

## VIII. 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). Drago Čavka were supported by a scholarship of the EUROWEB+ Project, which is funded by the Erasmus Mundus Action II programme of the European Commission.

## References

[1] "IEEE Standard for Local and Metropolitan Area Networks– Bridges and Bridged Networks - Amendment 24: Path Control and Reservation," *IEEE Std 802.1Qca-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qcd-2015 and IEEE Std 802.1Q-2014/Cor 1-2015)*, pp. 1–120, March 2016.

[2] "IEEE Standard for Local and Metropolitan Area Networks–Frame Replication and Elimination for Reliability," *IEEE Std 802.1CB-2017*, pp. 1–102, Oct 2017.

[3] I. Álvarez, J. Proenza, M. Barranco, and M. Knezic, "Towards a time redundancy mechanism for critical frames in Time-Sensitive Networking," in *Proceedings of the 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sept 2017, pp. 1–4.

[4] A. Varga, "The OMNeT++ Discrete Event Simulation System," in *Proceedings of the European Simulation Multiconference (ESM)*, 2001.

[5] H. Kopetz and G. Grunsteidl, "TTP - A Protocol for Fault-Tolerant Real-Time Systems," *Computer*, vol. 27, no. 1, pp. 14–23, Jan 1994.

[6] D. Gessner, J. Proenza, M. Barranco, and A. Ballesteros, "A Fault-Tolerant Ethernet for Hard Real-Time Adaptive Systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 2980–2991, May 2019.

[7] "The INET Framework—An Open-Source OMNeT++ Model Suite for Wired, Wireless and Mobile Networks." [Online]. Available: https://inet.omnetpp.org/

[8] P. Heise, F. Geyer, and R. Obermaisser, "TSimNet: An Industrial Time Sensitive Networking Simulation Framework Based on OMNeT++," in *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Nov 2016, pp. 1–5.

[9] I. Álvarez, J. Proenza, and M. Barranco, "Mixing Time and Spatial Redundancy Over Time Sensitive Networking," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, June 2018, pp. 63–64.

[10] "IEEE Draft Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications," *IEEE P802.1AS-Rev/D6.0 December 2017*, Jan 2018.