Universitat
de les Illes Balears

DOCTORAL THESIS
2021

TIME REDUNDANCY MECHANISMS FOR TOLERATING TEMPORARY FAULTS IN THE COMMUNICATION SUBSYSTEM OF SYSTEMS BASED ON TIME-SENSITIVE NETWORKING STANDARDS

Inés Álvarez Vadillo

Universitat
de les Illes Balears

DOCTORAL THESIS
2021

Doctoral Programme of Information and Communications Technology

TIME REDUNDANCY MECHANISMS FOR TOLERATING TEMPORARY FAULTS IN THE COMMUNICATION SUBSYSTEM OF SYSTEMS BASED ON TIME-SENSITIVE NETWORKING STANDARDS

Inés Álvarez Vadillo

Thesis Supervisor: Dr. Julián Proenza Arenas
Thesis Supervisor: Dr. Manuel Alejandro Barranco González
Thesis tutor: Dr. Julián Proenza Arenas

Statement of Authorship

This thesis has been submitted to the Escola de Doctorat, Universitat de les Illes Balears, in fulfilment of the requirements for the degree of Doctora Internacional en Tecnologías de la Información y las Comunicaciones. I hereby declare that, except where specific reference is made to the work of others, the content of this dissertation is entirely my own work, describes my own research and has not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university.

Inés Álvarez Vadillo Palma, November 2021

Supervisors' Agreement

Julián Proenza, Ph.D. in Computer Science and *Profesor Titular de Universidad* at the Department of Mathematics and Computer Science, Universitat de les Illes Balears and Manuel Barranco, Ph.D. in Computer Science and *Profesor Titular de Universidad* of the same department

DECLARE

that the thesis titled *Time redundancy mechanisms for tolerating temporary faults in the communication subsystem of systems based on Time-Sensitive Networking standards*, presented by Inés Álvarez Vadillo to obtain the degree of *Doctora Internacional en Tecnologías de la Información y las Comunicaciones*, has been completed under our supervision and meets the requirements to opt for an International Doctorate.

For all intents and purposes, we hereby sign this document.

Dr. Julián Proenza Arenas

Dr. Manuel Alejandro Barranco González

Palma, November 2021

Abstract

In the last years several novel industrial applications have emerged, such as Industry 4.0, autonomous driving or intelligent energy distribution. On top of the intrinsic characteristics of traditional industrial applications, i.e. real-time constraints and increased dependability; these novel applications exhibit new requirements such as adaptivity, high inter-connectivity or integration of services with different operation requirements over a single infrastructure.

Because of their nature, the systems that support these applications are distributed, which means that the communication subsystem is fundamental to guarantee the proper operation of the overall system. Nonetheless, currently there is not a standardised communication protocol capable of fulfilling all the needs of the communication subsystems of these novel applications.

For this reason, the Time-Sensitive Networking (TSN) Task Group has been working on proposing a series of technical standards to provide Ethernet with hard real-time guarantees, on-line management of the traffic and fault tolerance mechanisms to increase *reliability*.

Specifically, TSN proposes using space redundancy to increase the reliability of Ethernet networks, which consists in transmitting each frame in parallel through several physical paths. Nevertheless, using space redundancy to tolerate temporary faults is not a cost-effective solution. For this reason, in this dissertation, we propose to use time redundancy to tolerate temporary faults in the links of TSN-based networks. Specifically, we present the Proactive Transmission of Replicated Frames mechanism, which consists in transmitting several copies of each frame in a preventive manner to increase the chances of at least one copy reaching its intended destination even in the presence of temporary faults. This mechanism is specially tailored to the specific characteristics of TSN.

In this dissertation we show that we can increase the reliability of TSN-based networks using PTRF to provide proactive time redundancy to tolerate

temporary faults in the links, while keeping the characteristics that make TSN an appealing technology for the communication subsystem of the aforementioned applications.

Abstract in Spanish

En los últimos años han emergido novedosas aplicaciones industriales, tales como la Industria 4.0, la conducción autónoma o la distribución inteligente de energía. Además de las características intrínsecas de las aplicaciones industriales, es decir, requisitos de tiempo real y una elevada garantía de funcionamiento; estas nuevas aplicaciones presentan nuevos requisitos, tales como la adaptividad, la elevada interconectividad o la integración de servicios con distintos requisitos operacionales sobre una única infraestructura.

Debido a su naturaleza, los sistemas que ejecutan estas aplicaciones son distribuidos, lo que convierte al subsistema de comunicaciones en una pieza fundamental para garantizar el correcto funcionamiento del sistema en su conjunto. Sin embargo, en la actualidad no hay ningún protocolo de comunicaciones que cubra todas las necesidades de los subsistemas de comunicaciones de estas nuevas aplicaciones. Por este motivo, el grupo de trabajo *Time-Sensitive Networking* (TSN) lleva años trabajando en una serie de estándares técnicos para dotar a Ethernet de garantías de tiempo real, gestión del tráfico en tiempo de funcionamiento y de mecanismos de tolerancia a fallos para incrementar la *fiabilidad*.

En concreto, TSN propone utilizar redundancia espacial para incrementar la fiabilidad de redes Ethernet, que consiste en enviar cada trama por varios caminos en paralelo. Sin embargo, usar redundancia espacial para tolerar fallos temporales no es una solución eficiente. Por esta razón, en esta monografía, proponemos utilizar redundancia temporal para tolerar fallos temporales en los enlaces de redes basadas en los estándares TSN. Concretamente, presentamos el mecanismo *Proactive Transmission of Replicated Frames* que consiste en transmitir varias copias de cada trama de manera preventiva para incrementar la probabilidad de que al menos una copia llegue a su destino correctamente, incluso en presencia de fallos temporales.

En esta monografía mostramos que podemos incrementar la fiabilidad de

redes basadas en TSN utilizando PTRF para dotar a la red de redundancia temporal para tolerar fallos temporales en los enlaces, todo ello manteniendo las características que hacen de TSN una tecnología adecuada para los subsistemas de comunicación de las aplicaciones arriba mencionadas.

Abstract in Catalan

En els últims anys han emergit noves aplicacions industrials, com ara la Indústria 4.0, la conducció autònoma o la distribució intel·ligent d'energia. A més de les característiques intrínseques de les aplicacions industrials, és a dir, requisits de temps real i una elevada garantia de funcionament; aquestes noves aplicacions presenten nous requisits, com ara, l'adaptabilitat, l'elevada interconnectivitat o la integració de serveis amb diferents requisits operacionals sobre una única infraestructura.

A causa de la seva naturalesa, els sistemes que executen aquestes aplicacions són distribuïts, la qual cosa converteix al subsistema de comunicacions en una peça fonamental per garantir el correcte funcionament del sistema en el seu conjunt. No obstant això, en l'actualitat no hi ha cap protocol de comunicacions que cobreixi totes les necessitats dels subsistemes de comunicacions d'aquestes noves aplicacions. Per aquest motiu, el grup de treball *Time-Sensitive Networking* (TSN) porta anys treballant en una sèrie d'estàndards tècnics per dotar a Ethernet de garanties de temps real, gestió del tràfic en temps de funcionament i de mecanismes de tolerància a fallades per incrementar la *fiabilitat*.

En concret, TSN proposa utilitzar redundància espacial per incrementar la fiabilitat de xarxes Ethernet, que consisteix a enviar cada trama per diversos camins en paral·lel. No obstant, usar redundància espacial per a tolerar fallades temporals no és una solució eficient. Per aquesta raó, en aquesta monografia, proposem utilitzar redundància temporal per a tolerar fallades temporals en els enllaços de xarxes basades en els estàndards TSN. Concretament, presentam el mecanisme *Proactive Transmission of Replicated Frames* que consisteix a transmetre diverses còpies de cada trama de manera preventiva per a incrementar la probabilitat que almenys una còpia arribi al seu destí correctament, fins i tot en presència de fallades temporals.

En aquesta monografia mostrem que podem incrementar la fiabilitat de

xarxes basades en TSN utilitzant PTRF per a dotar a la xarxa de redundància temporal per a tolerar fallades temporals en els enllaços, tot això mantenint les característiques que fan de TSN una tecnologia adequada per als subsistemes de comunicació de les aplicacions a dalt esmentades.

To my parents and my sister.

Invention is the most important product of man's creative brain. The ultimate purpose is the complete mastery of mind over the material world, the harnessing of human nature to human needs.

Nikola Tesla

Preface

Publications Arising from This Dissertation

We here list the publications that gather the results arising from this thesis, as well as the ones related directly or inderectly to the reasearch carried out through its development.

Publications that present direct results of this dissertation

This first set of publications describe the design, simulation, implementation and evaluation of the Proactive Transmission of Replicated Frames mechanism.

Peer-reviewed papers published in international journals

This paper presents the design, implementation and experimental evaluation of PTRF.

- I. Álvarez, I. Furió, J. Proenza, M. A. Barranco. *Design and Experimental Evaluation of the Proactive Transmission of Replicated Frames Mechanism over Time-Sensitive Networking.* In Sensors, MDPI, vol. 21, no. 3, January, 2021

Peer-reviewed papers published in international conferences

This paper presents a preliminary proposal of the PTRF mechanism.

- I. Álvarez, J. Proenza, M. A. Barranco, 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 2017), Cyprus, 2017

This paper presents a simulation and a first evaluation of the fault tolerance capacities of PTRF.

- I. Álvarez, D. Cavka, J. Proenza, M. A. Barranco. *Simulation of the Proactive Transmission of Replicated Frames Mechanism over TSN.* In Proceedings of the IEEE 24th International Conference on Emerging Technologies and Factory Automation (ETFA 2019), Zaragoza, 2019

Publications that present results related to this dissertation

These publications present different works that arouse from this dissertation but that are not part of its core.

Peer-reviewed papers published in international journals

This paper presents a survey about fault tolerance in Ethernet-based distributed embedded systems, which includes a survey of the related work of this dissertation.

- I. Álvarez, A. Ballesteros, M. A. Barranco, D. Gessner, S. Derasevic, J. Proenza. *Fault Tolerance in Highly-Reliable Ethernet-based Industrial Systems.* In Proceedings of the IEEE, vol. 107, no. 6, pp. 977-1010, June, 2019

Peer-reviewed papers published in international conferences

This paper presents a first proposal to integrate PTRF and the spatial redundancy mechanisms proposed in standard TSN.

- I. Álvarez, J. Proenza, M. A. Barranco. *Mixing Time and Spatial Redundancy over Time Sensitive Networking.* In Proceedings of the 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Luxemburg, 2018

This paper presents a first proposal of a complete fault-tolerant architecture based on TSN.

- I. Álvarez, M. A. Barranco, J. Proenza. *Towards a Fault-Tolerant Architecture based on Time Sensitive Networking.* In Proceedings of the IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA 2018), Torino, 2018

Publications co-authored during the development of this dissertation that benefit from this thesis although they do not present direct results

This journal paper presents a thorough comparison of two different admission control architectures for real-time Ethernet. More specifically, the paper presents a comparison between TSN's distributted control architecture and the Flexible Time-Triggered (FTT) centralised architecture.

- I. Álvarez, L. Moutinho, P. Pedreiras, D. Bujosa, J. Proenza, L. Almeida. *Comparing Admission Control Architectures for Real-Time Ethernet.* In IEEEAccess, IEEE, vol. 8, pp. 105521-105534, June, 2020

Publications co-authored prior the development of this dissertation

These conference papers present two preliminary comparisons between TSN's distributted admission control and FTT's centralised admission control architectures.

- I. Álvarez, L. Almeida, J. Proenza. *A First Qualitative Comparison of the Admission Control in FTT-SE, HaRTES and AVB.* In Proceedings of the 12th IEEE World Conference on Factory Communication Systems (WFCS 2016), Aveiro, 2016
- I. Álvarez, M. Knezic, L. Almeida, J. Proenza. *A First Performance Analysis of the Admission Control in the HaRTES Ethernet Switch.* In Proceedings of the 21th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2016), 2016

Acknowledgements

First and foremost I want to thank Julián Proenza and Manuel Barranco for their unconditional support during the development of this PhD. I can say with no mistake that without them I would have never completed this Thesis. They believed in me even before I did and they welcomed me in their group from the moment I started collaborating with them back in 2014. During these years they have taught me everything I know about doing research but, most importantly, they have showed me the joy of being part of a team that respects and supports all of its members. Not only they have been my supervisors, they are also my colleagues and friends. Without them I would not be the researcher and the person I am today.

I also want to thank Alberto Ortiz for his help regarding all the administrative issues, specially in the last months of this process. Without his support, submitting this dissertation would have been even more challenging.

I am specially thankful to Alberto Ballesteros, who has been my colleague since I started my journey in the *System, Robotics and Vision research group (SRV)*. He has always been there, willing to help me out with any problem I faced. I would also like to thank Daniel Bujosa and Ignasi Furió, with whom I have collaborated closely. They have both taught me things I would have not learnt otherwise.

Special thanks go to Toni Martorell, Emilio García, Xisco Bonnín, Joan Pep Company, Francesc Bonin, Miquel Massot and Eric Guerrero, all members of the SRV lab. Without them, my journey would have been lonely and sad. Also Julia Navarro, Xisca Hinarejos and Onofre Martorell, who have chosen to be by my side despite not being in the same research group. They have filled with joy many hours in the University.

I cannot forget about the persons who helped me when starting this journey. David Gessner, with whom I co-authored my first paper and who trusted me to present what was at the time part of the results of his PhD. His trust and his commitment to hard work helped me take the step to enroll in research. Luís Almeida, who welcomed me to his lab in Porto, giving me the opportunity to learn from his experience.

From my stay in Porto I also want to thank Paulo Pedreiras for his passion and time. From my stay in Banja Luka I want to thank Mladen Knezic and Aleksandar Pajkanović, for working with me and helping me during the 10 months I lived there. From my stay in Vienna I would like to thank Wilfried Steiner who welcomed me in his lab.

I have a special place in my heart for Luís Pinto, Sidney Carvalho, André Moreira, Aqsa Aslam, Pablo Gutiérrez and Marina Gutiérrez, who welcomed me not only to their groups, but also to their lives.

Finally, I want to thank my mother, my father, my sister and David, who have always trusted and supported me, long before I started this journey. Without them I would not be who I am today. My dearest friends Tomeu, Joana, Franco, Mateu and Pau, who are always willing to make me laugh and Daniel, who has always believed in me.

Funding

This work was 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).

During my research stay in the University of Banja Luka I was supported by a scholarship of the EUROWEB+ Project, which was funded by the Erasmus Mundus Action II programme of the European Commission.

Contents

List of Figures

List of Tables

Acronyms

ACK *acknowledgement*. 6

ARQ *automatic repeat request*. 6

BER *bit error rate*. 130

CBS Credit Based Shaper. 59

COTS *commercial off the shelf*. 37

CQF Cyclic Queuing and Forwarding. 71

CRC *cyclic redundancy check*. 20

DTMC *discrete-time Markov chains*. 139

ET *event-triggered*. 25

FT *fault tolerance*. 9

MAC *medium access control*. 25

MTSN Multiport TSN. 104

NACK *negative ACK*. 6

PTP Precision Time Protocol. 73

PTRF Proactive Transmission of Replicated Frames. 7

RT *real-time*. 2

SoC-e System on Chip. 104

SPoF *single point of failure*. 20

SRP Stream Reservation Protocol. 59

TAS Time-Aware Shaper. 69

TDMA *time-division multiple-access*. 25

TG Task Group. 3

TSN Time-Sensitive Networking. 3

TT *time-triggered*. 25

Chapter 1

Introduction

We start this chapter providing an overview of the context of this dissertation, then we move to the specific problem that we have addressed and we describe the aim of this dissertation. After that, we list the contributions of this dissertation, we further define its scope by describing what we do *not* cover and we end with an overview of what the reader will find in the remainder of the document.

1.1 Background

In the last two decades several novel industrial applications have emerged, such as Industry 4.0 systems (Wollschlaeger, Sauter, and Jasperneite, 2017), autonomous vehicles (Samii and Zinner, 2018) or efficient energy management infrastructures (Salazar et al., 2019). These applications usually have two common characteristics. First, they normally interact with the real world. This imposes tight timing constraints, so that these applications must provide their services in real time, i.e. they must produce their results within a bounded time (Kopetz, 2011a). Second, they frequently are considered to be critical, as their failure could have catastrophic consequences. Thus, they must be highly dependable in general; and *highly reliable* in particular (Åkerberg et al., 2021), i.e. they must provide a correct service throughout a given interval of time with a very high probability.

It is important to note that, because of their nature, the systems that support these novel applications are commonly large and distributed, i.e. these systems are usually comprised of several nodes that are physically

dispersed and that must exchange information to fulfil a common goal. We call these, *distributed systems* and their communication subsystem is fundamental to guarantee their correct operation. This means that the communications of distributed systems must exhibit the same characteristics as the system itself, i.e. for the overall system to be *real-time* (RT) and highly reliable the communication subsystem must allow nodes to exchange information in real time with high *reliability*.

Furthermore, the ever-growing complexity and size of these applications and the systems that support them are imposing new challenges in terms of the design of their communication subsystems. In particular, there is an urgent need to build communication subsystems capable of supporting the integration of the Information Technology (IT) and the Operation Technology (OT) over a single network infrastructure (Åkerberg et al., 2021). This integration is one of the multiple reasons for the transmission of traffic with diverse timing characteristics over the same network infrastructure, i.e. the network needs to provide *real-time flexibility*. On top of that, there is also a growing interest in building these systems in a way that they are capable of adapting to unforeseen changes in the environment or in the application requirements, e.g. the network must exhibit *operational flexibility*. Finally, these networks must have several bridges in order to connect all the nodes in the system, i.e. the networks must be *multi-hop* (Ashjaei et al., 2014).

Ethernet is a key technology in this context, given its advantages in terms of high bandwidth, low cost, ubiquity in IP-based networks, know-how and scalability. In fact, Ethernet is probably the network technology that has drawn the most attention in the last years not only in data communications, but also in the industrial context (Wollschlaeger, Sauter, and Jasperneite, 2017). Nevertheless, Ethernet was not designed to provide dependability nor RT guarantees (Wilamowski and Irwin, 2011) to the communications, which makes it an inadequate solution for the networks of many applications. As a result, a myriad of Ethernet-based communication mechanisms and protocols have been proposed to cope with these limitations (Finn, 2018), specially in the industrial context (Wollschlaeger, Sauter, and Jasperneite, 2017).

Nonetheless, each one of the existing Ethernet-based solutions addresses just one or, at most, only a subset of the dependability, RT and flexibility requirements that we have previously discussed (Álvarez et al., 2019). Certainly, some of these solutions could be combined among them to build networks that meet all the requirements of the systems capable of implementing the aforementioned applications, but the high heterogeneity of these solutions imposes strong compatibility or interoperability limitations between communication

subsystems that do rely on them. For all these reasons, the Time-Sensitive Networking (TSN) Task Group (TG) (*Time-Sensitive Networking (TSN) Task Group.*) from IEEE has been working on a set of technical standards to provide RT, dependable and flexible communications over Ethernet while overcoming the existing interoperability limitations. These standards are commonly referred to as TSN standards, and do provide standard Ethernet with enhanced services in the layer 2 of the network architecture. When properly combined, TSN standards allow to build *multi-hop* networks with *real-time* and *operational flexibility*.

For the specific case of dependability, the TSN TG has standardized two mechanisms in three different documents (802.1, 2016b; 802.1, 2017c; 802.1, 2017a), which are devoted to deal with faults affecting the channel, i.e. faults affecting links and bridges. The first mechanism is a scheme of *space redundancy*, which consists in providing the system with more hardware components than those strictly necessary for the system to work in the absence of faults(Johnson, 1988). Space redundancy in communication networks translates to the use of more bridges and links to allow transmitting several copies of a frame in parallel, each copy through a different physical path in order to tolerate faults. The second mechanism consists of a set of error-containment features included in the bridges to drop untimely or babbling-idiot frames. With these mechanisms, the TSN TG aims at providing Ethernet with increased dependability, more specifically, increased *reliability*.

1.2 Problem statement

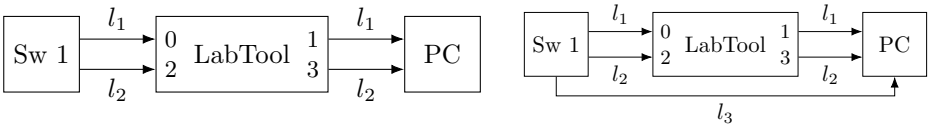
Unfortunately, although TSN standards provide space redundancy to tolerate permanent faults affecting the channel, they do not provide any mechanism specially tailored to tolerate temporary faults. We must note that temporary faults in the links are the most common type of fault that affect distributed systems. The fact that TSN standards do not count with any mechanism specifically devised to deal with this type of faults is an important limitation of its current specification, in terms of reliability. From now on we will use the initialism TSN to refer to TSN standards.

Certainly, space redundancy could be used to tolerate temporary faults, but it is not the most suitable solution. The number of redundant independent paths required to tolerate faults when using only space redundancy increases with the number of simultaneous temporary and permanent faults that want to be tolerated. This has an important impact on the cost of the system,

since each additional redundant path requires additional hardware, i.e. links, bridges and new ports in end-systems; and increases the size, cost and energy consumption of the network. We next discuss this problem with an example.

1.2.1 The drawbacks of only using space redundancy

We illustrate with an example that the number of redundant independent paths must be at least one more than the number of simultaneous faults that we want to tolerate if we use space redundancy only. To that, we have carried out four different experiments. Figure 1.1 shows the topologies used for these experiments.



(a) Topology used to study the number of simultaneous faults that can be tolerated using two disjunctive paths.

(b) Topology used to study the number of simultaneous faults that can be tolerated using three disjunctive paths.

Figure 1.1: Topologies used to study the number of simultaneous permanent and temporary faults that can be tolerated using spatial redundancy solely.

Specifically, we use a real device that implements some of the most relevant TSN standards, namely global clock synchronisation, traffic shaping of scheduled traffic and online management of the network. This device is indicated in Figure 1.1 as Sw 1 and it acts as an end-system that generates traffic and transmits all this traffic in parallel through several interfaces. The fault injection device (indicated in Figure 1.1 as LabTool) is used to inject faults in the links as needed in each experiment. Finally, the PC executes the network analyzer tool Wireshark to capture the traffic received through the different links. Figure 1.1a shows the topology used for experiments I, II and III with two redundant paths; whereas Figure 1.1b shows the topology used for experiment IV, with three redundant paths. Regarding the traffic configuration, frames transmitted are configured to have the same priority to ensure that they traverse the exact same path and their frame size is 787 bytes, i.e. 754 bytes of payload plus control information. Furthermore, we transmit the same 1000 frames in parallel through each link in each experiment.

Table 1.1 shows the number of frames received through each interface of the PC and the number of frames lost in each experiment, i.e. the number of errors that are not tolerated. Experiment I shows the behaviour of the network in the absence of faults. In this experiment the end-system sends the same 1000 frames through both links and the fault injection device simply forwards the frames with no modifications. We see that the PC receives 1000 frames through each interface, proving the correct operation of the system.

In Experiment II, the end-system sends 1000 frames through both links and the fault injection device provokes one error in one of every 100 frames in link 1 and forwards the frames in link 2 with no modifications. This configuration allows us to study the behaviour in the presence of temporary faults in link 1. We can see that the PC receives 990 frames through the first interface and 1000 frames through the second one. Therefore, the temporary faults in link 1 are tolerated using link 2.

In Experiment III, the end-system sends 1000 frames through both links and the fault injection device provokes one error in one of every 100 frames in link one, just like in Experiment II. Nevertheless, in this case the fault injection device provokes errors in all the frames transmitted through link 2, emulating a permanent fault in the link. In this case, we see that the PC only receives 990 frames through the first interface and none through the second one. Therefore, all the frames affected by a temporary fault in the first link are lost, experimentally proving the rather obvious fact that not all combinations of one permanent fault and a temporary one can be tolerated using only two redundant paths.

Finally, Experiment IV uses the same configuration as Experiment III, but with an additional link connecting the end-system to the PC. The third link is fault-free and, therefore, the PC receives 990 frames through the first interface, 0 through the second one and 1000 through the third one, tolerating the simultaneous permanent and temporary faults.

We can see that when considering concurrent permanent and temporary faults, even if the number of temporary faults is low, we need to add additional disjunctive paths if we want to guarantee the correct reception of all frames using only space redundancy. Nevertheless, the use of space redundancy implies a significant increase in the size, cost and energy consumption of the system. Obviously, space redundancy is required to tolerate permanent faults, but *time redundancy* is more adequate to tolerate temporary faults. Time redundancy consists in carrying out the same action several times using the same hardware (Johnson, 1988), e.g. transmitting the same frame several

Table 1.1: Number of frames received through each port in the absence of faults, presence of temporary faults only and presence of temporary and permanent faults.

Experiment	**# received frames**			**# lost frames**
	Interface 1	Interface 2	Interface 3	
Exp I	1000	1000	—	0
Exp II	990	1000	—	0
Exp III	990	0	—	10
Exp IV	990	0	1000	0

times through the same link.

One of the main advantages of time redundancy over spatial redundancy is that the former does not require a significant increase in the hardware. Moreover, when used together, time redundancy allows taking full advantage of space redundancy to tolerate permanent faults, since space redundancy is not wasted tolerating temporary ones. Finally, once the system has suffered the permanent fault of all the redundant paths that connect any two nodes except one, time redundancy allows tolerating temporary faults, which are the most frequent ones ("The Impact of Bit Error Rate on LAN Throughput White Paper") (*Error Rates and Testability.*) (*BER Requirements.*).

1.2.2 The use of time redundancy to tolerate temporary faults

Traditionally, temporary faults in the communication channel have been addressed by means of time redundancy (Avizienis, 1976), i.e. by means of retransmissions. In the particular case of data networks, retransmissions are normally based on *automatic repeat request* (ARQ) techniques (Shu Lin, Costello, and Miller, 1984), which rely on *acknowledgement* (ACK)/*negative ACK* (NACK) messages and/or timeouts to trigger the retransmission of lost frames. This simply means that the receiver of a frame is expected to send an ACK indication when correctly receiving a frame or a NACK indication when not receiving or incorrectly receiving one. With this indications the transmitter knows if it has to trigger the retransmission of the original frame or such redundancy is not needed. However ARQ techniques are not well-suited for RT highly reliable industrial networks for the reasons that follow:

(i) Due to the random nature of temporary faults, ARQ solutions are non-deterministic in terms of end-to-end delay and bandwidth consumption.

(ii) The end-to-end delay of a frame significantly increases when, and only when, retransmissions are required, which leads to a high jitter in the communications.

(iii) ARQ solutions introduce additional scenarios involving faults, which are harder to tolerate (Ha, Nguyen, and Tsuru, 2019) as temporary faults can also affect ACK/NACKs messages.

(iv) When the network conveys scheduled traffic, the schedule must cope with the worst-case scenario involving temporary faults. Since in such worst-case scenario, multiple ACK/NACKs have to be scheduled in addition to multiple frame retransmissions, ARQ solutions further worsen the utilization efficiency of the bandwidth.

Since Ethernet has been extensively used in data communication networks, it has commonly relied in higher-layer ARQ-based protocols to tolerate temporary faults in the links. Nonetheless, for the reasons previously discussed, it is necessary to propose a suitable solution to deal with temporary faults in the links of reliable RT TSN networks.

1.3 Thesis statement

The aim of this study is to prove the following thesis (our thesis):

> *We can increase the reliability of multi-hop networks based on TSN standards, which support real-time and operational flexibility, by using proactive time redundancy to tolerate temporary faults in the links in a way that is suitable for the RT response of the network.*

In order to increase the reliability of TSN networks we have proposed a time redundancy mechanism called Proactive Transmission of Replicated Frames (PTRF). PTRF is a proactive time redundancy mechanism which consists in transmitting several copies of each frame in a preventive manner to increase the probability that at least one copy of each frame reaches its destination even in the presence of temporary faults.

Note that unlike space redundancy, PTRF does not require the addition of extra links and bridges, which can help reduce the cost, size, energy consumption and complexity of the network. Moreover, PTRF is better suited than ARQ-based solutions for RT networks for the following reasons:

(i) PTRF is deterministic in terms of time and bandwidth consumption, which is of paramount importance for RT traffic.

(ii) PTRF introduces less jitter, as the time elapsed between frame replicas is as short as the interframe gap (a detailed discussion can be found in Chapter 8).

(iii) In PTRF the fault scenarios are simpler than in ARQ-based solutions since PTRF does not rely on extra ACK/NACK messages that could be affected by faults.

(iv) When the network conveys scheduled traffic, the schedule must cope with the worst-case scenario involving temporary faults, i.e. the schedule must take all the retransmissions of the frame and the transmission of ACK/NACKs into account. Since in such worst-case scenario, multiple frames and ACK/NACKs have to be scheduled, ARQ solutions further worsen the utilization efficiency of the bandwidth..

As discussed, PTRF would be a suitable mechanism for tolerating temporary faults in the links of RT TSN networks, but fault tolerance mechanisms always increase the cost of the final systems and thus should be able to clearly increase the "quality" of the system as a whole, which translates as increasing the dependability of it. In the case of PTRF and due to the fact that it should be able to increase the chances of providing an uninterrupted correct service, the reliability of the networks should be increased.

1.4 Contributions of this dissertation

Naturally, the main contribution presented in this dissertation is the proof of the thesis statement. That is, we have proven that we can increase the reliability of multi-hop networks based on TSN standards, which support real-time and operational flexibility, by using proactive time redundancy to tolerate temporary faults in the links in a way that is suitable for the RT response of the network. Nonetheless, in order to achieve this goal we have taken several steps that resulted in other contributions. This section

provides an overview of these contributions, which are described thoroughly in the rest of this dissertation.

Even though there are many voices claiming that TSN is the most adequate network technology to support these novel networks (multi-hop networks with real-time and operational flexibility), part of this work consists in assessing its adequacy and studying other existing Ethernet-based solutions. More concretely, we focus on industrial Ethernet-based protocols, as they provide Ethernet with some RT capabilities, network flexibility and *fault tolerance* (FT) mechanisms. We carry out a thorough study of the existing Ethernet-based industrial solutions and we classify them following different criteria. First, we group them depending on whether they provide FT only, FT and RT or FT, RT and flexibility. Furthermore, we also classify the protocols in terms of the dependability attribute they improve: availability, safety and reliability. Finally, we classify the protocols in terms of the fault tolerance aspects and the network features they provide.

This study represents a relevant contribution to the area as it describes each protocol, it provides three different classifications and identifies the need to develop a communication infrastructure capable of providing high reliability to RT, multi-hop, and flexible systems.

The rest of our work focuses on increasing the reliability of networks based on TSN. Specifically, we want to tolerate temporary faults in the links by means of proactive time redundancy, i.e. unconditionally transmitting several copies of each frame to increase the probability of at least one copy reaching its intended destination correctly. We design the Proactive Transmission of Replicated Frames mechanism in such a way that it is specially tailored to the specificities of TSN. Concretely, our design ensures the following characteristics:

- PTRF can be used to replicate all the types of traffic conveyed in a TSN network.
- PTRF allows parameterising the number of replicas to be transmitted for each traffic type.
- PTRF can be used with the RT mechanisms that are already present in TSN, as well as with the mechanisms designed to schedule TSN traffic. Therefore, even though PTRF does not enforce RT response, a TSN network that implements PTRF can provide RT guarantees to the traffic it conveys.

- PTRF has been designed taking into consideration the FT mechanisms present in TSN and it will be possible to integrate it with such mechanisms in the future.

We propose three different approaches of the mechanism. The approaches differ from each other in two aspects, (i) which are the devices that carry out the replication of frames and the elimination of surplus replicas; and (ii) which is the technique used to calculate the number of replicas to be transmitted. Note that each one of these approaches represents a contribution in itself.

We have also carried out a series of experiments using a real implementation of TSN and PTRF in order to measure the impact that PTRF has on the end-to-end delay, the jitter and the bandwidth consumption. Specifically, we carry out a sensitivity analysis that allows us to study the behaviour of PTRF under different operating conditions. Through this analysis we study the impact that PTRF has on the efficiency of the network and we can conclude that it is an adequate solution to provide time redundancy in real-time TSN networks; helping us fulfil the last part of our thesis ([...] in a way that is suitable for the RT response of the network.)

In order to prove our thesis (We can *increase the reliability* [...]) we develop a series of parameterised models based on the PRISM probabilistic model checker, which allow quantifying the reliability benefits that TSN networks can obtain thanks to several fault-tolerance mechanisms proposed in TSN standards, as well as to our PTRF mechanism. More precisely, we develop a model for a standard TSN network, as well as two models for the PTRF approaches. These models represent a relevant contribution as they constitute the first tool that has been proposed so far to quantify the reliability of TSN networks.

On top of that, we have carried out an evaluation of the reliability achievable by TSN networks when using PTRF to tolerate temporary faults. Concretely, we carry out a parametric sensitivity analysis that allows us to quantify how several dependability-related aspects of PTRF affect the reliability considering only temporary faults. Despite the interest in providing TSN networks with fault-tolerance mechanisms, to the best of our knowledge, there are no previous works that quantify the reliability that these mechanisms can yield. In fact, most of the works that treat space redundancy in TSN focus on evaluating the impact on the performance, bandwidth or schedulability of the network (Nasrallah et al., 2019). Furthermore, even the works that do focus on dependability aspects do not provide a quantitative measure for

the reliability. Instead, they analyse it in a qualitative manner or focus on specific gains in the fault tolerance capabilities, e.g. counting the number of simultaneous faults that can be tolerated.

Finally, even though PTRF was initially conceived to tolerate temporary faults in the links only, its design allows tolerating certain temporary faults that may affect the communication controllers in the nodes and bridges. For instance, a temporary fault that could cause a frame not to be properly buffered in the output queue of the device can be tolerated as long as one of the copies is properly buffered.

1.5 What this dissertation does not address

During the development of this work we have encountered many interesting problems that could be addressed. Nonetheless, to conclude our work and prove our thesis, we focus on a single problem. For the sake of clarity, and in order to further define the focus of our work, we next want to make explicit some related aspects that we do not cover in this dissertation.

- This dissertation does not aim at providing a solution for tolerating permanent faults, TSN already does. We only address non-malicious temporary hardware faults that affect the links of the network.
- We do not develop a prototype that implements PTRF together with TSN's space redundancy. As we mention in Section 1.4, we have designed PTRF so that it can be integrated with TSN's FT mechanisms in the future, but we have not carried out a verification of the integration.
- This dissertation does not aim at designing a complete network architecture nor assess the reliability of complete TSN networks.
- We do not aim at tolerating temporary faults in the software of the nodes, only in the links of the communication network.
- We only consider TSN networks that use global clock synchronisation to provide RT guarantees to the communications.
- We do not ensure RT response nor we study the impact that PTRF has on schedulability. Nonetheless, we design PTRF so it can be used with TSN's RT mechanisms.

- We do not increase the flexibility of TSN networks, we only take advantage of the flexibility they already provide.

- All the work presented in this dissertation is application-independent. Therefore, the reliability analyses here presented consider a generic communication subsystem and do not take into consideration the application nor the whole distributed system.

1.6 Overview of the dissertation

The remainder of this dissertation is organised as follows. Chapter 2 covers the dependability concepts that are relevant for this dissertation, while Chapter 3 covers the real-time concepts that are necessary to properly understand the rest of this dissertation. Even though TSN fulfils the requirements reflectedd in our thesis statement (i.e. support for multi-hop topologies, real-time response and operational flexibility), Chapter 4 surveys the existing industrial Ethernet protocols that provide FT, RT or flexibility to the communications in order to assess that TSN is the only current technology that presents all the characteristics which are assumed to be required by the novel applications mentioned at the beginning of this chapter.

In Chapter 5 we provide an overview of the operation of a TSN network that implements the standards that we use in this dissertation. We describe the main characteristics of such a network, the TSN mechanisms that we consider as a starting point in this dissertation and how they are integrated. We also provide an overview of other TSN standards that could be used together with the ones we consider for this dissertation.

Chapters 6 to 9 are devoted to the specific parts of this dissertation that allow us to prove our thesis statement. More specifically, Chapter 6 thoroughly describes PTRF, its design rationale, the fault types we have considered, the operation of the three approaches of PTRF and their design. Chapter 7 presents the validation and feasibility analysis of PTRF, based on the simulation models we have created for the three proposed approaches and on an analysis of the fault scenarios that each approach can tolerate.

Chapter 8 covers the implementation of PTRF in a real TSN prototype as well as the experimental evaluation of standard TSN and two of the PTRF approaches. Specifically, we carry out a quantitative evaluation from three perspectives: the end-to-end delay, the jitter and the bandwidth consumption. Furthermore, we use fault injection to evaluate PTRF in the presence of

temporary faults.

Chapter 9 presents the models used to carry out the reliability analysis, as well as the parameterised sensitivity analysis of the reliability achievable by standard TSN and PTRF. We start describing the modeling rationale, which includes the reliability metric, the fault types that are modelled, the modeling assumptions and the modeling strategy. We then move on to the sensitivity analysis where we vary different dependability related aspects and we study their impact on the reliability.

Finally, Chapter 10 concludes this dissertation by summarising the most important aspects discussed and proposing several lines of work for the future.

Chapter 2

Basic Concepts on Dependability

Dependability is the discipline responsible for providing systems with means so that they can be trusted to deliver a correct service with a high probability. In fact, dependability is a wide discipline which encompasses reliability, availability, safety, security, performability, maintainability and testability.

Because of its extension, we can not cover all the dependability aspects of a system. Instead, in this study we focus on a subset of them. Figure 2.1 shows the relationship among the dependability concepts considered in this dissertation. Furthermore, it classifies them into three groups: attributes, impairments and means.

Furthermore, from the three attributes presented in the figure, in this dissertation we focus on reliability, even though we have considered availability and safety while studying the related work.

A system is highly reliable if it has a high probability of performing its function *correctly in a continuous manner*, without failing due to faults. A *fault* is a defect in the behavior of a system or in the way the system is designed or built. A fault may cause an *error* (or errors), which is an incorrect result delivered by the system. Finally, an error can lead to the *failure* of the system, which is a deviation from the correct service the system intends to deliver.

There are two main ways to procure reliability, namely *fault prevention*, i.e. to prevent the occurrence of faults, for instance, investing in the components quality; and *fault tolerance* (FT), i.e. to include mechanisms for the system

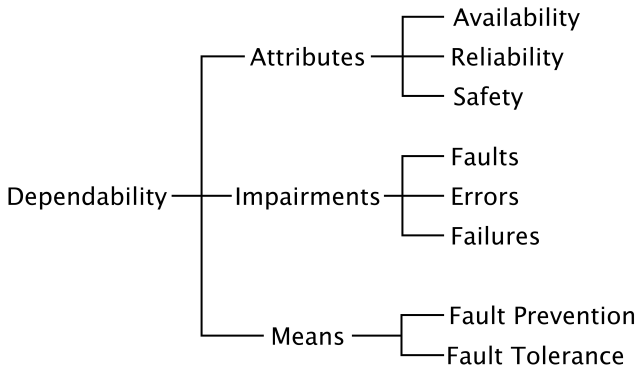


Figure 2.1: Subset of the dependability tree showing the concepts considered in this paper and how they are classified. Figure based on one appearing in (Laprie, 1992).

to provide its service even when it suffers from faults. Even when fault prevention is used, in complex systems faults eventually happen; thus, it is fundamental to provide them with adequate FT mechanisms if high reliability is required.

We next describe the dependability concepts depicted in Figure 2.1 related to the work presented in this dissertation.

2.1 Fault model and failure modes

In order to correctly design an FT system it is important to take into account the *fault model*, i.e., the types and number of faults the system has to deal with. Furthermore, it is also important to identify the *failure mode*, i.e., the deviation in the service provided by the system (or in the service provided by a subsystem within the system) which is caused by the faults.

An exhaustive classification of faults can be found in (Laprie, 1992). Roughly speaking, we can differentiate between (i) hardware vs software, (ii) temporary vs permanent and (iii) simultaneous vs sequential.

On the other hand, failures can be classified following different criteria. For instance, they can be hierarchically classified according to their degree of restriction, as proposed in (Poledna, 1996). We only refer here to the failure modes considered in this dissertation:

- Byzantine: lack of restrictions on the way the system can behave. It includes two-faced behaviours, i.e., a faulty device sending different information to different devices; and impersonations, i.e. a faulty device pretending to be a different device.

- Incorrect computation: the system delivers incorrect results, either in the value or the time domain.

- Performance: the system delivers correct result in the value domain, but fails to do it in the time domain.

- Crash: the system omits the delivery of results from the moment the failure happens on.

Figure 2.2 depicts an inclusion hierarchy of the described failure modes, where the outer failure modes are the least restricted and harder to deal with, while inner failure modes are more benevolent. In this sense, byzantine devices exhibit the most complex behaviours, which include two-faced behaviours or impersonations; while fail-stop devices exhibit the easiest behaviour to deal with as the device simply stops producing results and delivers a constant value. Furthermore, since failure modes are classified using an inclusion hierarchy, a failure mode includes all the failure modes below it, e.g., a device that exhibits timing failure semantics also includes omission, crash and fail-stop failures.

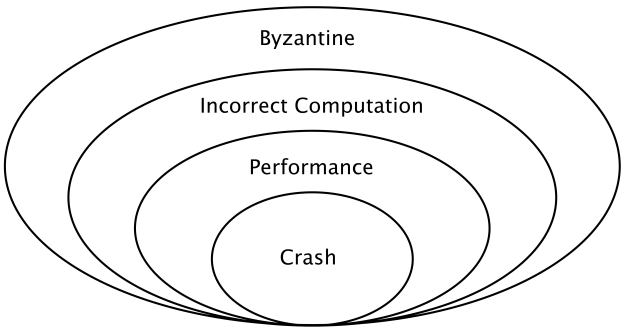


Figure 2.2: Inclusion hierarchy diagram that depicts the presented failure modes. The inner failure modes are more restricted and benevolent; whereas outer failure modes are less restricted and harder to deal with.

2.2 Achieving fault tolerance

Figure 2.3 shows a classification of the different means to achieve Fault Tolerance and how they are related. FT is carried out by means of *error processing* and *fault treatment* (Anderson and Lee, 1981). Error processing, consists in eliminating errors from the state before they cause the failure of the system. In contrast, fault treatment consists in preventing faults from causing errors again.

Moreover, there are two different techniques for error processing. On the one hand, *error recovery* consists in detecting the error and replacing the erroneous state by an error-free state. If the error-free state is a past state, it is called *backward recovery*; and if the error-free state is a new state, it is called *forward recovery*. On the other hand, *error compensation* consists in designing the system with enough redundancy to produce correct results even in the presence of faults.

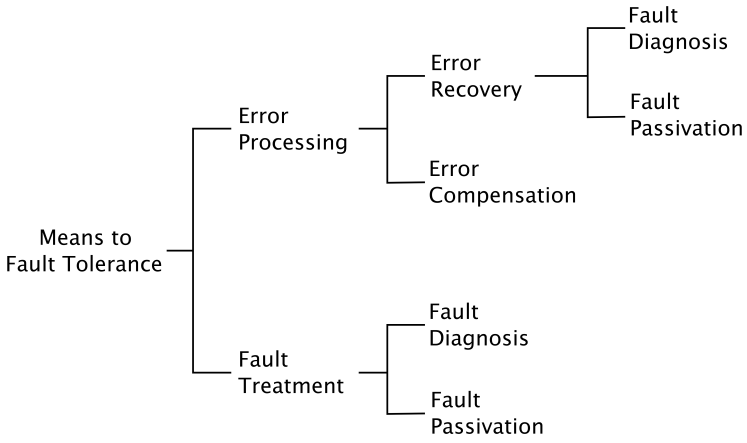


Figure 2.3: Tree that classifies the different techniques used to achieve fault tolerance.

In a distributed systems, fault tolerance can be achieved at different levels of the system's architecture and using different types of redundancy. Specifically, we distinguish four different types of redundancy (Johnson, 1988):

- *Hardware redundancy* (also known as *space redundancy*) consists in providing the system with more hardware components than would be needed if faults could not occur. Examples of hardware redundancy are the use of redundant (multiple) sensors, nodes, switches or

communication links.

- *Software redundancy* consists in providing the system with additional software beyond what is needed to carry out the operation of the system. An example of software redundancy is the use of redundant tasks to carry out the same action.

- *Time redundancy* consists in carrying out the same action several times with the same hardware and software. An example of time redundancy is message retransmission.

- *Information redundancy* consists in introducing more information than needed if faults could not happen.

A specific type of redundancy is *replication*, in which the system is provided with several identical copies of software and/or hardware components. With replication a fault affecting one of the replicas can be tolerated thanks to the non-faulty ones that take over the operation. There are three main types of replication (Poledna, 1996).

- With *active replication* all the replicas perform the same operation in parallel, i.e., all the replicas provide the same service and, in absence of faults, the result can be obtained from any of them. In order to tolerate potential faults, the result is actually obtained by voting on the values proposed by each replica. This corresponds to the concept of error compensation described above.

- With *semi-active replication* one of the replicas is in charge of the non-deterministic decisions. This replica is commonly referred to as central or *leader*. The leader must inform the rest of the replicas about the results of the decisions. The rest of the replicas are commonly referred to as *followers*. In the case of deterministic decisions, the results can be obtained just from the leader or from all the replicas; like in active replication. If the leader fails, its failure is detected and one of the followers becomes the leader.

- With *passive replication* only one of the replicas, the primary replica, performs the operation while the rest, the standby replicas, remain inactive. If the primary replica fails, its failure is detected and one of the backup replicas takes over the operation of the primary replica, typically by starting in a previously stored (using checkpointing) state

of the faulty primary replica. This corresponds to the concept of backward error recovery presented above.

Note, however, that redundancy is not only used to achieve fault tolerance. An example is the use of *cyclic redundancy check* (CRC) in Ethernet frames. CRC is a way of information redundancy used to perform error detection. Another example is the use of software redundancy to carry out specific checks to assess the correctness of the results provided by a task.

In distributed systems with stringent RT requirements, error compensation represents the most suitable solution to increase the reliability. This is so because error compensation allows tolerating faults without introducing a recovery (failover) time, which could prevent the system from meeting its deadlines. In this case it is said that the system *seamlessly tolerates* faults or that the system has *seamless redundancy*. Furthermore, when error compensation is used, the system does not need to be aware of the existence of faults to produce a correct result. This is known as *fault masking*.

It is also noteworthy that replication is specially suitable to eliminate *single point of failure* (SPoF). A SPoF is any single component within a system whose failure could cause the failure of the whole system, even if that component is not directly responsible for calculating application results. Thus, eliminating any SPoF is essential to increase the reliability of any system or subsystem.

However, to properly implement replication, it is fundamental to address two issues. On the one hand, faults affecting different replicas must be independent of each other, i.e., when a fault affects a replica that same fault does not affect any other. For instance, it would be desirable that replicas are powered by different power supplies, so that a failure in one power supply only affects the replica it is connected to.

On the other hand, faults affecting a replica must not propagate to other correct subsystems (Kopetz, 2011b). Note that this is important since otherwise a single fault can affect the whole system. Nevertheless, this can be solved by means of *error containment*. Specifically, it is necessary to define one *error-containment region* for each replica, i.e., regions from which errors cannot propagate to other parts of the system.

Finally, note that error containment can be used to restrict the potential failure modes of a subsystem, i.e., restrict its *failure semantics*. Restrict the failure semantics of the subsystems allows significantly simplifying the FT mechanisms of the system, as dealing with byzantine subsystems is much

harder than dealing with crashed ones.

2.3 Tolerating permanent faults

Permanent faults can be tolerated by means of replication based on space redundancy. Moreover, in distributed systems with hard RT requirements, faults have to be seamlessly tolerated. In these cases, active replication is typically used.

To ensure the correct operation of the system when using active replication, all non-faulty replicas must produce consistent (equivalent and often identical) results. This is known as *replica determinism* (Poledna, 1996) and we distinguish two levels:

- *Internal replica determinism* in which replicas produce corresponding results, as long as they start in the same initial state and they are provided with identical inputs. This determinism is basically achieved avoiding the use of non-deterministic program constructs, e.g. random numbers, or any other technique that could cause non-deterministic decisions, such as relying on information that is only known to the local replica.
- *External replica determinism* in which replicas are provided with corresponding inputs to carry out their operations. This can be achieved by forcing replicas to agree on a state or set of values, e.g. by forcing replicas to exchange and vote on their state/values to reach a consensus.

An important feature related to agreement is *consistency*. This feature is desirable, if not fundamental, for any distributed system (even if it is not fault-tolerant), since a distributed system consists of different subsystems (replicated or not) having to have a common view of a given piece of information, e.g. the set of messages exchanged so far. To achieve consistency in a highly-reliable way, it is possible to use similar techniques as those to provide replica determinism.

2.4 Tolerating temporary faults

Time redundancy is specially suitable for tolerating temporary faults; as it is more cost-effective and simpler than using space redundancy. Time redun-

dancy is a common technique used in the communication subsystem of DESs. The most widespread techniques are ARQ and proactive retransmissions.

ARQ solutions rely on the transmission of ACK and NACK messages to trigger the retransmission of frames when these are lost. This technique is not the most suitable to tolerate temporary faults of RT systems, as the bandwidth and time required to complete the transmission are non-deterministic. Moreover, the jitter introduced by these solutions is high and the worst-case response time has to include the transmission of the additional ACK and NACK messages, and the timeouts to detect the omissions of frames.

Proactive retransmissions consist in transmitting several copies of each frame in a preventive manner, to ensure that at least one copy reaches the destination in the presence of faults. This is a more suitable solution for RT systems as it is deterministic in the resource consumption, introduces less jitter and requires less time in the worst case.

Since tolerating temporary faults in the links of communication networks is the core of this study, we find a thorough discussion on time redundancy throughout this dissertation.

Chapter 3

Basic Concepts on Real-Time Communications

As introduced previously, a system provides a real-time service if it does so timely, i.e. before a specific deadline expires (Burns and Wellings, 2009). Figure 3.1 depicts the main aspects of real-time systems that will be described in this section. Generally speaking, a system handles events by means of one or more tasks, thereby providing a set of services. An event can be internal, e.g. the need of taking a sample; or external, e.g. a sensor value reaching a specific threshold.

Furthermore, an event can be periodic, aperiodic or sporadic (aperiodic with a minimum inter-arrival time). In distributed systems, the nature of events also dictates the nature of the messages they produce.

Depending on the strategy a system uses to handle its events, it can be classified as either (i) *time-driven*, when it executes its functions following a given internal time basis; or as (ii) *event-driven*, when it executes them on demand when the corresponding events happen.

Each one of these strategies has its pros and cons. For instance, a time-driven system is naturally well suited to handle periodic events, but it may not be as reactive as an event-driven one to handle aperiodic/sporadic ones. Conversely, although more reactive, an event-driven system may be vulnerable to event showers, which may exhaust the system resources and prevent it from meeting its deadlines.

With respect to the consequences of missing a deadline, a system can be roughly classified as follows:

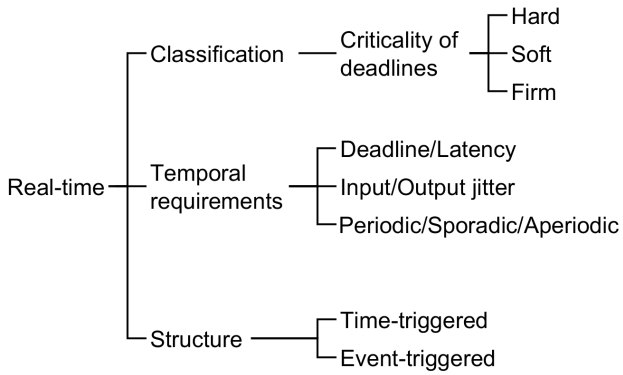


Figure 3.1: Aspects of the real-time systems. Figure based on (Burns and Wellings, 2009).

(i) *Hard real-time*, when missing a deadline leads the system to fail in a catastrophic manner.

(ii) *Firm real-time*, when the system has a rigid deadline, i.e., missing a deadline invalidates the results but the consequences are not catastrophic.

(iii) *Soft real-time*, when missing a deadline is acceptable as it merely degrades —even temporarily— the service.

In order to fulfill the real-time requirements of distributed system, the transmission of messages have to be appropriately scheduled so as for messages to meet their own deadlines. To achieve this, it is fundamental that the maximum end-to-end delay of messages are both deterministic and bounded. Moreover, many RT systems require messages to exhibit a low *jitter*; where jitter can be understood as the variability with which a message is transmitted or received.

There are two main types of real-time schedulers, namely *cyclic executive schedulers* and *priority-based schedulers*. Basically, a cyclic executive scheduler calculates beforehand the instants of time at which each message is going to be transmitted so that all messages meet their deadlines; whereas a priority-based scheduler assigns priorities to messages so that the order in which they access the resources in case of conflict, allows them to meet their deadlines.

In the context of communication networks, a cyclic executive scheduler has traditionally been used in what are called *time-triggered* networks (Poledna,

1996). In this kind of networks the application or the network trigger the transmission of messages at the predefined instants of time. The traffic these messages constitute is commonly referred to as *time-triggered* (TT) *traffic*, and it is specially adequate for supporting time-driven systems. More specifically, these networks divide the communication in rounds generally called *communication cycles* that are periodically triggered to allocate the different messages.

Conversely, a priority-based scheduler has traditionally been used in what are called *event-triggered* (ET) networks, in which the application triggers on demand the transmission of the messages. The traffic in this case is known as *event-triggered traffic*, and it is specially well suited for event-driven systems.

Both TT and ET real-time networks must rely on a deterministic *medium access control* (MAC) mechanisms, i.e. a MAC mechanism that ensures that each message has a deterministic and bounded maximum end-to-end delay. In this sense, these networks must rely either on a static or on a contention-free dynamic MAC mechanism. The most natural choice for a TT network is to use a static MAC mechanism such as *time-division multiple-access* (TDMA); but a contention-free dynamic one, e.g. polling-based, can also be used. In the case of an ET network, the preference is to use a contention-free dynamic MAC protocol, e.g. based on polling, token passing, etc.

TT and ET networks have pros and cons that are more or less analogous to the ones of time- and event-driven systems respectively. In this sense TT networks are preferred in critical real-time applications, because all the communication resources are reserved in advance. This means that, the messages that are transmitted at every instant are known, which makes it easier to proof their determinism. However, when compared to ET networks, TT ones present limitations to efficiently convey traffic that is inherently event-triggered, e.g. alarms.

Currently several real-time communication networks combine and extend the mechanisms of traditional TT and ET networks to adequately support both types of traffic. The capacity of doing so is known as *real-time flexibility*. A usual strategy to provide this capacity consists in dividing each communication cycle into *communication windows* to accommodate different types of traffic.

Moreover, on the quest for supporting adaptivity, a number of newer communication networks also provide what is known as *operational flexibility*. This is the ability to change the RT requirements and the schedule of the traffic online, to provide an adequate communication as the operational

requirements of (and the services executed by) the system change.

Another important aspect of RT systems is that they usually require time synchronization mechanisms, so that tasks (in the same or in different nodes) can adequately coordinate among them, and messages are timely transmitted and forwarded by the different network devices. In some systems, each node and network device has its own local timer and are not tightly synchronized. When so, the nodes and network devices implement some mechanisms to bound bursts of activity so that they can share the resources and operate in a real-time manner.

However, most systems that are time-driven or that rely on a TT network, require a tight time synchronization. Moreover, a tight time synchronization simplifies the real-time and the fault-tolerance mechanisms, e.g. the forwarding of messages at specific instants of time, the detection of errors like message omissions, or the management of the consistency among devices that perform the same tasks. This is so because this kind of synchronization allows having more knowledge about the current and future actions that should occur in the different parts of the system, e.g. the simultaneous reception of two copies of the same message through two different links.

There are two main strategies to provide tight time synchronization. One of them consists in having a global (system-wide) notion of time. This strategy is commonly known as global *clock synchronization* and it can be implemented in several ways. For instance, some systems include a privileged device or node that provides a master clock, which then is disseminated among the clocks of the rest of devices and nodes. In other systems devices and nodes synchronize their own local clocks by exchanging information about their visions of the current time and then implementing a convergence function. The other main strategy to achieve a tight time synchronization consists in having a privileged device or node that polls the actions and the communications in the rest of the system according to its own clock.

As we anticipated in the Introduction, there is great interest in using Ethernet for industrial applications. Nevertheless, Ethernet was not designed to provide RT guarantees and, thus, it provides no mechanism for real-time communication (Wilamowski and Irwin, 2011). Shared Ethernet relies on a contention-based (collision-based) MAC mechanism called CSMA/CD, which cannot guarantee access to the communication medium in a deterministic and bounded time, and which cannot bound the jitter either. Moreover, full-duplex Ethernet switches no longer rely on CSMA/CD, and prevent contentions (collisions) by using a dedicated uplink and a dedicated downlink

per node. Nevertheless, switches may also introduce non-deterministic and unbounded forwarding delays and jitter. This is due to frames being buffered upon reception in the switch and then queued before retransmission. Also, for a long time Ethernet did not provide any sort of time synchronization mechanisms (except the ones needed to synchronize the receiver's communication controller with the beginning of each frame to not misinterpret its bits).

Fortunately, the TSN TG has added several mechanisms to provide RT guarantees to Ethernet. Some of these mechanisms include MACs based on TDMA; full-duplex switches with adequate queuing policies; *traffic shaping*, in which the network devices delay the transmission/forwarding of certain types of traffic during a limited amount of time to provide bounded latency to other kinds of traffic; *resource reservation*, which consists in checking that there are enough communication resources for the traffic; *traffic policing*, which consists in monitoring the traffic to ensure that it meets a set of requirements previously specified and global clock synchronization.

Chapter 4

Study of Existing Ethernet-based Solutions

We have studied a great variety of protocols to find the one to be used in this dissertation. We must recall that we want to find the most adequate network technology to build the communication subsystem of novel applications that are highly reliable, real-time and flexible and that are based on Ethernet. Since there are countless protocols that aim at providing Ethernet with these characteristics, we have narrowed our search by starting from protocols that already count with at least one fault tolerance mechanism.

Networks can take advantage of almost any FT technique. From error recovery to error compensation and from fault diagnosis to fault passivation. Moreover, these techniques can be combined to achieve highly-reliable communication services. Common implementations of these techniques are recovery of network devices, such as switches; use of alternative paths, when the main path suffers a fault; retransmission of frames; and error containment to prevent a faulty component from jeopardizing the communication among fault-free ones.

Nevertheless, not all the aforementioned techniques and mechanisms are suitable for highly-reliable real-time distributed systems. This is so because for a system to be reliable it has to provide a correct service *continuously*. Thus, mechanisms that introduce failover times that prevent the system from meeting the deadlines are not suitable for said systems.

On top of that, the network architecture is commonly divided in layers following the Open Systems Interconnection (OSI) model (ISO/IEC, 1994).

The OSI model helps enforcing interoperability among different communication systems and protocols. To do so, it divides the different functionalities that a network can offer into layers, where each layer provides a service to the immediately upper layer. The original version of the model defines seven layers, which we briefly introduce from the lowest to the highest.

- Physical (layer 1): responsible for the transmission of raw data bits, includes aspects such as the topology, the transmission medium and the bit-encoding.
- Data Link (layer 2): commonly referred to simply as link-layer. It is responsible for the physical addressing, controlling how components gain access to the medium, frame synchronization and error detection.
- Network (layer 3): responsible for the logical addressing and routing of frames from the source to the destination when these are not directly connected.
- Transport (layer 4): performs the end-to-end communication control, providing a certain quality of service for the communications. Mechanisms such as flow control, packet segmentation/reassembly and error management are performed in this layer.
- Session (layer 5): responsible for controlling remote actions. It establishes, manages and terminates connections between local and remote applications.
- Presentation (layer 6): maps the syntax and semantics of the application data to the ones used by the network.
- Application (layer 7): provides common services to different classes of application to use the communication network.

Although the OSI model presents great advantages, such as modularity and interoperability, it presents some drawbacks too. Specifically, each protocol at each layer adds control information to the frames forwarded down the stack. Thus, this approach causes important computation and communication overheads, which may not be bearable for RT systems. Moreover, the services provided by certain layers may not be needed depending on the network. For instance, many fieldbuses do not need the services of the network layer as they comprise a single network.

For these reasons, RT networks commonly use the OSI collapsed model. This version of the model only includes physical, link and application-layers.

Some functionalities from the network and transport layers (such as routing or error management) are moved to the link-layer; whereas more sophisticated functionalities of the missing layers are moved to the application-layer.

As in this dissertation we focus on RT systems, we have narrowed our search down to only include protocols that could be part of the OSI collapsed model. Note then, that IT protocols such as the Transport Control Protocol (TCP), the Internet Control Message Protocol (ICMP), Virtual Router Redundancy Protocol (VRRP), Transparent Interconnection of Lots of Links (TRILL), link aggregation protocols, Automatic Repeat Request (ARQ) protocols, etc. are out of the scope of this dissertation.

We next describe the protocols studied and we classify them in three groups depending on the network services they provide. First, we survey protocols that represented the first steps towards achieving high reliability in Ethernet-based networks; even though we found out that some of these protocols are not adequate to be used in highly-reliable hard real-time distributed systems, as we explain. Second, we study more complete solutions that provide real-time and reliability, and we classify them according to the degree of fault tolerance they provide. Finally, we present the solutions that provide real-time, reliability and operational flexibility and we discuss their characteristics to help us select the one to be used in this dissertation.

Furthermore, we also group the protocols into four different categories depending on the dependability attribute they improve: availability, safety, reliability and high reliability. A definition of availability, safety and reliability can be found in Chapter 2. Furthermore, we must note that this is a qualitative classification and, thus, we do not use a quantitative metric to differentiate between reliability and high reliability. Instead, we reserve the high reliability group for those protocols that implement mechanisms that, not only improve the reliability of the network itself, but the reliability of the overall system, e.g. protocols that provide error containment to prevent faulty nodes from disrupting the system's operation.

Note that during the description of the protocols we use the terminology chosen by the specific designers of each solution when describing their protocols. Tables 4.1 and 4.2 show the specific terminology and its correspondence with a more general one.

4.1 Protocols that only provide Fault Tolerance to Ethernet

This section comprises different protocols that provide Ethernet with a single fault tolerance service, such as space redundancy. It is important to note that in this section we took into consideration certain protocols that do not meet the stringent requirements of highly critical applications. Nevertheless, these protocols are a key part of Ethernet and can be used to understand the need of further mechanisms and protocols to suit the needs of highly-critical RT distributed systems. Table 4.1 shows the correspondence between the terminology used in each protocol and a more general one.

Table 4.1: Specific terminology used in the fault-tolerance protocols surveyed.

Protocol	**Node**	**Switch**	**Transmitter**	**Receiver**	**Comm. Manager**
STP	End System	Bridge	Transmitter	Receiver	-
SPB	End System	Bridge	Transmitter	Receiver	-
MRP	End Node	Media Redundancy Client	Source	Destination	Media Redundancy Manager
PRP	Node	Switch	Source	Destination	-
HSR	Node	-	Source	Destination	-

4.1.1 Spanning Tree Protocols

The Spanning Tree Protocol (STP) is a link-layer protocol that allows establishing a single loop-free logical topology. STP was standardised as part of Ethernet in the document IEEE Std 802.1D–1998 (802.1, 1998). Nevertheless, STP was superseded by the Rapid Spanning Tree Protocol (RSTP) in the standard IEEE Std 802.1D–2004 (802.1, 2004).

STP was devised to be used on *bridged* (switched) Ethernet networks, with mesh topologies in which components can be connected to each other through redundant physical paths. STP is implemented on the bridges and allows them to monitor the network topology in order to decide on a unique

logical path to connect each pair of *end-systems* (nodes). Specifically, STP can identify redundant links to logically disconnect them from the network and create a logical tree topology.

One bridge in the network is selected to be the root bridge. The root bridge is then used to select which ports are active and which are not in all the bridges of the network. A non-root bridge calculates which is the link with the highest bandwidth that connects it to the root and selects it as preferred link.

Bridges can detect changes in the topology, e.g. the failure of a link or the connection of a new component. Whenever a bridge detects a change in the topology, it notifies the root bridge. Then, the root bridge notifies all bridges in the network for them to discard their current configuration. After that, a new tree is calculated. On the other hand, if the root bridge fails, bridges select as the new root bridge the one with the lowest identifier.

In this way, STP is a recovery protocol that allows providing space redundancy for the network and the root bridge, as long as the network has enough redundant bridges and links. However, STP introduces a high failover time of up to 50 seconds; which improves availability but is unacceptable in highly-reliable systems. To deal with this drawback, the IEEE proposed the Rapid Spanning Tree Protocol (RSTP).

RSTP is an update of the STP that achieves error recovery with shorter failover times. Just like STP, RSTP is a link-layer protocol that allows creating loop-free logical paths between any pair of end-systems in the network. RSTP selects which ports are active and which are not in each bridge, in a similar way STP does. However, in RSTP all bridges exchange information with their neighbors to detect which neighbor has the lowest-cost path to the root.

This allows reconfiguring the network without the root bridge intervention, which significantly reduces the time required for reconfiguration in case of failure. RSTP provides space redundancy with a failover time of up to 10 seconds, which significantly reduces the time required by STP, but it is still unacceptable in highly-reliable systems.

The Multiple Spanning Tree Protocol (MSTP) is an evolution of RSTP that supports the deployment of more than one spanning tree on the same network; thus allowing to use VLANs. MSTP also operates on the link-layer of the network architecture. It is part of the IEEE Std 802.1Q–2005 standard (802.1, 2006). MSTP is compatible with RSTP and STP.

The number of spanning trees created by MSTP will depend on the number of existing regions. Each MSTP tree can have a single or multiple VLANs assigned and each VLAN will use one and only one spanning tree. This allows MSTP to operate in large networks, as the number of trees is not dependent on the number of VLANs, as in other proprietary solutions such as Cisco's Per-VLAN Spanning Tree (PVST) or Rapid Per-VLAN Spanning Tree (RPVST) (*Understanding Rapid PVST+*).

MSTP also supports the creation of new paths whenever a network device fails, as well as the definition of new root device whenever one fails. The failover times are similar to those provided by the RSTP; improving availability but making it not suitable for highly critical systems.

4.1.2 Shortest Path Bridging

Shortest Path Bridging (SPB) is a link-layer protocol devised to enable the creation of loop-free communication paths while supporting multipath routing. It was standardised as part of the IEEE Std 802.1aq and merged in the IEEE 802.1Q–2012 (802.1, 2012).

It has been devised to replace STP, RSTP and MSTP; as it can support larger mesh networks, provides shorter recovery times and allows load share across the multiple active paths. In order to prevent loops SPB relies on a control plane in which each bridge has a global view of the network. This also reduces the time required for reconfiguration in case of failure. All of this is done while guaranteeing that the route follows the shortest path tree and that the forward and reverse paths are symmetric, feature needed by certain synchronization protocols.

To achieve this, SPB creates several logical networks on top of the existing physical network. Frames transmitted by an end-system are only transmitted to the members of the same logical network. To achieve this, each logical network must have a unique identifier. *Edge bridges* (bridges attached to the nodes) are responsible for adding membership information into the frames; which will in turn be extracted from the frame at the receiving edge bridge.

SPB relies on Intermediate System to Intermediate System (IS-IS) to provide bridges with a complete view of the network. IS-IS supports IEEE Std 802.1ag (802.1, 2007) Continuity Check Messages, which monitors, detects and reports failures to IS-IS.

All of these make SPB a standard solution to further support active spatial replication and, therefore, meet the needs of highly-reliable systems.

Nevertheless, SPB in itself cannot be used for RT applications, as it does not provide mechanisms to support RT data communications. As we will explain in the following section, SPB has been extended to provide support for such kind of communications.

4.1.3 Media Redundancy Protocol

The Media Redundancy Protocol (MRP) is a link-layer recovery protocol standardised in the IEC 62439 document, part 2 (IEC, 2010a).

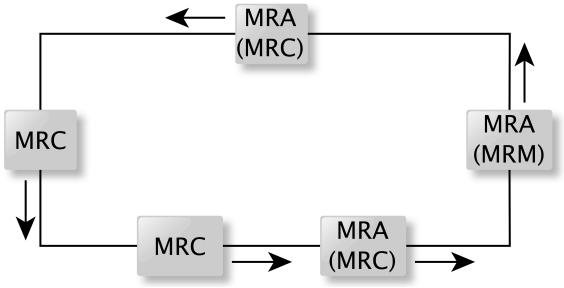


Figure 4.1: Example of 5 nodes interconnected using an MRP ring. The network counts with three MRAs; one acting as the MRM and two acting as MRCs. The other nodes are simple MRCs. The arrows show the direction of frames in a fault-free state.

MRP relies on a ring topology that counts with a central element, called the Media Redundancy Manager (MRM). The MRM is the responsible for the configuration of the network and for the management of network failures. The rest of the nodes in the network are called Media Redundancy Clients (MRCs). MRCs follow the configuration established by the MRM and can detect the failure of an attached link or a neighbor device and notify about it to the MRM.

The revision of the standard in 2016 (IEC, 2016) describes a new element, called Media Redundancy Automanager (MRA). MRAs are components that can act as MRM or MRC. During network start-up, MRAs vote to decide which one will act as MRM. The selected MRA switches to MRM mode; whereas the other MRAs switch to MRC mode. Just one MRM can be active at the same time.

Figure 4.1 shows an MRP network comprised of 5 nodes. Three of these nodes are MRAs, from which one is the MRM and the others act as MRCs. The other two nodes are plain MRC nodes. The arrows represent the

transmission of frames. To avoid loops, MRP turns the ring into a logical line topology. When the ring is fault-free, the MRM blocks a ring-port to not receive or propagate any data frames; whereas all MRCs have both ring-ports active and can receive and forward through both of them. Once a fault occurs, the MRM activates both ring-ports to re-establish the connectivity.

The MRM can detect failures in the network by transmitting a special frame periodically through one of the ports. If the MRM does not receive the frame through the other port it will detect that the ring is open and will activate its blocked port.

MRP considers the presence of more than one MRA in the network, but only one MRA can act as MRM at any time. Whenever the MRM fails, MRCs must detect its failure. If there are other MRAs in the network, they can vote and select a new MRM. Nevertheless, switching to the new MRM introduces a failover time that interrupts the communication until the switch over is completed.

MRP can tolerate one permanent fault in a ring link or MRC with failover times of up to 26.2 ms with 50 nodes. Moreover, MRP does not include any time redundancy mechanisms. This together with the fact that space redundancy is passive, makes MRP vulnerable to temporary faults.

Moreover, MRP does not define the failure semantics assumed for the nodes. In this sense, nodes that exhibit byzantine failure semantics could compromise the correct operation of the system. This is so as nodes are responsible for propagating frames from other nodes and could corrupt their content; e.g. introducing a bit flip that could cause the frame to be dropped by the rest of nodes. To prevent this, the failure semantics of nodes should be restricted using additional mechanisms not devised in the standard.

All of these make MRP an adequate protocol for systems that require high availability, but not for highly-reliable ones.

4.1.4 Parallel Redundancy Protocol

The Parallel Redundancy Protocol (PRP) is a link-layer protocol that provides space redundancy by connecting nodes to two independent and similar networks. The transmission of frames is done through both networks in parallel and the receiver must detect and discard duplicated frames. PRP is standardised in the IEC 62439 document, part 3 (IEC, 2012).

Both networks must use the same link-layer protocol but can differ in

performance and topology, which can be bus, ring or mesh. Figure 4.2 shows 5 nodes connected using PRP; where one of the networks is a bus and the other one is mesh. This can result in different delays and in the reception of out-of-order frames. Moreover, both networks must be isolated from each other and must be fail-independent.

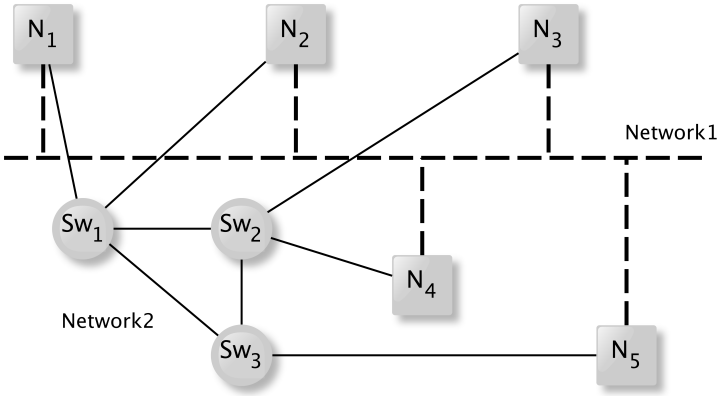


Figure 4.2: Example of 5 nodes interconnected through two networks using PRP. Network 1 has a bus topology; whereas Network 2 has mesh topology.

Nodes are modified to support the parallel connection to both networks; to duplicate frames on transmission; and to detect and eliminate duplicates on reception. Nevertheless, *commercial off the shelf* (COTS) nodes can be attached to a single network or they can be attached to both networks using a switching device called RedBox. Moreover, RedBoxes also allow connecting a duplicated network to a simple network.

The redundant ports of each modified node and RedBox have the same MAC and a single set of IP addresses. This way, frames transmitted through both ports are exact duplicates. Moreover, this also allows each network to operate without having to be aware of the existence of the other network.

PRP includes a mechanism to detect and discard frame duplicates. To do it, PRP relies on sequence numbers. Basically, each frame is assigned a sequence number. Whenever two frames received have the same sequence number and the same MAC address, the receiver knows they are the same frame and discards the duplicate.

This mechanism allows PRP to tolerate permanent faults in the networks seamlessly, i.e. with a failover time equal to 0. The number of failures that can be tolerated in each network will depend on the number of redundant paths. Moreover, this active replication allows tolerating temporary faults in

one network, as long as they are not concurrent to permanent faults in the equivalent devices of the other network.

Moreover, PRP is compatible with protocols for error recovery such as STP or RSTP. Therefore, these protocols can be used when a permanent fault affects a network device of one of the redundant networks to establish new paths and prevent redundancy attrition. Note that even though this recovery takes time, frames will continue to flow through the independent parallel network.

PRP does not describe any mechanisms to restrict the failure semantics of the nodes or switches. Therefore, the system is vulnerable to byzantine behaviours that could cause inconsistencies, e.g. a node could send frames with different content through each redundant port, causing some nodes to receive the right information and others the wrong one.

PRP provides a certain degree of reliability, but it is not suitable for highly-reliable networks. Furthermore, it does not provide mechanisms to support RT data communications and cannot be deployed in RT distributed systems by itself. Nevertheless, note that some of the protocols described in section 4.2 use PRP to provide fault tolerance.

4.1.5 High-Availability Seamless Redundancy

High-Availability Seamless Redundancy (HSR) is a network protocol that, like PRP, aims at providing zero failover time by means of space redundancy. It is a link-layer protocol and it is also standardised in the part 3 of the IEC 62439 (IEC, 2012).

HSR is based on a ring topology, to which nodes are directly attached through two ring ports. Each node acts also as a switch for the traffic received from other nodes. Figure 4.3 shows an HSR network where Node 2 transmits a broadcast message. Just like in PRP, the redundant ports of a node share the same MAC and the same set of IP addresses. This simplifies the identification of frames coming from the same node as duplicates.

To tolerate one permanent fault, nodes send frames through both ring ports at the same time. Multicast frames are read by the destination nodes and forwarded through the other ring port; unicast frames are not forwarded by the destination node.

To prevent frames from indefinitely circulating the network, nodes do not forward frames through a port if they already forwarded said frame through

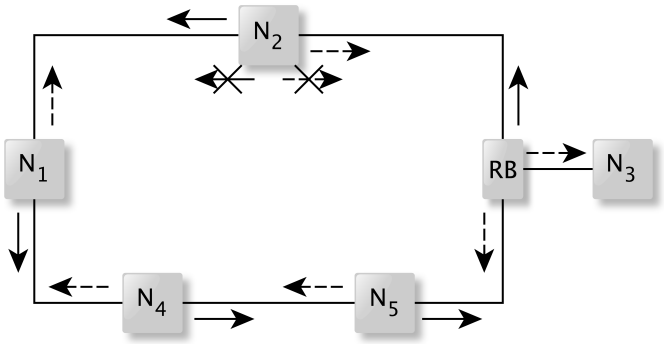


Figure 4.3: Example of 5 nodes interconnected using HSR. Nodes $N_{1,2,4,5}$ have two ports through which they connect to the ring; whereas Node N_3 is attached through a RedBox. The arrows represent the transmission of a broadcast message by Node N_2 through both interfaces in parallel.

said port. To preserve the ring topology, COTS nodes can only be connected to the ring through a RedBox that acts as proxy for the node and as switching device for the messages traversing the ring.

HSR can also support the connection of multiple rings. Two rings can be connected through a QuadBox, device used to forward frames from one ring to the other. Even though one QuadBox is enough to connect two rings, the standard recommends using two Quadboxes to avoid introducing a SPoF. Quadboxes apply the same technique as nodes to prevent flooding the network with several copies of the same frame.

Moreover, HSR also allows connecting a ring to a PRP network, using a RedBox for each PRP network. These RedBoxes must be capable of transforming PRP frames into HSR frames and vice versa. Moreover, it must allow to identify replicas received through each RedBox, as PRP nodes will send one copy of each frame through each network.

Actually, HSR allows creating mesh topologies as long as all nodes involved are QuadBoxes. Nodes will forward frames through all the ports, except for the one through which the frame was received, and those ports will forward the frame except if they already forwarded said frame.

Moreover, even though HSR does not define specific mechanisms to enforce hard real-time response, it does suggest the separation of traffic in classes and the allocation of specific moments in time to transmit each type of traffic.

HSR does not include any time redundancy mechanisms to tolerate temporary faults. Thus, temporary faults can only be tolerated using the existing

spatial redundancy, as long as such redundancy is available.

HSR supports time synchronization among the nodes of the system. Even though it does not specify the protocol to be used for this purpose, it is fully compatible with PTP. Note that the available versions of PTP are vulnerable to master clock failures, as such failure introduces non-negligible failover times.

Finally, HSR does not define any error containment mechanisms, other than detecting frames that traverse a node several times. This could lead to the propagation of errors from one faulty-node to the rest of the network. This is specially critical as HSR does not include mechanisms to restrict the failure semantics of the components. Thus, byzantine nodes could introduce inconsistencies that could have catastrophic consequences in the system's operation. These reasons make HSR not adequate to support high reliability, but it does provide certain degree of reliability.

Table 4.2: Specific terminology used in the fault tolerance and real-time protocols surveyed.

Protocol	**Node**	**Switch**	**Stream**	**Comm. Cycle**	**Comm. Window**	**Transmitter**	**Receiver**	**Time-Triggered**	**Event-Triggered**
HSE FF	Device	Switch	-	Macrocycle	-	Transmitter	Receiver	Scheduled	Unscheduled
PROFINET	I/O-controller, I/O-device	-	Application Relation	Bus Cycle	Channel	-	-	Time-Triggered	Isochronous Real-Time
SERCOSIII	Slave	-	-	Cycle	Channel	Transmitter	Receiver	Time-Triggered	Event-Triggered
EtherCAT	Slave	-	-	Cycle	-	Transmitter	Receiver	Cyclic	Acyclic
EPL	Controlled Node	-	-	POWERLINK Cycle	Phase	-	-	Isochronous	Asynchronous
AFDX	End-System	Switch	Virtual Link	-	-	Transmitter	Receiver	Scheduled	-
AeroRing	-	T-AeroRing	Contract	-	-	Source	Destination	-	Event-Triggered
TTEthernet	End-System	Switch	-	Cluste Cycle	-	Transmitter	Receiver	Time-Triggered	Rate-Constrained
AVB	Station	Bridge	Stream	-	-	Talker	Listener	-	Time-Sensitive
TSN	Station	Bridge	Stream	Cycle	Window	Talker	Listener	Scheduled	Time-Sensitive
FTTRS	Node	Switch	Stream	Elementary Cycle	Window	Publisher	Subscriber	Synchronous	Asynchronous

4.2 Protocols that provide Fault Tolerance and Real-Time to Ethernet

In this section we describe protocols that offer all the services needed to deploy them in fault-tolerant real-time distributed systems, i.e., we describe protocols that provide different real-time guarantees and reliability levels and that can be used in different systems. Table 4.2 shows the equivalence between the terminology used by the designers of each protocol and a more general one.

4.2.1 FOUNDATION Fieldbus High-Speed Ethernet

FOUNDATION Fieldbus (FF) is an application-layer communication protocol designed by the Fieldbus FOUNDATION to suit the needs of industrial systems. FF counts with a specification on top of High Speed Ethernet (HSE) included to support manufacturing applications. It is stacked on top of TCP-UDP/IP, which allows using standard Ethernet on the link and the physical layer. HSE FF is standardised in the document IEC 61158 (IEC, 2014).

The FF H1 bus provides hard real-time guarantees for the underlying communications. To do so, applications rely on holistic scheduling to enforce the timing requirements of each application. Each H1 bus counts with a scheduler that polls the transmission of scheduled (time-triggered) traffic, but also nodes can request transmissions.

HSE FF can be used as a backbone to connect FF buses to each other. This is done through the HSE Linking Device (LD). Nevertheless, this connection does not offer any RT guarantees as the connected FF buses do not share the scheduler.

HSE FF supports space redundancy by allowing to have two completely independent networks. Devices must have two HSE interfaces to connect them to both networks. This allows tolerating at least one permanent fault in any network component with zero failover time. Moreover, even though HSE FF does not include any time redundancy mechanisms to tolerate temporary faults, these can be tolerated using space redundancy as long as it is not degraded by permanent faults. Figure 4.4 shows a redundant HSE FF network, connected to a non-redundant H1 FF bus through an HSE LD.

Moreover, HSE FF supports device redundancy, using passive replication. If the primary device fails, the backup device becomes active. Both devices

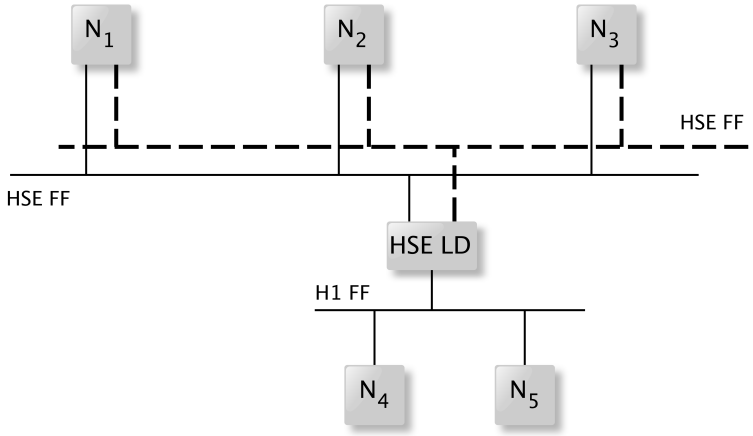


Figure 4.4: Example of 5 nodes interconnected using HSE FF and H1 FF. The HSE FF is duplicated to tolerate permanent faults.

need to share the configuration to ensure fast and fault-free switch-over in case the primary device fails. Nevertheless, HSE FF does not define mechanisms to carry out the switch-over nor to ensure replica determinism.

HSE FF provides mechanisms to monitor the status of the network. Each device and HSE LD contains a component that keeps track of the healthy and faulty network components and connected devices. Each component monitors the network independently, by sending and receiving redundancy diagnostic messages. These messages are used to track the health of redundant devices and ports; and allows the device to change the transmission port in consequence.

Finally, to the best of the authors' knowledge HSE FF does not propose mechanisms to restrict the failure semantics of devices or to contain the errors that may derive from faulty ones. Thus, HSE FF provides reliability, but is not suitable for highly-reliable networks.

4.2.2 PROFINET

PROFINET stands for PROcess Field NET and it is a link-layer protocol that supports the transmission of real-time traffic over standard Ethernet. It was specifically designed for industrial applications and is standardised in IEC 61158 and IEC 61784 (IEC, 2014).

PROFINET supports the use of ring, star and tree topologies. To do so,

nodes are always connected to the network through a switch; which can be either embedded in the device or a separate component. To provide fault tolerance, PROFINET relies on ring topologies.

It divides the communication in cycles, called bus cycles. Each cycle is in turn divided into three channels, namely the Real-Time (RT), the Non-Real-Time (NRT) and the Isochronous Real-Time (IRT) channel. It follows a provider/consumer model for data exchange.

Each channel uses a different policy to provide timing guarantees. The RT channel uses priority-based scheduling, in which the RT traffic has the highest priority to prevent it from being blocked. Moreover, IRT traffic has tighter timing constrains and, to meet them, this traffic is scheduled following a cyclic executive scheduler. This allows avoiding collisions and buffering, but it requires dedicated hardware and a tight synchronization among devices. UDP/IP traffic is transmitted if there is time available after the IRT slots, in the NRT channel.

PROFINET differentiates three types of devices: I/O-Controller, which controls the automation task; I/O-Device, a field device that receives commands from the I/O-Controller; and I/O-Supervisor, which configures and monitors the network.

Each I/O-Device can diagnose its own faults (such as defective voltage) and inform the I/O-controller. The information related to faults is transmitted as an alarm. Moreover, I/O-devices take advantage of the scheduled communication to detect errors in the providers.

PROFINET defines methods to use ring topologies to tolerate permanent faults. More precisely, it defines the use of the previously described MRP if failover times can be accepted and the use of Media Redundancy for Planned Duplication (MRPD) otherwise. MRPD relies on a ring topology to provide fault tolerance against permanent faults. Figure 4.5 shows a PROFINET network with a ring topology managed by MRP.

On the other hand, PROFINET does not include any time redundancy to tolerate temporary faults. Thus, it can only tolerate temporary faults if two networks work in parallel, as long as there are no permanent faults.

Even though PROFINET relies on master/slave clock synchronization to ensure that the real-time requirements of the network are met, it does not include any mechanisms to provide the master clock with fault tolerance. Thus, the master clock represents a SPoF.

Finally, to the best of the authors' knowledge PROFINET does not include

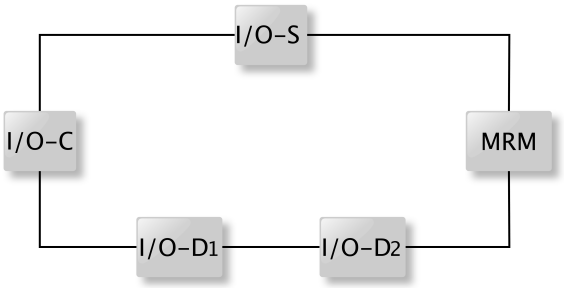


Figure 4.5: Example of 5 nodes interconnected using PROFINET, where redundancy is managed using MRP. The network counts with an I/O-Controller, an I/O-Supervisor, two I/O-Devices and one MRP MRM (Media Redundancy Manager).

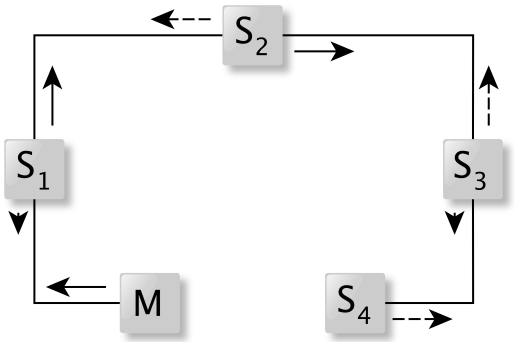
error containment mechanisms nor restricts the failure semantics of the devices. Thus, a device with byzantine failure semantics could interfere with the correct operation of the system. For these reasons, PROFINET is a suitable protocol to provide safety to the network, but not high reliability.

4.2.3 SERCOS III

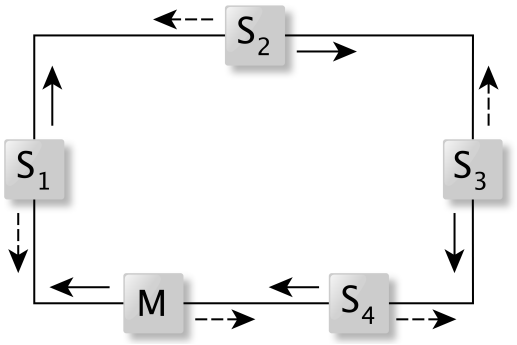
SERCOS stands for SErial Real-time COmmunications System. SERCOS III is the third generation and it is an automation bus based on Ethernet. It is a data link-layer protocol that provides real-time guarantees and low bandwidth consumption to support automation applications. It is standardised as part of the IEC61784 standard part 2 (IEC, 2010b).

SERCOS follows a master/multi-slave architecture; in which the master manages the communication among slaves. SERCOS III supports line and ring topologies, as shown in Figure 4.6. Nevertheless, regardless of the physical topology SERCOS III enforces a logical circular topology. To do so, each slave sends the telegrams (frames) received in one of its ports through the other one. In a line topology, the last slave will send the telegrams through the receiving port once they are processed, as shown in Figure 4.6a. In a ring topology, the last slave is connected to the master and the master sends each telegram in both directions simultaneously, as shown in Figure 4.6b.

SERCOS divides the communication time into cycles, and each cycle is in turn divided into two parts, called channels. The first channel is devoted to the transmission of real-time time-triggered traffic; the second channel is used for the transmission of event-triggered traffic with no real-time requirements.



(a) Example of 5 nodes connected using SERCOSIII logical ring topology.



(b) Example of 5 nodes connected using SERCOSIII physical ring topology.

Figure 4.6: Examples of 5 nodes connected using the logical and physical ring topologies proposed in SERCOSIII. The M box represents the Master; whereas the S_i boxes represent the slaves that communicate.

The channels are isolated to provide RT guarantees to TT traffic.

Moreover, the master transmits a telegram that contains the schedule for that specific channel. After that, the master transmits the Acknowledge Telegram (AT), which is populated by the slaves that are scheduled with the data they need to transmit. Slaves process the telegrams on-the-fly, reducing the time required to carry out the data exchange.

SERCOS III relies on clock synchronization to ensure real-time capabilities. The master is used as a reference clock and transmits synchronization information at the start of every cycle. Even though it is clear that the master is key for the operation of the system, SERCOS III does not include any mechanisms to eliminate the SPoF it represents.

Moreover, the telegram used to synchronize and trigger the communication is a SPoF too. This is so because if the telegram is lost, nodes cannot communicate during the first channel. This is specially critical, as no time redundancy mechanisms are included to tolerate the loss of said telegram.

The ring topology allows the system to tolerate a single permanent fault affecting a link or a slave. Any slave can detect the failure of a neighbor slave in less than one cycle time and start transmitting as in a line topology. Moreover, SERCOS III supports the hot-plugging of devices and the same mechanism can be used to reintegrate components after a fault.

The use of a single telegram to transmit the data of all slaves helps ensuring data consistency among slaves, as all slaves will receive the AT with the same information when the last slave sends it back to the master. However, a slave with byzantine failure semantics could corrupt the information of any node. Moreover, when using a ring topology, a node could introduce different information in the ATs that it receives in opposite directions, creating inconsistencies.

Moreover, SERCOS III supports oversampling to allow slaves to send several values in a single AT every cycle. Nevertheless, as all the values are sent in a single AT, it does not provide any benefits in front of temporary faults. Thus, SERCOS III can only tolerate a single temporary fault as long as the network does not suffer any permanent faults.

For the reasons discussed above, SERCOS III is a suitable protocol to provide safety, but not high reliability.

4.2.4 EtherCAT

EtherCAT stands for Ethernet Control Automation Technologies. It is a data link-layer protocol that provides real-time guarantees and high bandwidth efficiency for automation applications. It is standardised as part of the IEC61158 standard part 1 (IEC, 2014).

EtherCAT is based on a master/multi-slave communication architecture; in which the master manages the slaves. EtherCAT supports line, tree, star and ring topologies. Nevertheless, regardless of the physical topology EtherCAT enforces a logical circular topology. To do so, each slave sends the frames received in one of its ports through the other one. In a line topology, the last slave will send the frames through the receiving port once they are processed. In a ring topology, the last slave is connected to the master and the master sends each frame in both directions simultaneously, as shown in Figure 4.7.

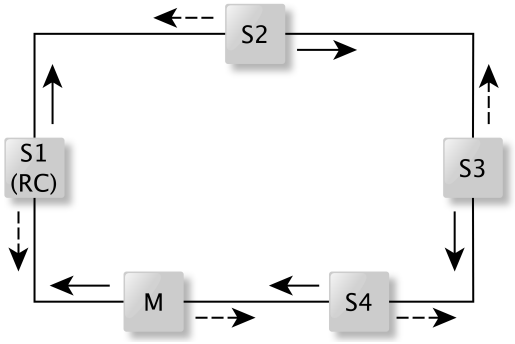


Figure 4.7: Example of 5 nodes interconnected using EtherCAT. The network counts with an EtherCAT master, a slave that also acts as reference clock (RC) and three regular slaves.

As mentioned, in EtherCAT the master manages the communication. To do so, it transmits an Ethernet frame that is then processed and populated by the slaves on-the-fly; reducing the time required for transmitting and receiving. In this way, the master in EtherCAT can be implemented using COTS Ethernet hardware, but slaves must use specialized hardware to process frames on-the-fly.

EtherCAT relies on clock synchronization to ensure real-time capabilities. Similarly to the communication, the EtherCAT master is responsible for managing the synchronization among slaves. To do so, it periodically transmits a special synchronization frame. Nevertheless, it is not the reference clock. Instead, the first slave attached to the master is the reference clock. When the reference clock receives the synchronization frame, it writes its local time on it. The rest of the slaves use this information to resynchronize. Note that synchronization frames are only transmitted through one port, even in ring topologies.

In EtherCAT the master is key for the correct operation of the system, as it is responsible for triggering both, the communication and the synchronization. Nevertheless, there are no mechanisms to tolerate its failure. Moreover, the frame used to synchronize the slaves is a SPoF. This is specially critical, as no time redundancy mechanisms are included to tolerate the loss of said frame.

When connected in a ring topology, the master transmits the Ethernet frame for communication through both ports simultaneously. Slaves populate the first master frame they receive and forward the redundant one without

information. This allows the system to seamlessly tolerate a single permanent fault affecting a link or a slave.

The use of a single frame to transmit the data of all slaves helps ensuring data consistency among slaves. However, a slave with byzantine failure semantics could corrupt the information of any node. Moreover, when using a ring topology, a node could introduce different information in the master frames that it receives in opposite directions, creating inconsistencies. Moreover, temporary faults in links could corrupt the frame, causing the loss of the information transmitted by all the slaves in a given cycle.

Moreover, EtherCAT provides mechanisms to detect the failure of a single slave and to enforce retransmissions. This allows tolerating temporary faults in the slaves. Nevertheless, this mechanisms rely on information provided by the slave. Therefore, it may not be possible to detect the failure of a slave that fails with byzantine mode, as it can introduce incorrect status information.

For the reasons discussed above, EtherCAT is a suitable protocol to provide safety, but not high reliability.

4.2.5 Ethernet Powerlink

Ethernet Powerlink (EPL) (*POWERLINK Basics: System Overview*) is a link-layer protocol that extends Ethernet's link-layer with a scheduler to support hard real-time communications. It is designed to support real-time applications. The protocol can be implemented in software, using standard Ethernet hardware to enable interoperability with standard Ethernet devices.

It is designed to operate in bus topologies supporting the use of hubs, but it can also operate in ring, star, tree or daisy-chained topologies. The use of switches is not recommended as queuing delays are not considered by the scheduler and could jeopardize the timeliness of real-time traffic.

EPL is based on a master/slave architecture, in which the master is called Manager Node (MN) and the slaves are called Controlled Nodes (CNs). The MN enforces synchronization in the network and controls the communication among the CNs.

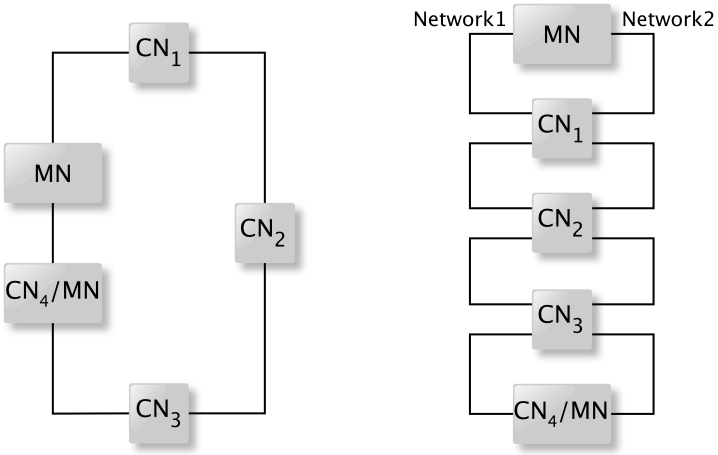
EPL provides hard real-time guarantees for time-triggered traffic, but not for event-triggered traffic. ET traffic can be transmitted with no timing guarantees. To support both, real-time and best-effort traffic, EPL divides the communication in cycles that are, in turn, divided into three phases. The

first phase is used by the MN to synchronize the CNs. The second phase is divided in slots, each assigned to a CN. The transmission in each slot is triggered by the MN. In the third phase, the MN grants access to the channel to a node to transmit ET traffic.

EPL provides fault tolerance against permanent faults. Specifically, it supports two types of channel redundancy, as shown in Figure 4.8:

- Ring redundancy: nodes are daysichained into a ring, this way, when one line (link) is affected by a permanent fault the ring topology becomes a line topology, maintaining the communication. However, this change requires the time equivalent to one cycle to become effective and, therefore, the redundancy is not seamless. Figure 4.8a shows an EPL ring topology.

- Independent networks: as in HSR, PRP and many other protocols, EPL allows to connect the nodes to two failure-independent networks, providing seamless redundancy. Figure 4.8b shows an EPL replicated network.



(a) Example of 5 nodes connected using an EPL ring topology.

(b) Example of 5 nodes connected using two independent daisy-chained EPL networks.

Figure 4.8: Examples of 5 nodes connected using a ring and two independent networks using EPL. The MN node represents the Manager Node, the CN_i nodes represent the Controlled Nodes that communicate; and the CN_4/MN node acts as a Controlled Node and a passive replica of the Manager Node.

Moreover, EPL describes redundancy mechanisms of the Management Node, to avoid the SPoF a simplex MN would represent. More precisely, EPL uses passive replication of the MN, allowing to have more than one MN in the network. A back-up MN is seen as a CN by the active MN, nevertheless, it continuously monitors the network to ensure that the switch-over can be done on-the-fly when the active MN fails. This mechanism allows ensuring that the network continues to operate even if the first MN fails.

EPL does not use time redundancy to tolerate temporary faults in the channel. Moreover, to the best of the authors' knowledge, there are no mechanisms to support the time redundancy of the synchronization frame, which is a critical frame as it is used to synchronize the start of the cycle in all the nodes.

Finally, EPL counts with the POWERLINK Safety protocol, which can detect errors in the communication to avoid catastrophic consequences. POWERLINK Safety is an error detecting protocol as it does not aim at tolerating faults but at detecting them and reporting them for the system to evolve to a potential safe state.

For the reasons discussed, EPL is a suitable protocol to provide safety, but not high reliability.

4.2.6 Avionics Full-Duplex

Avionics Full-DupleX (AFDX) is a communication protocol originally developed by Airbus that aims at providing hard real-time and high-reliability over Ethernet networks for airplane control communications. To do so, AFDX proposes a series of mechanisms to extend Ethernet's link-layer. Nowadays, AFDX has become an ARINC standard and is specified in part seven of ARINC 664 (Airbus, 2009).

AFDX relies on the use of two parallel fail-independent switched-Ethernet networks to carry out the communication (Fuchs, 2012). These networks must have the same topology. Figure 4.9 shows an example of 5 nodes connected using redundant networks managed by AFDX. Each frame transmitted by a node is transmitted through the two parallel networks. Receivers must identify and eliminate duplicates upon reception. Moreover, AFDX guarantees that frames are delivered in the same order in which they were transmitted.

AFDX provides bounded latencies for the transmission of data traffic. To do so, AFDX relies on resource reservation. This guarantees the availability

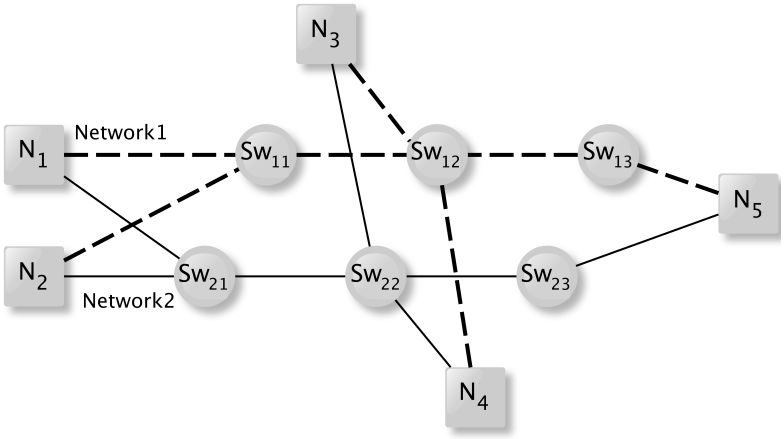


Figure 4.9: Example of 5 nodes interconnected using two independent switched-Ethernet networks using AFDX.

of resources during the communication and limits the bandwidth used by each transmitter. Resources must be reserved for each type of traffic a node wants to transmit. Moreover, resources must be reserved off-line and can not be changed in runtime.

The space redundancy allows tolerating permanent faults in any of the redundant networks with zero failover time. On top of that, to support reconfiguration after a failure in the network, AFDX supports the existence of passive alternative paths that are also scheduled off-line and can be activated in the event of a permanent fault in the active path.

Furthermore, AFDX does not use time redundancy to tolerate temporary faults. Thus, temporary faults can only be tolerated as long as the network is not affected by any permanent faults.

AFDX switches include mechanisms to enforce error containment. More precisely, in case a node or switch fails to meet the bandwidth restrictions, switches prevent the traffic from flooding the network. Nevertheless, to the best of the authors' knowledge, AFDX does not describe mechanisms to deal with byzantine failures. Thus, to prevent inconsistent behaviours that may jeopardize the system's operation, the failure semantics of nodes should be restricted.

For the reasons discussed above, we can conclude that AFDX has mechanisms to provide the network with a certain degree of reliability. Nevertheless, to achieve high reliability must be combined with other protocols to prevent

inconsistencies.

4.2.7 AeroRing

AeroRing is a communication protocol compatible with AFDX that relies on a ring topology to provide tolerance in front of permanent faults (Amari et al., 2016). As AFDX, AeroRing relies on Ethernet, and is designed to support airplane control communications.

In order to support ring topologies, AeroRing defines a specific component called T-AeroRing, a three-port switch that connects to an end node through one port and to the ring through the other two ports. Figure 4.10 shows an example of AeroRing network with 5 nodes connected through T-AeroRings.

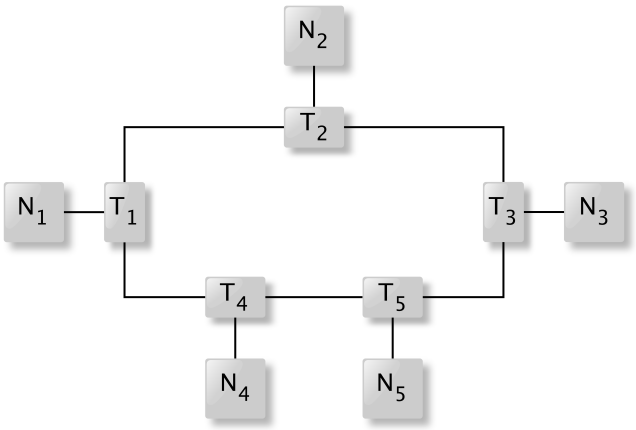


Figure 4.10: Example of 5 nodes interconnected using Aeroring. Each node is connected to the ring using a T-AeroRing. Reproduced as in (Amari et al., 2016).

AeroRing defines four traffic classes, namely the network management class with the highest priority, the Hard Real Time (HRT) class with the second highest priority, the Soft Real Time (SRT) class with medium priority and finally the Non Real Time (NRT) class with the lowest priority. To provide hard RT guarantees Aeroring relies on cut-through switches with static priorities; which allows isolating the different classes of traffic.

T-AeroRing components implement traffic shapers to enforce the requirements established for each type of traffic. Moreover, AeroRing uses traffic policing to ensure that each traffic class consumes the right resources. If a

T-AeroRing detects that a traffic class consumes more resources than entitled it discards the excess frames.

T-AeroRings send critical traffic through both ring-ports in opposite directions, making it possible to tolerate the permanent fault of a single T-AeroRing or link. Non-critical traffic is sent through a single port, that corresponds to the shortest path to the destination.

Moreover, AeroRing describes mechanisms to allow T-AeroRings to detect the failure of a network device. In this way, the other T-AeroRings can change the routing tables accordingly to ensure that the transmission of non-critical traffic is re-established. Moreover, T-AeroRings also support the reintegration of network devices.

AeroRing supports another level of redundancy by allowing devices to be connected to two independent and active rings. To that, the nodes should have two ports, each connected to one of the rings. In this case, additional frame replica detection and elimination mechanisms must be implemented in the nodes to discard replicated frames received through both rings.

AeroRing does not use time redundancy to tolerate temporary faults. Thus, temporary faults can be tolerated as long as permanent faults do not affect any network device. When using two rings at the same time, one simultaneous permanent and temporary fault can be tolerated.

In order to prevent frames from traversing the network several times T-AeroRings discard the frames they sourced. Moreover, T-AeroRings can detect frames with erroneous source addresses by comparing them to the routing table. This allows eliminating impersonations. Nevertheless, to the best of the authors' knowledge AeroRing does not include the use of mechanisms to eliminate two-faced behaviours in the T-AeroRings. For these reasons, AeroRing provides the network with reliability, but not high reliability.

4.2.8 Time-Triggered Ethernet

Time-Triggered Ethernet (TTEthernet) is a real-time protocol compatible with IEEE 802.3 Ethernet. It provides link-layer services to support time-triggered hard real-time traffic over Ethernet devices (Kopetz et al., 2005) (Kopetz, 2008). It is designed to support safety applications and cyber-physical systems.

TTEthernet is based on switched Ethernet and supports mesh, star and

ring topologies. Moreover, it proposes to use redundancy on the network to tolerate the permanent fault of any system (device). In order to do that, it provides support for using replicated star topologies. Figure 4.11 shows a TTEthernet network with two stars used to tolerate faults.

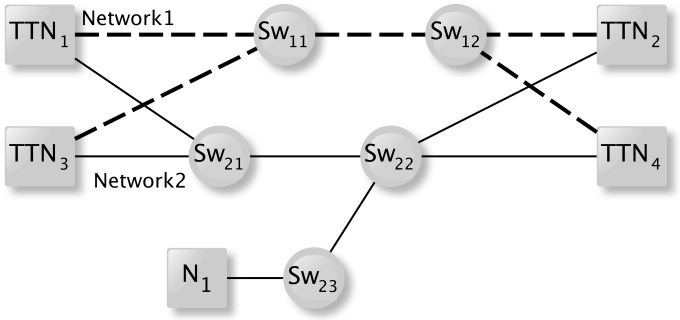


Figure 4.11: Example of 5 nodes interconnected using TTEthernet. The TTN_i nodes represent the Time-Triggered nodes, which are interconnected through two independent networks. The N_1 node represents a COTS node that is connected to the rest using a single network.

TTEthernet supports different classes of traffic, with different timing and reliability guarantees, namely Time-Triggered scheduled traffic; Rate-Constrained (event-triggered with bounded transmission rate) traffic; and Best-Effort standard Ethernet traffic with no guarantees.

In TTEthernet the communication follows a predefined schedule, which allows providing hard RT guarantees to the communication. Moreover, TTEthernet relies on clock synchronization. Specifically, on a clock synchronization mechanism that is based on a convergence function; i.e. the time reference is calculated as a function of the values of different physical clocks. Thus, the synchronization mechanism is fault-tolerant, as it does not rely on a central component.

TTEthernet supports redundancy in the switches and links; eliminating any possibly existing hardware SPoF. Actually, a TTEthernet end-system (node) can have up to three different ports, each attached to a separate network. Being more specific, TTEthernet supports two different replicated topologies:

- Replicated Star: it can support up to three independent stars connecting any pair of nodes. TT and RC traffic is transmitted in parallel through all the available stars. Replicas of the traffic are eliminated at

the receiving end-system. This topology allows tolerating the failure of any network device.

- Ring: TTEthernet-switches are connected forming a ring and each end-system is connected directly to its own switch, which acts as its physical interface with the rest of the network. Switches forward TT and RC frames through both ring-ports; and eliminate replicas and forward one to the attached node. This topology allows tolerating the failure of a single inter-switch link.

On the other hand, TTEthernet does not provide any time redundancy mechanisms to tolerate temporary faults in the links. Thus, temporary faults can only be tolerated as long as the space redundancy is available.

Finally, switches are provided with error containment mechanisms to prevent faulty systems from jeopardizing the correct operation of the overall system. More precisely, the error containment mechanisms proposed in TTEthernet can eliminate byzantine behaviours; such as two-faced behaviours or impersonations. For all these reasons, TTEthernet provides high reliability to the network.

4.3 Protocols that provide Fault Tolerance, Real-Time and Operational Flexibility to Ethernet

In this section we describe protocols that offer all the services needed to deploy them in fault-tolerant real-time distributed systems that are capable of adapting to changes in the environment, i.e., we describe protocols that provide different real-time guarantees, reliability levels and that provide real-time and operation flexibility. Table 4.2 shows the equivalence between the terminology used by the designers of each protocol and a more general one.

4.3.1 Flexible Time-Triggered Replicated Star

The Flexible Time-Triggered Replicated Star (FTTRS) is a highly reliable network architecture built on top of the Flexible Time-Triggered (FTT) communication paradigm. Moreover, FTTRS uses Ethernet as the link and physical layer protocol. FTTRS' FT mechanisms are placed on the link layer (Gessner et al., 2019).

FTT is based on a master/multi-slave architecture, where the master acts as a centralised controller that manages the communication among the slaves. FTTRS is based on an existing FTT implementation which uses a specifically-designed switch called Hard Real-Time Ethernet Switching (HaRTES), in which an FTT master is embedded. The used topology is a mono-hop replicated star with HaRTES switches, shown in Figure 4.12.

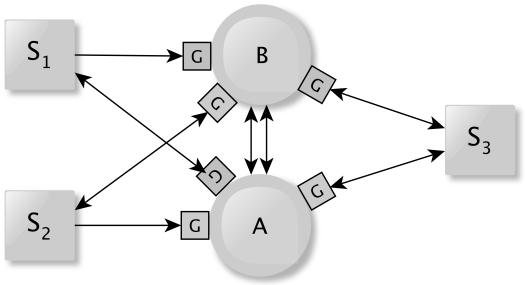


Figure 4.12: Example of 3 nodes interconnected using FTTRS. A and B represent the two HaRTES switches; the S_i boxes represent the communication nodes and the G boxes represent the port guardians.

FTTRS is interoperable with standard Ethernet, supporting the connection of both, COTS nodes and switches, as it can transmit standard Ethernet frames as best-effort traffic without interfering with the RT traffic.

The communication in FTT, and therefore in FTTRS; is divided in communication cycles called Elementary Cycles (ECs). Each EC is in turn divided into three windows, that isolate the transmission of the different types of traffic. At the start of the EC, there is a window for the master to transmit a message that serves to both synchronize the slaves and to communicate them the transmission schedule (which slave has to transmit what) for the current EC. After that, the RT time-triggered and event-triggered traffic are transmitted in two different and consecutive windows. Finally, if there is enough time, best effort traffic is sent.

Moreover, FTT allows changing the traffic requirements (e.g. the periodic messages to be exchanged, their actual periods, etc.) online. Slaves can request changes to the master, which decides whether those specific changes can be made or not and carries out the configuration of the network.

FTTRS provides space redundancy to tolerate permanent faults in the communication channel. As mentioned, FTTRS is based on a replicated HaRTES star, where both HaRTESs are active. FTTRS provides mechanisms to ensure replica determinism between the two active replicas of the

master. All these mechanisms are designed to tolerate at least the concurrent occurrence of one permanent and one temporary fault.

FTTRS provides time redundancy of frames to tolerate temporary faults in the links. This is specially important in the case of the Trigger Message. The TM is used to synchronize, trigger the communication and notify changes in the network to the slaves. In FTTRS the TM is replicated in the time domain to eliminate the SPoF a simplex transmission would represent.

Finally, FTTRS proposes error containment mechanisms that guarantee that the rest of the network is not affected even if slaves fail with byzantine behaviour. These mechanisms are implemented in the *port guardians*, which are placed between the nodes and the HaRTESs and they eliminate frames that correspond to two-faced behaviours, impersonations and timing faults.

For all the reasons previously discussed FTTRS provides a mono-hop network with a high level of reliability.

4.3.2 Fault Tolerance over Audio Video Bridging

Audio Video Bridging (AVB) is the first generation of IEEE standards that aimed at providing Ethernet with real-time capabilities. These standards operate at the link-layer and were devised to support the transmission of audio and video traffic (802.1, 2011b).

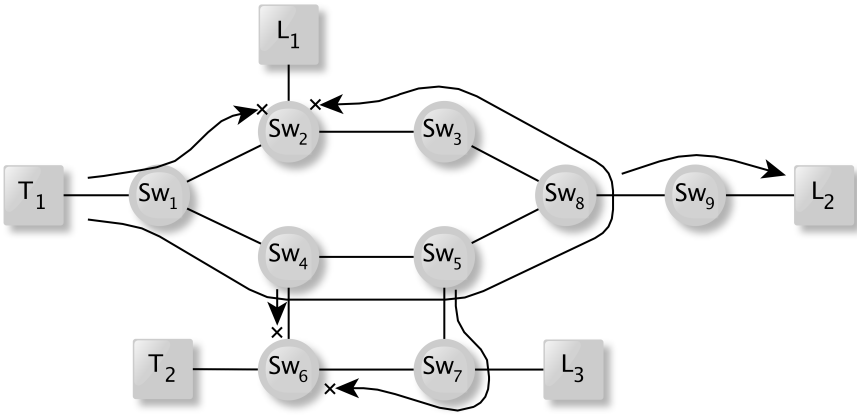


Figure 4.13: Example of 5 nodes interconnected using space redundancy for AVB. The arrows show the logical paths established to communicate the talker T_1 to the listener L_2. Reproduced as in (Kleineberg, Fröhlich, and Heffernan, 2011).

Even though AVB standards cannot provide timing guarantees on their own, they provide the means to do so. In fact, due to the relevance and the impact of these standards there are several analyses to provide AVB with timing guarantees. Some examples of this can be found in (Cao et al., 2018; Ashjaei et al., 2017; Cao et al., 2016; Bordoloi et al., 2014).

Given that AVB is standard Ethernet, it supports mesh topologies. Nevertheless, there are restrictions to the size of the network imposed by certain standards. For instance, the clock synchronization standard only guarantees synchronization accuracy under 1μs for systems up to 7 hops apart. For systems that require highly-accurate synchronization this could lead to a violation of the timing guarantees in larger networks.

To provide soft RT guarantees while keeping the plug&play nature of Ethernet, AVB includes the IEEE Std 802.1Qat (802.1, 2010) Stream Reservation Protocol (SRP). SRP allows end-systems (nodes) that want to communicate to reserve resources along the path that connects them. In this way SRP reduces the probability of frame loss, as there is no possibility of buffer overflow; and increases the determinism in the end-to-end delay.

AVB is comprised of two more standards to enforce real-time guarantees. The first one is IEEE 802.1Qav (802.1, 2009), which describes what is called the Credit Based Shaper (CBS); and the second one is the IEEE Std 802.1AS–2011 (802.1, 2011a), which describes a global clock synchronization mechanism based on PTP.

AVB was designed to provide soft real-time guarantees, but not high reliability via FT. Nevertheless, the interest in extending AVB to provide services for automation and automotive applications motivated the design of fault tolerance mechanisms. Specifically, SRP was extended to support the transmission of frames through several paths in parallel.

To this end, in (Kleineberg, Fröhlich, and Heffernan, 2011) the authors propose a mechanism to allow SRP to establish redundant streams. These redundant streams are registered whenever two or more end-systems signal their wish to communicate. Figure 4.13 shows an example of the redundant logical paths established using the mechanism to interconnect T_1 and L_2.

Nevertheless, AVB cannot manage redundancy and, therefore, an additional redundancy control protocol needs to be used. If RSTP is used, there is a failover time whenever a network device fails, as RSTP needs some time to establish the new logical path before frames can be forwarded through it. Nevertheless, if used together with other redundancy control protocols, such as PRP or HSR, the failover time is zero, as the redundant paths are active

simultaneously.

Regardless of these efforts to increase the reliability of AVB networks, there are some limitations to their deployment in critical systems. For instance, the IEEE Std 802.1AS standard is vulnerable to failures in the master clock which is a SPoF.

Moreover, AVB does not include any time redundancy mechanisms to tolerate temporary faults in the links. Thus, if passive replication is used for spatial redundancy, temporary faults cannot be tolerated even in the absence of permanent faults.

Finally, there are no mechanisms to restrict the failure semantics of the devices or to achieve error containment. Therefore, inconsistencies in the communications may happen, among other potential error scenarios. For the discussed reasons, AVB provides availability, but not reliability.

4.3.3 Time-Sensitive Networking

Time-Sensitive Networking is a Task Group of the IEEE that has been working on the standardization of hard real-time, high-reliability and on-line configuration services for standard Ethernet (*Time-Sensitive Networking (TSN) Task Group.*). The set of standards developed by this group are usually referred to as Time-Sensitive Networking (TSN) too. TSN is an evolution of AVB, which aims at providing mechanisms to support automation, control and automotive networks, among others.

In TSN each standard defines a service and these services can be combined to create tailored networks that meet the requirements of a great variety of applications; including the ones supported by AVB. TSN supports mesh topologies, but like in AVB there are restrictions to the accuracy of clock synchronization achievable in systems separated by more than 7 hops.

TSN supports time-triggered hard real-time traffic with the standard IEEE Std 802.1Qbv (802.1, 2016a) Time-Aware Shaper; event-triggered soft real-time traffic is supported with the AVB standard IEEE Std 802.1Qav Credit-Based Shaper. Other traffic shapers for TT traffic are defined in the standard IEEE Std 802.1Qch (802.1, 2017b) Cyclic Queuing and Forwarding and IEEE Std 802.1Qcr (802.1, 2020a) Asynchronous Traffic Shaper.

To provide the timing guarantees required by control applications, TSN relies on clock synchronization. In fact, the TSN TG has developed the IEEE Std 802.1AS-2020 (802.1, 2020b) to provide the clock synchronization

mechanism proposed in AVB with higher reliability by enforcing seamless redundancy of the master clock.

Moreover, the standard IEEE Std 802.1Qcc (802.1, 2018) is an evolution of the IEEE Std 802.1Qat SRP. Qcc introduces mechanisms to carry out the reconfiguration of the network in a centralised manner. Specifically, Qcc describes two new configuration models; the *Centralized Network/ Distributed User* (CN/DU) and the *Fully Centralized* (FC) models.

The CN/DU model proposes the use of a Centralized Network Configuration (CNC) entity, responsible for managing and configuring the network. Stations can transmit their network-related requirements to the bridge they are attached; which will in turn forward it to the CNC. Finally, the CNC processes all the requests and distributes the changes through the network.

On the other hand, the FC model proposes the creation of the Centralized User Configuration (CUC) entity, responsible for managing the end-systems requirements. The CUC receives application-related requirements and transforms them into network-related requirements for the CNC to process them.

These models allow to significantly reduce the time required for reconfiguration as we show in (Álvarez et al., 2020), and they also allow supporting new enhanced services for reliability. Nevertheless, the specification of the CNC and the CUC is out of the scope of the standards. Moreover, to the best of the authors' knowledge there are no mechanisms to support their replication. Therefore, the centralised architectures introduce new SPoFs.

In order to increase the reliability of data frames, the TSN Task Group proposed three different standards. The first two standards are concerned with space redundancy. More precisely, the IEEE Std 802.1Qca (802.1, 2016b) allows establishing multiple redundant logical paths to connect nodes that want to communicate; whereas IEEE Std 802.1CB (802.1, 2017c) allows creating redundant streams on top of the logical paths created by Qca. Specifically, Qca is an extension of the SPB that supports seamless redundancy and resource reservation. Figure 4.14 shows an example of the redundant logical paths established using Qca and CB to interconnect T_1 and L_2. It is important to note that the CB standard is independent from the IEEE Std 802.1Q, in the sense that it is not an amendment nor a revision of the latter. Thus, CB can be used to provide space redundancy to other protocols.

There can be as many redundant logical paths as the physical topology allows. Thus, the number of permanent faults that can be tolerated will depend on the physical topology. Moreover, Qca also allows establishing a new logical path whenever a network device fails, reducing the redundancy

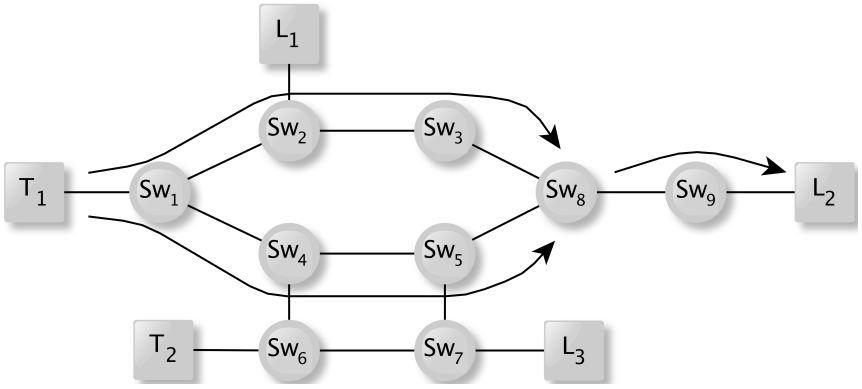


Figure 4.14: Example of 5 nodes interconnected using TSN. The arrows show the logical paths established to communicate the talker T_1 to the listener L_2.

attrition. Furthermore, as long as there are several active paths, this reconfiguration will be done seamlessly. Nevertheless, CB introduces the possibility of frames arriving in a different order they were transmitted.

TSN does not include any time redundancy mechanisms specifically designed to tolerate temporary faults on the network. Therefore, in order to tolerate both, temporary and permanent faults happening simultaneously, there must be at least three redundant paths to connect any pair of nodes.

The last standard, IEEE Std 802.1Qci (802.1, 2017a) Per-Stream Filtering and Policing defines two mechanisms for error containment. The first one allows to identify and eliminate frames that arrive out of time, to prevent them from interfering with the rest of the traffic. The second mechanism allows detecting components failing as babbling idiots, i.e. components that get stuck transmitting a frame over and over again. Frames arriving from the failing component are dropped or reallocated to prevent them from flooding the network.

Nevertheless, TSN does not propose any mechanisms to detect and eliminate byzantine behaviours, such as two-faced behaviours. These behaviours may arise due to spatial replication and could cause inconsistencies in data transmissions. For the reasons discussed above, TSN provides a certain degree of reliability, but it is not suitable as it is for highly reliable networks.

4.4 Summary

In this Chapter we have described industrial protocols that use fault tolerance in different ways. More specifically, we have distinguished protocols that offer a single fault tolerance solution, those that provide fault tolerance and real-time services and those that also provide operational flexibility. We have also grouped the protocols described into four different categories depending on the dependability attribute they enforce: availability, safety, reliability and high reliability. Table 4.3 shows this classification.

Attribute	**Protocols**
Availability	STP, SPB, MRP, AVB
Safety	PROFINET, SERCOS III, EtherCAT, EPL
Reliability	PRP, HSR HSE FF, AFDX, AeroRing, TSN
High Reliability	TTEthernet, FTTRS

Table 4.3: Summary of protocols grouped by dependability attribute.

As we have discussed, all STP versions, SPB, MRP and the AVB FT proposed solution improve the availability of the network. PROFINET, SERCOSIII, EtherCAT and EPL all increase the safety of the system. PRP and HSR provide reliability with no real-time; whereas HSE FF, AFDX, AeroRing and TSN provide reliability together with timing guarantees. Finally, TTEthernet and FTTRS provide FT services to achieve high reliability for RT applications.

On top of that, Table 4.4 shows a classification of the protocols in terms of the fault tolerance aspects and the most relevant network features discussed. Specifically, we distinguish the type of RT guarantees of the protocols; whether they can operate in multi-hop networks and whether they provide any flexibility, i.e. real-time or operational flexibility.

We can see that there are few protocols that provide real-time guarantees and flexibility. Specifically, STP, SPB, MRP, PRP and HSR offer certain degree of operational flexibility, as they allow to modify the operation of the network to deal with failures or to support the connection of new devices. Nevertheless, these changes are not done with any real-time guarantees; i.e. the time required for the changes is not bounded. Moreover, they do not provide real-time flexibility, as they do not support RT communications.

On the other hand, we find more sophisticated protocols such as AeroRing, AVB and TSN that provide higher degrees of flexibility. First, all these protocols support real-time flexibility; even though AVB does not support hard real-time traffic. Moreover, AeroRing, AVB and TSN provide operational flexibility, as they support changes in the traffic, but do not bound the time required to perform the changes.

Regarding TTEthernet, we see that it provides the services required by highly-reliable systems and can operate in multi-hop networks. Moreover, it supports real-time flexibility, as it can convey different types of traffic. Nevertheless, it does not provide any operational flexibility. Finally, FTTRS supports both, operational flexibility, as it allows modifying the traffic requirements online and within a bounded time; and real-time flexibility, as it supports hard, soft and non-real-time traffic. Nevertheless, FTTRS is a mono-hop architecture and cannot be deployed in larger systems.

This analysis allows us to conclude that TSN is actually the most adequate technology to build networks that meet the all the requirements of our thesis statement, i.e. networks that are multi-hop and provide real-time and operational flexibility. On top of that, TSN is actually an evolution to standard Ethernet and, thus, it has the potential to provide system-wide integration to the networks of future systems.

Unfortunately, TSN cannot provide high reliability in a cost-effective manner as it does not include any time redundancy mechanisms. For this reason, we propose to use PTRF to increase the reliability of TSN networks in the presence of temporary faults.

Table 4.4: Classification of the protocols in terms of fault tolerance aspects and network features they fulfil.

Protocol	Fault Tolerance Aspects					Network Features		
	Path Redund.	**Time Redund.**	**Manager Redund.**	**Master Clock Redund.**	**Network Consistency**	**Real-Time**	**Multihop**	**Any Flexibility**
STP	Failover	None	Failover	-	No	No	Yes	Yes
SPB	Failover	None	-	-	No	No	Yes	Yes
MRP	Failover	None	Failover	-	No	No	Yes	Yes
PRP	Seamless	None	-	-	No	No	Yes	Yes
HSR	Seamless	None	-	Failover	No	No	Yes	Yes
HSE FF	Seamless	None	Failover	Failover	No	Hard	Yes	No
PROFINET	Seamless	None	Failover	None	No	Hard	Yes	No
SERCOSIII	Seamless	None	None	None	No	Hard	Yes	No
EtherCAT	Seamless	Failover	No	Failover	No	Hard	Yes	No
EPL	Seamless	None	Failover	Failover	No	Hard	Yes	No
AFDX	Seamless	None	-	-	No	Hard	Yes	No
AeroRing	Seamless	None	-	-	No	Hard	Yes	Yes
TTEthernet	Seamless	None	-	Seamless[a]	Yes	Hard	Yes	No
FTTRS	Seamless	Seamless	Seamless	Seamless	Yes	Hard	No	Yes
AVB	Failover	None	-	Failover	No	Soft	Yes	Yes
TSN	Seamless	None	None	Seamless	No	Hard	Yes	Yes

Seamless: redundancy allows tolerating faults without interruption in the service.
Failover: the service is interrupted for certain time after a fault occurs.
None: there is no redundancy to tolerate faults.
- : does not apply to the protocol.
TTEthernet's synchronization is actually based on a convergence function; i.e. the time is calculated as a function of different physical clocks.

Chapter 5

Operation of a Network based on the Time-Sensitive Networking Standards

The Time-Sensitive Networking (TSN) Task Group (TG) is part of the IEEE 802.1 Working Group, which is in charge of developing standards for the data link layer of the Ethernet protocol. Specifically, the TSN TG has been working to provide the data link layer of Ethernet with real-time capabilities, online management of the traffic and fault tolerance mechanisms. To achieve this goal, the TSN TG has already completed 27 technical standards and is currently working on 18 standardisation projects.

In the previous chapter, we have discussed what the community considers to be the most relevant standards proposed by the TG. Nonetheless, the mechanisms proposed in the standards must be properly selected and integrated to build an adequate network for each application. Therefore, an important part of our work consists in deciding which standards will build our network. In this chapter we provide a general description of the type of TSN network that we assume in this dissertation and we discuss the most relevant aspects of the TSN standards that implement it. On top of that, we briefly describe other TSN standards that can operate in our network but that are not key for the proper operation of the mechanisms proposed in this dissertation.

5.1 A General Overview of a TSN Network

As we already know at this point, TSN networks are multi-hop bridged networks. Therefore, the main devices in a TSN network are the end-systems, the bridges and the links that connect them. The end-systems are responsible for executing the application and creating or consuming the information that is exchanged through the network; the bridges are responsible for switching the information from the source to the destination through the established path and the links physically connect end-systems and bridges.

Figure 5.1 depicts a TSN network architecture. End-systems are represented using squares, bridges are represented with circles and links are represented with arrows. As we can see, there are four end-systems and six bridges, connected in a mesh topology. Furthermore, end-systems can play two different roles: talkers (represented with a T), which produce and transmit information, and listeners (represented with an L), which consume the information created by a certain talker. Each end-system can transmit and consume several flows of information, i.e. each end-system can be a talker and a listener at the same time.

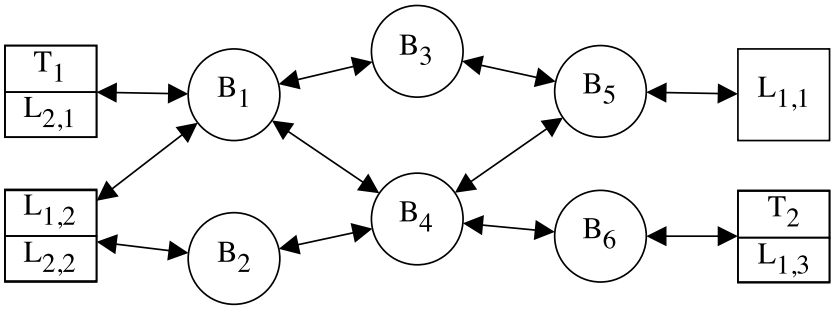


Figure 5.1: An example of a TSN-based network architecture with four end-systems and six bridges in a mesh topology. End-systems are represented with squares, bridges with circles and links with arrows. The T means that the end-system is a talker, while the L means it is a listener.

The applications executed in the end-systems use the network to exchange information through *messages*. In TSN networks, messages are exchanged using *streams* (802.1, 2010; 802.1, 2018). A stream is a virtual communication channel used to convey traffic with specific characteristics, e.g. traffic with certain period and frame size. Each stream has a single talker and one or several listeners which communicate following a publisher-subscriber model.

All the traffic transmitted through a specific stream conveys information from the same source, e.g. a temperature sensor or a camera. In fact, the traffic of a stream conveys the same information from the same source captured in different moments, e.g. the value of the temperature in instant t_0, t_1, t_2, etc. We call these the *editions* of a message, or message editions. Whenever an edition of a message must be transmitted through the network, it is embedded together with control information, such as the stream identifier, in a *frame*. A frame is a chain of bits that is injected and transmitted through the physical layer.

As we know, TSN networks can convey traffic with different real-time characteristics. To that, TSN divides the communication time into slots called cycles. Each cycle is in turn divided into windows and each window is dedicated to a different type of traffic. Figure 5.2 shows an example where the communication cycle is divided between a protected and an unprotected window, separated by a guard band to prevent unprotected traffic from interfering with protected one. In the example, the protected window is devoted to the transmission of traffic with real-time requirements, while the unprotected window is devoted to the transmission of best-effort traffic.

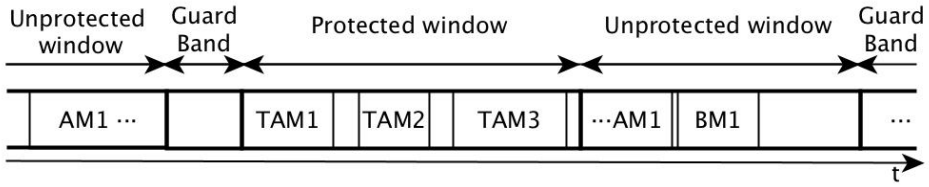


Figure 5.2: A communication cycle example divided into a protected window, an unprotected window and a guard band.

Nevertheless, the communication cycle can be fully customised, i.e. it can be divided in as many windows and traffic types as desired. To achieve this, the Time-Aware Shaper (TAS) is based on the *transmission gate* concept. Specifically, TAS proposes the addition of transmission gates to the output queues of the ports. Each gate determines whether the frames in the queue can be elected for transmission or not. In this way, TAS can keep the main structure of standard TSN ports untouched, with eight queues, each one corresponding to a different priority, 0 being the lowest and 7 being the highest.

Regarding the operation of TAS, each transmission gate can be in one of two states: (i) open: queued frames can be elected for transmission by the

transmission selection algorithm, or (ii) closed: queued frames cannot be elected for transmission. Figure 5.3 shows the internal structure of a port that implements the TAS. At time instant T0, the gate of the egress queue of priority 7 is open, whereas the other gates are closed.

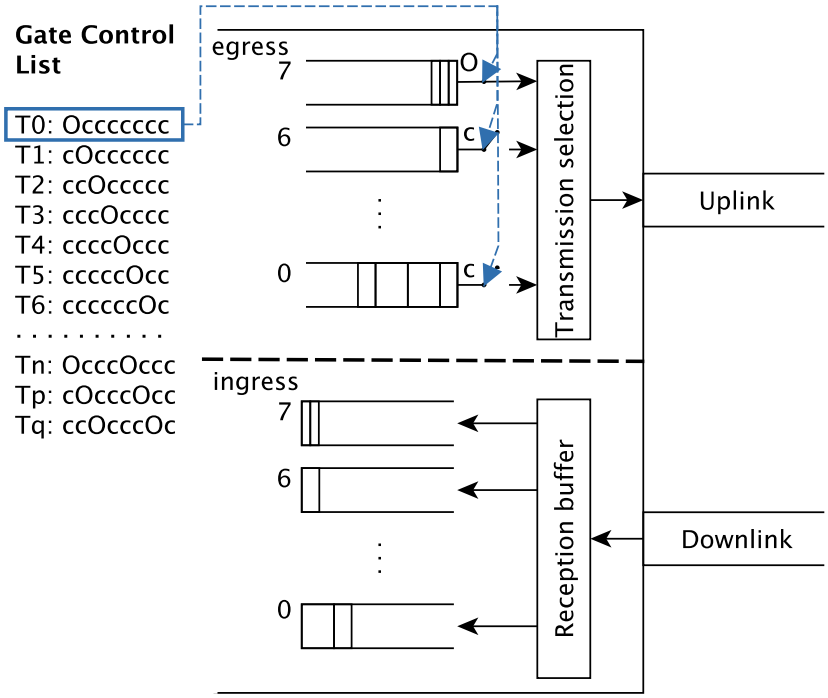


Figure 5.3: Internal structure of a port with Time-Aware Shaper. Each egress queue of the port has a gate that can be configured as open 'O', to allow frames to be transmitted, or closed 'c', to prevent frames from being transmitted. On the left-hand side of the figure we can see an example of gate control list.

To provide hard real-time guarantees the gates of all queues in all ports of both nodes and bridges must be properly configured and synchronised. To that, TAS relies on *gate control lists*. A gate control list is an ordered sequence of gate states which defines the state of each queue during the different windows of the communication cycle. That is, a gate control list dictates when each gate of a port is open and when it is closed, enforcing the desired schedule. On the left-hand side of Figure 5.3 we see an example of gate control list with the current gate states highlighted in blue.

Furthermore, TAS can be used together with other traffic shapers and transmission policies that rule the transmission of frames within a window. To that, switches implement the shapers and policies, and the adequate rule is selected by the transmission selection algorithm in the port, as shown in Figure 5.3. For instance, the TSN TG has standardised the CBS (802.1, 2009) to provide soft real-time guarantees to the transmission of event-triggered traffic.

Furthermore, to synchronise the transmission of frames throughout the network TSN can use the Cyclic Queuing and Forwarding (CQF) (802.1, 2017b). This mechanism roughly consists in assigning a transmission window to each hop, in such a way that frames are forwarded in consecutive windows.

Let us illustrate the operation of CQF with an example. Let us assume that we have the network depicted in Figure 5.1. Let us also assume that talker T_1 wants to communicate with listeners $L_{1,1}$, $L_{1,2}$ and $L_{1,3}$ using the CQF mechanism. T_1 transmits frame $f_{1,1}$ in the communication window w_i, and $f_{1,1}$ reaches bridge B_1 within the same window. In turn, B_1 forwards frame $f_{1,1}$ during window w_{i+1}, and the frame reaches listener $L_{1,2}$ and bridge B_3 within the same window again. Bridge B_3 forwards frame $f_{1,1}$ during window w_{i+2}, bridge B_5 forwards frame $f_{1,1}$ during window w_{i+3} and the frame reaches listeners $L_{1,1}$ and $L_{1,3}$ within window w_{i+3}. Thus, the maximum end-to-end delay of a frame corresponds to the window size $\times$ the number of hops, i.e. it takes four windows for frame $f_{1,1}$ to be transmitted to the furthest listener.

When using CQF, we can define the period of any given stream as a multiple of communication windows, as long as all the windows in the path have the same duration. Thus, we can define the period T of a stream as $T = n \times w$, where T is the period, n is the number of windows and w is the duration of a window. When using PTRF to replicate frames in the time domain, we must take into account that all the copies of a frame (a.k.a. replicas) must fit in the same window when defining w, as we explain in Chapter 9.

5.2 Other relevant TSN standards

In this section we provide an overview of TSN standards that could be part of the networks that we aim at implementing but that are not part of the core of this dissertation. We start this section describing a series of standards that are key for the operation of any Ethernet network. Then, we

describe TSN standards and we classify them into real-time and management standards. We do not describe the fault tolerance standards as they are already described and their limitations are discussed in the previous chapter.

5.2.1 Base Ethernet Standards

Before getting into the new TSN standards we list a series of Ethernet standards that are required for the correct operation of a TSN network.

- IEEE Std 802.1Q-2018 : This is the main Ethernet data link layer standard. This standard describes Media Access Control services for bridged networks and the services and operations to be carried out by bridges.

- IEEE Std 802.1AB-2016 : This standard defines a protocol and objects to discover the network topology of adjacent stations. That is, it defines a protocol to create paths between nodes that are connected through a physical topology. The protocol is called Link Layer Discovery Protocol.

- IEEE Std 802.1AS-2020 : This document standardises the use of the Precision Time Protocol for global clock synchronisation. It describes how to select a clock master in a redundant manner, how to synchronise the devices and how to detect unsynchronised ones.

- IEEE Std 802.1AX-2020 : This standard defines link aggregation services. It allows using several paths as a single path to increase bandwidth in a way that is transparent for end-systems. It also allows communicating end-systems that belong to different networks, ensuring that the physical path is the same between the two networks.

5.2.2 Real-Time Standards

This set of standards define specific services to provide real-time guarantees to the different types of traffic supported in TSN networks. We have already described the operation of TAS (802.1, 2016a) and CQF (802.1, 2017b), so next we briefly introduce the standard for global clock synchronisation and for the transmission of event-triggered traffic.

IEEE Std 802.1AS-2020

The IEEE Std 802.1AS-2020 standard (802.1, 2020b) describes a profile of the Precision Time Protocol (PTP). PTP is a master-slave synchronization protocol that can be used in networks that rely on frames to convey information (Zurawski, 2015). Specifically, PTP provides the network with global clock synchronization; which means that all the systems share the same notion of time. To that, PTP supports the selection of a master clock to which all other clocks (slave clocks) in the network synchronize.

In PTP, the clock selected as master clock is critical; meaning that a failure of the master clock would cause the loss of synchronization and the system failure. For this reason, the IEEE Std 802.1AS-2020 standard proposes several modifications to increase the reliability of the clock synchronisation mechanism. On the one hand, AS allows sending the synchronization frames through specific routes different from the ones used for data frames. This allows using the optimal synchronization path and reduce the time needed to find a new master clock after a failure. On the other hand, the drift from the master clock is calculated in an accumulative manner in each slave clock. This means that each slave clock knows its relative drift from the master clock which reduces the time needed for synchronization.

Finally, to eliminate the SPoF that the master clock represents, AS supports several active master clocks. This makes resynchronization transparent for slave clocks after a master clock failure. Moreover, each master clock can have its own path for synchronization communication, making this approach more robust in front of failures in the network components between the master clock and slave clocks.

IEEE Std 802.1 Qav-2009

The IEEE Std 802.1Qav-2009 document (802.1, 2009) standardises CBS, which provides soft RT guarantees for two different classes of event-triggered traffic, namely class A and class B. Figure 5.4 shows the operation of the CSB mechanism. Each traffic class has a given credit assigned, which decreases whenever a frame that belongs to the class is transmitted. Once the credit is exhausted (negative), the traffic of that class must wait for transmission and its credit increases. Meanwhile, the traffic from the other class can be transmitted, as long as its credit is positive. This mechanism prevents starvation and bounds the end-to-end delay of the soft RT traffic.

Class A traffic has higher priority and shorter end-to-end delays than Class

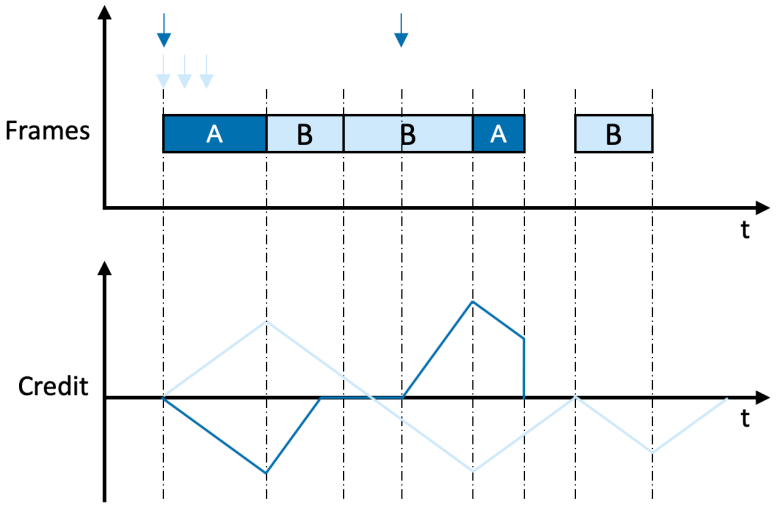


Figure 5.4: Example of the operation of the Credit Based Shaper, reproduced as in (Cao et al., 2018).

B traffic. Traffic that does not belong to class A or B is transmitted in the background, on a best-effort approach. This approach allows transparently attaching standard Ethernet end-systems that do not support any RT mechanisms.

The use of CBS by itself is not enough to bound the end-to-end delay of communications. To do so, CBS must be used together with the IEEE std 802.1Qat which is described in the network management standards.

5.2.3 Network Management Standards

We will next describe the standards that define tools, components, mechanisms and services to allow to control the configuration of the network, including the ones describing YANG data models.

IEEE Std 802.1 Qat

This standard defines the Stream Reservation Protocol (SRP). It introduces the concept of stream into standard Ethernet. Streams in TSN are similar to FTT, but with less flexibility. For starters, there are only two types of

stream, which are referred to as Classes of traffic. The classes are Class A and Class B, and the main difference is the priority, having A higher priority than B. Another limitation is that the period and deadline of streams must be the same. The QoS that each traffic will have is actually conditioned by the Qav standard, which we describe later.

On top of introducing the concept of stream, this standard defines the protocol to reserve resources through the network in a distributed manner. This protocol has two main functions: first it allows for the each talker to signal the listeners and the bridges that it has a new stream with certain characteristics; second, it allows the listeners to signal the bridges and the talker that they want to bind to a given stream previously announced (advertised) by the talker. When a bridge receives a signal from the talker and a response from the listener it checks whether there are resources and reserves them to guarantee that said resources will be available during transmission.

IEEE Std 802.1Qcc

This amendment was devised to enhance and improve the performance of the previously described Stream Reservation Protocol (802.1, 2018). Nonetheless this standard actually covers new aspects such as the definition of new network architectures.

Regarding the enhancements to the SRP, this new amendment allows creating more traffic classes. The main contribution in this sense is the definition of the Scheduled traffic Class, which is basically time-triggered traffic with hard real-time requirements. The amendment also allows users to define their own traffic classes. Finally, the amendment also provides SRP with determinism and it bounds the time needed for reservations, making it suitable to be used in real-time systems.

As we said before, the SRP is a distributed protocol, but this amendment devises two new architectures. These architectures will allow to manage the network requirements in real-time. The first architecture is Centralised Network/Distributed User. This architecture is characterised by having a Central Network Controller (CNC) that is in charge of managing and reconfiguring the network. Note that the CNC is a logical element, which means it may be placed in a network component such as a bridge or node; even though it can also be a stand-alone component.

The second network architecture proposed in the amendment is the Fully Centralised architecture. In this architecture, on top of having a CNC, there

will also be a Central User Controller (CUC). The CUC will be responsible for triggering the changes in the network that are related to the users' requirements.

IEEE Std 802.1CS-2020

This standard describes the Link-local Registration Protocol (802.1, 2021) and defines mechanisms to propagate attributes' registrations. To that it allows replicating the registration database from one end to the other of a point-to-point link and to replicate changes to said database. It also allows removing registrations if the source becomes unresponsive. This new standard is optimised for databases on the order of 1 Mbyte.

The registration of attributes is a low-level service used by a great amount of protocols, among which we find SRP. In fact, the requests transmitted by talkers and listeners to create and bind to a stream are actually attributes. Therefore, the registration of attributes allows identifying the ports through which a specific stream is transmitted, among many other things.

YANG Data Models Standards

- IEEE Std 802.1Qcp-2018: This amendment defines the YANG data models to configure and gather information from bridges and bridge components.
- P802.1Qcw: This document will define the YANG data models for Scheduled Traffic (Qbv), Frame Preemption (Qbu) and Per-Stream Filtering and Policing (Qci).
- IEEE Std 802.1Qcx-2020: This document defines the YANG data model for Connectivity Fault Management. It will support the reconfiguration and the status reporting of network components.
- P802.1ABcu: This amendment of the Link Layer Discovery Protocol describes the YANG data model to configure and status report for topology discovery.
- P802.1CBcv: This document covers the YANG data model for the replication and elimination of frames (to be transmitted through different paths).

Chapter 6

The Proactive Transmission of Replicated Frames Mechanism

In Chapters 2 and 3 we have discussed aspects related to the background of this dissertation, while in Chapters 4 and 5 we have analysed and selected the most adequate technology for developing this dissertation, as well as the type of network that we will use. From now on, we discuss the specific points that will allow us to achieve our thesis statement.

As we have discussed, we want to prove that we can increase the reliability of multi-hop networks based on TSN standards, which support real-time and operational flexibility, by using proactive time redundancy to tolerate temporary faults in the links in a way that is suitable for the RT response of the network.

To prove our thesis we have designed the PTRF mechanism, which provides time redundancy by transmitting several copies of each frame in a preventive manner. In fact, we have designed three different approaches of PTRF which differ from each other in the devices which carry out the implementation and how to calculate the number of frame replicas that must be transmitted.

In this chapter we describe the design of the PTRF mechanism, starting by the design rationale and the fault types and failure modes considered. Once we establish the bases for the design of our mechanism we proceed to describe the details of its operation and design.

6.1 PTRF design rationale

When designing a system or mechanism, there are virtually infinite ways to carry out such design. In order to produce the design of our mechanism and the resulting three approaches, we have taken into account the main devices and characteristics of TSN-based networks. Specifically, we have considered three aspects: (i) which devices should replicate, (ii) which types of traffic should be replicated and (iii) which is the adequate granularity for deciding the level of replication.

As we have seen, there are two devices in a TSN-based architecture that can carry out the replication of frames, end-systems and bridges. In our designs we have covered all the combinations of devices that replicate: only end-systems replicate, only bridges replicate or both end-systems and bridges replicate.

Regarding the types of traffic, we must note that TSN supports TT and ET traffic with real-time guarantees, as well as best-effort traffic. PTRF is designed to provide time redundancy for critical frames, which are usually TT communications. Nevertheless, ET traffic may also convey critical information, such as alarms. Thus, the PTRF mechanism can be used to replicate any type of traffic, even best-effort one. It is the responsibility of the network manager or management entity to decide which traffic should be replicated or not and to ensure the timing guarantees of the traffic.

Regarding the granularity of the replication, we have considered two different options. First, we designed the mechanism to allow for a different level of replication for each stream. Nevertheless, when we moved to the implementation phase we encountered an important limitation to this decision. The number of streams in a network is unpredictable and, most likely, high. Therefore, storing and processing information in a per-stream manner required reserving a significant amount of memory and processing resources. To solve this, we have designed PTRF in a way that the level of replication depends on the priority of the stream. We consider this to be a reasonable trade-off, as in TSN streams that share the same priority usually convey traffic with similar characteristics. Thus, we can assume that their reliability requirements are also similar and can share the same level of replication.

Furthermore, in order to ensure that our design helps us reach our thesis statement we have defined a series of requirements that PTRF must meet. The requirements are the following:

- R1: the mechanism must be fully compatible with standard devices.

- R2: the mechanism must be easily integrable with existing standards.
- R3: the mechanism must not imply significant modifications of standard devices.
- R4: the mechanism must not have a high memory consumption as bridges have a limited amount of memory.
- R5: the mechanism must be flexible enough to be used in virtually any network, even those for adaptive systems.

Requirements R1 to R4 are key to ensure that we can use PTRF in networks that are based on the Ethernet standards, as well as to ensure that we can keep the real-time and operational flexibility that they provide. Thus, all the approaches that we have designed meet requirements R1 to R4.

Regarding R1, even though PTRF requires modifying the devices that carry out the replication of frames and the elimination of surplus replicas, these PTRF devices can coexist with standard devices and PTRF frames can be forwarded by any standard bridge, and vice versa. Regarding R2, we have designed PTRF taking into account the main characteristics of TSN networks. More concretely, we have taken into account the use of streams, priorities and gates to guarantee that our mechanism can be used together with some of the most relevant TSN standards. Unfortunately, we cannot guarantee an easy integration of PTRF with each and every standard proposed by the TSN TG as many of them are unfinished and the number is still growing.

As we have mentioned, we have designed three different approaches. In our first approach, only end-systems replicate frames. This design satisfies requirements R1, R2, R3 and R4. Regarding R3, the modifications are limited to end-systems, which are prone to having custom functionalities, which allows using standard bridges. Regarding R4, since the mechanism is placed in end-systems, bridges do not consume any additional resources. Nevertheless, R5 is not satisfied by this approach, as only nodes can replicate frames and the level of replication is the same for each priority in the whole network, which discards it as an adequate solution for adaptive systems.

In our second approach, both end-systems and bridges replicate frames. This approach meets requirements R1, R2, R3 and R4. R3 is satisfied because bridges do not require the modification of existing mechanisms to support PTRF, only the addition of several components which are described in 6.4.3. Furthermore, Even though this approach implies the modification of bridges, R4 is satisfied because the amount of additional information that

bridges store is not significant, as we use a per-priority replication scheme. Nevertheless, similarly to what happens with the first approach, R5 is not satisfied as all the devices use the same level of replication in the whole network, which varies only depending on the priority.

Finally, we designed a third approach in which both, end-systems and bridges may or may not replicate. That is, all devices can replicate, but they do not necessarily do it. Furthermore, the devices that do replicate can have different levels of replication. This approach meets all the requirements we have defined. Requirements R1 to R4 are met in the same way as in the previous approach, as the components are the same. Nevertheless, in this approach R5 is now met because it supports the full customisation of the devices and their level of replication.

Therefore, even though we could explore other design options, the approaches we have designed cover most or all of our requirements. Thus, exploring other designs and their interest is left as future work, e.g. a per-stream replication scheme where only end-systems replicate.

6.2 Fault types and failure modes

In order to properly design any fault tolerance mechanism, we must first describe the fault types and failure modes that are expected for the devices, in this case, the links. Concretely, we need to specify the types of faults that we want to tolerate with our solution. In this work, our fault model covers *temporary non-malicious hardware faults* (Avižienis et al., 2004). That is, faults that last for a finite amount of time, that are caused in an involuntary manner and that only affect the hardware, e.g. electromagnetic interference. We have decided to tolerate these faults as this is the type of faults most commonly considered when designing fault-tolerant communication systems.

On the other hand, the failure mode defines how a device may behave in the presence of faults. We assume that links exhibit omission failure semantics, i.e. whenever a fault affects a frame that is being transmitted through a link, the frame becomes erroneous and it is omitted by the system. This is a reasonable assumption since Ethernet frames convey a CRC code which allows detecting virtually all errors in the frames upon reception (Fujiwara, Kasami, and Lin, 1989). Erroneous frames are dropped upon reception, manifesting as omissions and thereby preventing the propagation of erroneous information.

6.3 PTRF operation

As we have already explained, we designed three different approaches to the PTRF mechanism, which differ from each other in the devices that carry the replication out and in the way replicas are handled. We introduced the first two approaches in (Álvarez et al., 2017) and the third one in (Álvarez et al., 2019). Furthermore, we described each approach and we discuss their advantages and disadvantages in (Álvarez et al., 2021).

6.3.1 Approach A: End-to-end estimation and replication

In this first approach, only end-systems implement PTRF, i.e. end-systems replicate frames during transmission and eliminate surplus replicas upon reception. Bridges are COTS standard TSN bridges that simply forward all the correct frames that they receive. The number of replicas k that end-systems must transmit is decided for the network as a whole using an end-to-end worst-case estimation of frame omission probability. We refer to this approach as approach A in the rest of the dissertation.

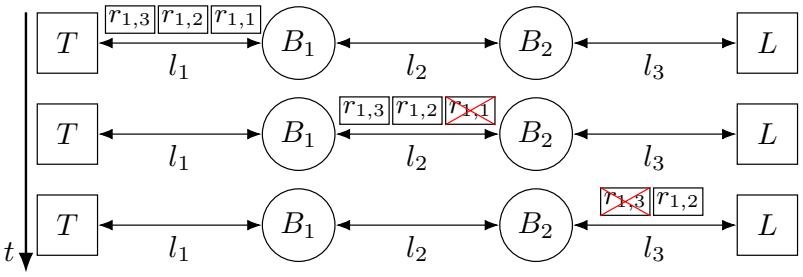


Figure 6.1: Behaviour of the approach A of the PTRF mechanism in the presence of temporary faults in the links. (Source (Álvarez et al., 2021))

Figure 6.1 depicts the behaviour of a network that uses approach A to tolerate temporary faults in the links. The network has one talker (T), one listener (L) and two bridges (B_1 and B_2), connected by three links (l_1, l_2, l_3) forming a line topology. In this figure, the value of k is 3 and, thus T transmits three replicas of frame f_1 ($r_{1,1}$, $r_{1,2}$ and $r_{1,3}$). As we can see, B_1 receives and forwards the three replicas of f_1, nevertheless, replica $r_{1,1}$ is affected by a temporary fault in link l_2, causing its omission. Thus, B_2 only forwards the correctly received replicas, $r_{1,2}$ and $r_{1,3}$. After that, a temporary fault affects replica $r_{1,3}$ in link l_3, which is dropped by L upon

reception. $r_{1,2}$ reaches L correctly and it is delivered to the application.

6.3.2 Approach B: End-to-end estimation, link-based replication

In this approach, both end-systems and bridges implement PTRF, i.e. all network devices replicate frames during transmission and eliminate surplus replicas upon reception. We refer to this approach as approach B in the rest of the article.

As in approach A, the number of replicas k' is decided for the network as a whole using an end-to-end worst-case estimation of frame omission probability. Nevertheless, we must note that the value of k and k' may not be equal for the same network. This is because in approach B all bridges generate a new set of replicas and, thus, the transmission is successful even if $k' - 1$ replicas are omitted in each link, whereas in approach A only $k - 1$ replicas can be lost for the path as a whole. This has a great impact on the number of fault scenarios that can be tolerated by each approach, as we will discuss in the next chapter.

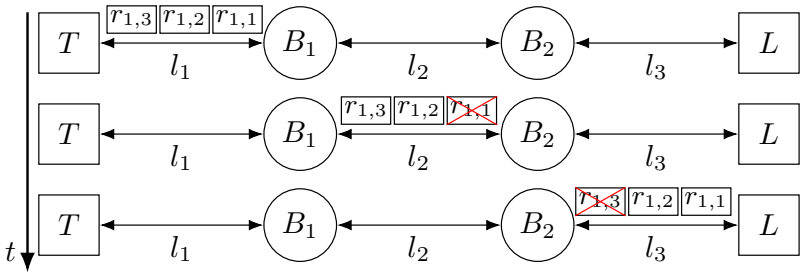


Figure 6.2: Behaviour of the approach B of the PTRF mechanism in the presence of temporary faults in the links.

Figure 6.2 depicts the operation of a network that uses approach B to tolerate faults in the links. The network topology and the faults are the same as in Figure 6.1, but we see that the network behaves differently. Specifically, we can see that, even though $r_{1,1}$ is lost in link l_2, B_2 transmits 3 new replicas through link l_3. This is because bridges only keep the first correct replica they receive and discard the surplus ones, and, then, they create a new set of replicas during transmission. Again, replica $r_{1,3}$ is affected by a fault in link l_3, but L receives two replicas now, instead of one. Thus, L receives replica $r_{1,1}$ first, and delivers it to the application and discards replica $r_{1,2}$.

6.3.3 Approach C: Link-based estimation and replication

In this approach, both end-systems and bridges implement PTRF, i.e. all the devices in the network replicate frames during transmission and eliminate surplus replicas upon reception, as in approach B. Nonetheless, the number of replicas that must be transmitted through each link m can vary depending on the loss probability of said link. Therefore, each device may transmit a different number of replicas k''_m through each link. We refer to this approach as approach C in the rest of the article.

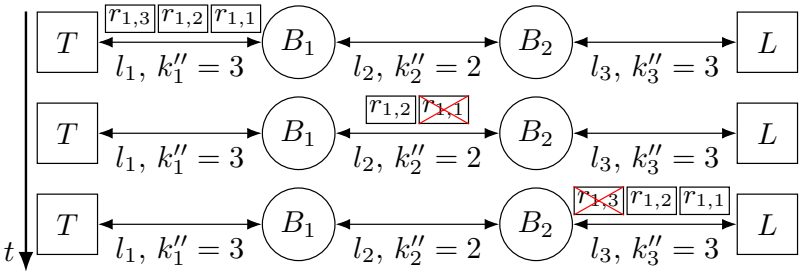


Figure 6.3: Behaviour of the approach C of the PTRF mechanism in the presence of temporary faults in the links.

Figure 6.3 depicts the behaviour of a network that uses approach C to tolerate temporary faults in the links. The scenario shown in Figure 6.3 is the same as in Figures 6.1 and 6.2, but we can see that the number of replicas transmitted by the devices varies. T transmits 3 replicas through link l_1, but we see that, even though all replicas are correctly received by B_1, it only transmits two replicas through link l_2, as it is configured to do so. Again, replica $r_{1,1}$ is affected by a fault in link l_2 and, thus, dropped by B_2. Nevertheless, B_2 transmits 3 replicas again and, even though replica $r_{1,3}$ is affected by a fault, L receives replicas $r_{1,1}$ and $r_{1,2}$, it delivers $r_{1,1}$ to the application and discards $r_{1,2}$.

6.3.4 Qualitative comparison of the approaches

Each approach presents a series of advantages and disadvantages. On the one hand, approach A allows using COTS TSN bridges, which can significantly reduce the cost of the network. On the other hand, the number of fault scenarios that can be tolerated before losing information is low compared to the other two approaches. We call a fault scenario to any of the possible

combinations of faults that cause the omission of a subset of replicas. Let us assume we have a talker that transmits frame f_1 with $k = 2$, resulting in the transmission of replicas $r_{1,1}$ and $r_{1,2}$. In this case, the number of fault scenarios that can be tolerated by approach A is two for the whole path, the omission of $r_{1,1}$ or the omission of $r_{1,2}$, as any other scenario would cause the omission of both $r_{1,1}$ and $r_{1,2}$. Nevertheless, when using approach B the number of scenarios that can be tolerated increases with the number of links. This is so as approach B can tolerate the loss of any of the replicas in each link, as the following device creates a complete set of replicas again. We provide more details on how to calculate the number of scenarios that can be tolerated by each approach in the next chapter.

Furthermore, in approach A the schedule must be adapted to take into account that there might be interfering frames between the transmission of replicas by the bridges. Figure 6.4 depicts a scenario in which there is interleaving of replicas when using approach A. In this example we see a network with two Talkers (T_1 and T_2), one bridge (B_1) and a Listener (L), connected through 3 links (l_1, l_2 and l_3). Each talker transmits a frame, f_1 and f_2 respectively, which are replicated twice, resulting in replicas $r_{1,1}$ and $r_{1,2}$ for frame f_1 and replicas $r_{2,1}$ and $r_{2,2}$ for frame f_2. Let us assume that f_1 and f_2 have the same priority and, thus, are queued in the same transmission buffer of bridge B_1. Let us also assume that replica $r_{1,1}$ reaches bridge B_1 and is forwarded to the egress port. Then, replica $r_{2,1}$ reaches bridge B_1 next and is forwarded to the egress port before replica $r_{1,2}$. As we can see, this situation can easily happen when frames with the same priority are transmitted through different network paths. Therefore, if this is not properly reflected in the scheduling the last replicas of a frame may violate the deadlines calculated for the first one.

Regarding approach B, it allows tolerating a significantly higher number of fault scenarios than approach A when $k' = k$, as we proved in (Álvarez et al., 2019). Moreover, we must note that the design of this approach prevents interleaving of replicas, as replicas are created in the egress port and they are immediately queued in the transmission buffer. Thus, all the replicas of a frame are created and queued before processing the following frame. This enables the use of existing schedulers which now only need to take into account the time required to create and transmit k' replicas instead of a single frame.

Nevertheless, in approach B, all the devices of the network must be modified to support PTRF, which can lead to an increase in the price of the

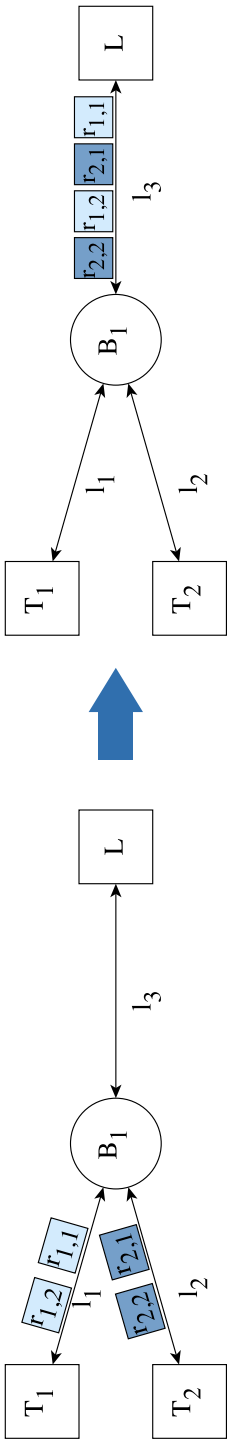


Figure 6.4: Example of interleaving replicas when using approach A of the PTRF mechanism to tolerate temporary faults in the links.

network. Furthermore, approach B can lead to an inefficient use of the network bandwidth, as the number of replicas to be transmitted in each link is calculated using a worst-case estimation for the whole network. Therefore, even if the omission probability is low in certain parts of the network the level of replication must be high enough to tolerate the faults in the areas with harsher conditions.

Approach C is a compromise between approaches A and B. Like in approach B, bridges must be modified to support PTRF and it can tolerate a significant amount of fault scenarios. But, contrary to approach B, approach C allows adapting the number of replicas transmitted according to the vulnerability of each link, thus reducing the bandwidth consumed to tolerate faults while guaranteeing an adequate level of fault tolerance. Furthermore, approach C prevents interleaving of frames between replicas as the replication mechanism is the same as in approach B.

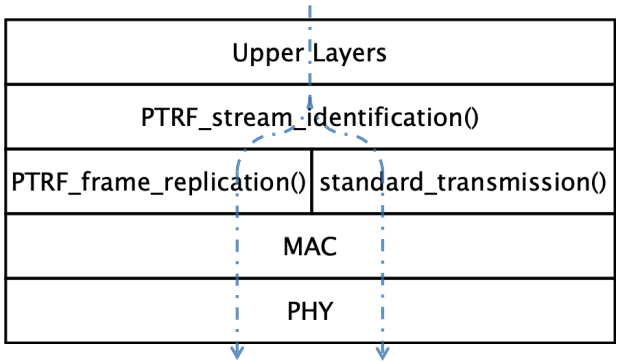
Finally, this approach is specially suitable to support dynamic fault tolerance and to integrate temporal and space redundancy (Álvarez, Proenza, and Barranco, 2018) in the same network. On the one hand, flexibility is key for adaptive systems and being able to adapt the level of replication depending on the vulnerability of the link provides a high level of flexibility for fault tolerance. On the other hand, in order to combine spatial and time redundancy in an efficient manner, it is important to adjust the level of time redundancy to the availability of space redundancy in the network. That is, whenever space redundancy is available, time redundancy can be deactivated to save resources.

6.4 PTRF design

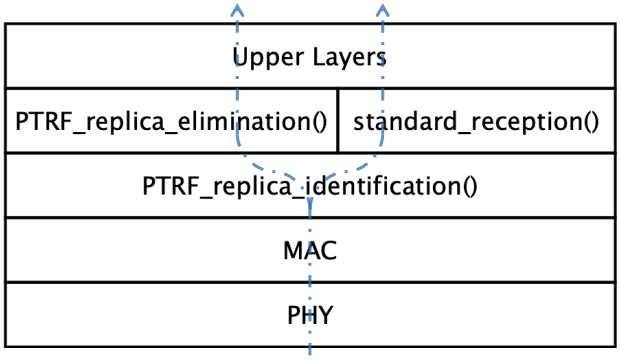
In order to support the PTRF mechanism, devices and frames must undergo a series of modifications. We have classified these modifications into three groups, namely (i) functions, (ii) frames and (iii) components. We next describe these modifications in detail.

6.4.1 Functions

We have divided the operation of PTRF into functions, which are shown in Figure 6.5. Specifically, Figure 6.5a shows the functions needed for the transmission of replicas and Figure 6.5b shows the functions needed to support the reception of replicas.



(a) PTRF functions during transmission.



(b) PTRF functions during reception.

Figure 6.5: Functions of the PTRF mechanism in reception and transmission.

We can divide PTRF in two functions during transmission.

- PTRF_stream_identification: this function is responsible for differentiating frames that must be replicated from those that must not. This is done by checking the priority of the frame, as it will be detailed in Section 6.4.3. If the frame must be replicated, the function PTRF_frame_replication is executed, otherwise, the standard transmission process is executed.

- PTRF_frame_replication: this function carries out the replication of frames. First, this function embeds in the frame some information which is important for the correct operation of PTRF, which we detail in section 6.4.2. Then, the frame is replicated the required amount of times, depending on its priority, and each replica is queued for

transmission.

We can divide PTRF into two functions during reception.

- PTRF_replica_identification: whenever a frame is received through a port that implements PTRF, this function detects whether the frame is a replica or not. This is done by looking for the PTRF Ethertype, which is introduced in the frame by function PTRF_frame_replication, as indicated in Section 6.4.2. If the frame conveys such Ethertype, the PTRF_replica_elimination function in executed. Otherwise, the standard reception process is chosen.

- PTRF_replica_elimination: this function is responsible for eliminating surplus replicas. To do so, the function checks whether this is the first replica of a given frame that has been received or not. To this aim, each port counts with a database that stores key information to identify the last received replicas, as we further explain in Section 6.4.3. If this is in fact the first received replica of a given frame, the database is updated and the replica is delivered to the application. Otherwise, the replica is discarded.

6.4.2 Frames

As we have already mentioned, the PTRF mechanism requires the addition of information to frame replicas. Specifically, we add three new fields in each replica. This additional information is conveyed in every replica, but it is not required for the transmission of non-replicated frames. Moreover, the structure of PTRF replicas is compatible with any TSN or standard Ethernet bridge, as we explain later on. Figure 6.6 shows the structure of replicas and we describe it next.

- PTRF Ethertype: This field allows differentiating replicas created by the PTRF mechanism from those frames that are not replicated. This field occupies 2 bytes and its value is 0x8815. It is important to note that if the original frame conveys an Ethertype to identify a higher layer protocol, said Ethertype is not deleted, only shifted to the Payload length/Ethertype field so the replica can be properly processed upon reception.

field	Destination MAC @	Source MAC @	C-tag Ethertype	Priority, DE, VLAN ID	PTRF Ethertype	PTRF Frame Identifier	Expected num replicas	Payload length /Ethertype	data	CRC
bytes	6	6	2	2	2	2	1	2	n	4

Figure 6.6: Format of an IEEE 802.1Q Ethernet Data frame that has been replicated using PTRF. As we can see highlighted in blue, the frame conveys new fields, namely the PTRF Ethertype, PTRF frame identifier and the expected number of replicas.

- PTRF frame identifier: This field allows distinguishing replicas that do not convey the same information. As we have explained previously, the frames transmitted through a single stream convey information from the same source produced in different moments, t_0, t_1, t_2, etc and we call these message editions. Therefore, to identify replicas we must distinguish whether they belong to the same message edition or not. Nevertheless, the control information of TSN frames is exactly the same for all the frames transmitted through a stream, i.e. frames that convey different message editions carry the exact same control information, only the payload is different. Therefore, we add this field to distinguish the replicas of different message editions, i.e. the replicas transmitted in t_0 from the ones transmitted in t_1, the ones transmitted in t_1 from those transmitted in t_2, etc. This field is 2 bytes long, allowing to transmit 65536 distinct editions of a message through the same stream before having to reset it.

- Expected number of replicas: This field contains the number of replicas k, k' on k''_m that must be transmitted and, thus, expected to be received in a fault-free scenario. This field is 1 byte long, allowing for a maximum of 255 replicas per frame.

With this design, all the replicas of a given frame convey exactly the same information. Let us assume that we have a stream S with a stream identifier st through which we want to transmit n frames, $f_{st,1}$, $f_{st,2}$ ... $f_{st,n}$. Let us also assume that the frames transmitted through stream st must be replicated m times. The PTRF-related information conveyed by each replica $r_{st,id,j}$ of a frame $f_{st,id}$ will be the PTRF Ethertype 0x8815, the PTRF frame identifier id and the expected number of replicas m. Furthermore, all TSN frames convey the stream identifier st. Note that in approach C the number of replicas transmitted by each bridge via each link may vary. Therefore, the field called expected amount of replicas must be updated in each egress port of each bridge.

This structure of PTRF frames is fully compatible with standard TSN or Ethernet bridges. Whenever a standard bridge receives a PTRF replica, it checks the PTRF Ethertype and it interprets the field as a higher layer protocol. Therefore, the bridge treats the frame as it would treat any other regular frame, which allows us to use COTS bridges in approach A.

6.4.3 Components

To enable the use of PTRF, the devices involved in the replication of frames and the elimination of replicas must include a series of new components. We must keep in mind during the next discussion that in approach A only end-systems implement PTRF, while bridges are standard COTS Ethernet bridges, and thus do not undergo any changes. On the other hand, in approaches B and C all end-systems and bridges implement PTRF. We next describe the new required components and we detail their deployment according to each approach.

- PTRF replication selection table: As we explain in Section 6.4.1, whenever a frame is transmitted, PTRF decides whether the frame must be replicated or not. This is done by consulting the PTRF replication selection table, which has two columns, (i) a list of all priorities, from 0 to 7 and (ii) the number of replicas to be transmitted for each priority. If the number of replicas for a given priority is 0, the PTRF mechanism is disabled for the streams of said priority. Therefore, frames of those streams are transmitted using the TSN standard transmission and frame format.

  In approach A there is a single table in each end-system. In approach B there is a single table in each end-system and bridge. In approach C there must be a different table in each eggress port of each end-system and bridge, to support different levels of replication for each link.

- Replica creation counter: This counter allows tracking the number of replicas of a given frame that have been created and queued. There must be one counter in each egress port of each PTRF-enabled device, in each approach. When the counter reaches the number of replicas k, k' or k''_m, it is reset and the same counter can be used to count the replicas of a new frame.

- PTRF replica identification table: This table stores the information of the last replica received through a stream by a PTRF-enabled device. This information is used to eliminate surplus replicas upon reception. Figure 6.7 shows the operation of this table. Whenever a PTRF-enabled device receives a replica, it must check whether this is the first received replica of a given message edition or not. To that, the device looks for the stream identifier up, to know to which stream the replica belongs to (stream 002 in the figure). Then, the device compares the PTRF frame identifier of the replica to the one stored in the table. If the

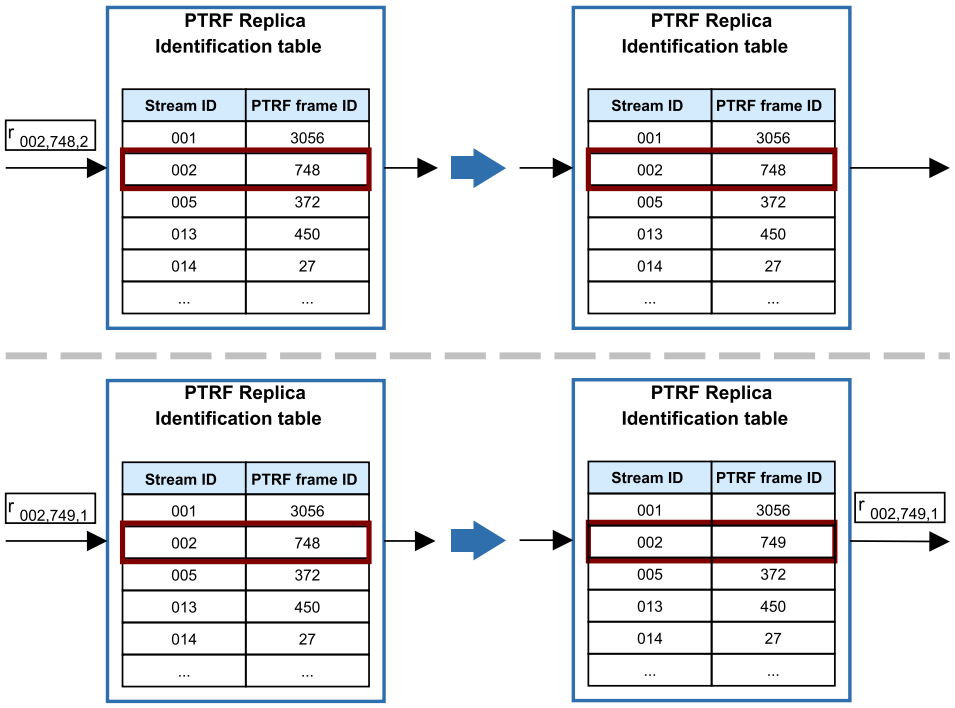


Figure 6.7: This figure shows the operation of the PTRF Replica Identification table. Specifically, on the top of the Figure, we see how the table processes a replica of a message edition that has previously been received. On the bottom, we see how the table processes a replica of a message edition that is received for the first time.

stream identifier and the PTRF frame identifier in the replica are the same as the ones stored in the table, it means that this is not the first replica of that message edition received and, therefore, it can be discarded, as we see in the top of Figure 6.7. Otherwise, if the PTRF frame identifier of the table is different from the one in the replica, then this is the first replica that is received from the message edition. Thus, the table is updated with the new PTRF frame identifier and the replica is forwarded or delivered to the application, as we see in the bottom of Figure 6.7. We must note that we take advantage of the fact that bridges must store the stream identifier of all the streams that they forward to simply add the two-byte PTRF frame identifier.

- Replica reception counter: This counter is not used to support the operation of PTRF. Instead, this counter is devised to support the

dynamic adaptation of the mechanism in the future. There must be one counter in each input port of each PTRF-enabled device, regardless of the approach. Specifically, the counter is increased each time a replica of a given frame is received. Thus, when compared to the expected amount of replicas k, k' or k''_m this counter provides information about losses. The counter is reset whenever a new replica of the same or of a different frame is received. This information can be gathered by a management entity, such as the IEEE 802.1 Std Qcc CNC (802.1, 2018), to create new network configurations that properly adapt to the needs of the system.

Chapter 7

Validation through Simulation of PTRF

In order to prove our thesis we must ensure that the TSN networks that implement PTRF keep the real-time and operational flexibility provided by TSN. On top of that, to take full advantage of TSN as a standard, we must guarantee that PTRF devices can coexist with TSN COTS devices in a transparent manner.

To that aim, we validate PTRF through simulation as a previous step to its implementation in a real prototype. Specifically, this simulation allows us to study the feasibility of our design and to assess its correctness. 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 scenarios are tolerated by each approach of PTRF.

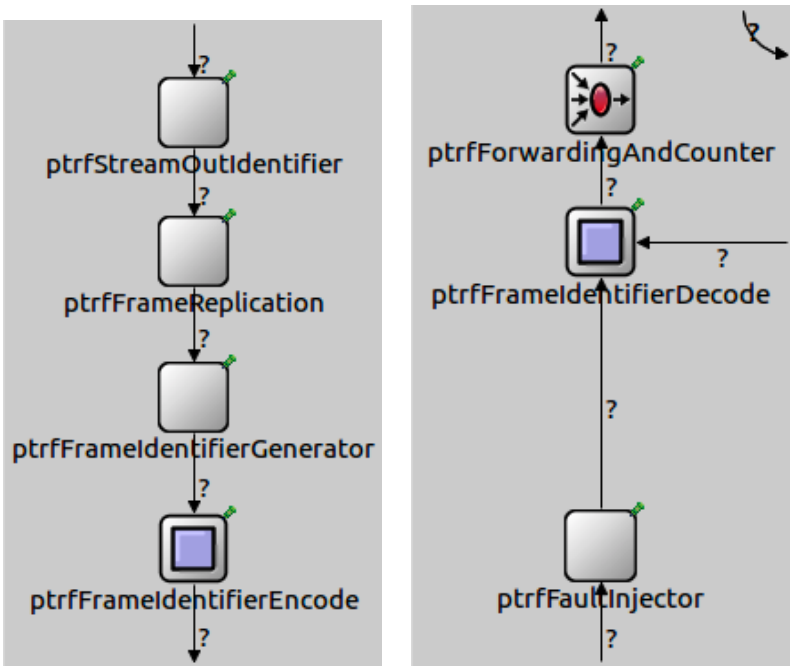
On top of that, we carry out a mathematical analysis of the fault scenarios that can be tolerated by each approach. This analyses allows us to assess the correctness of the results obtained using simulation.

7.1 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++ (Varga, 2001), a modular event-based simulation framework to model distributed systems and networks in

C++. Furthermore, OMNeT++ counts with the INET library ("The INET Framework—An Open-Source OMNeT++ Model Suite for Wired, Wireless and Mobile Networks."), 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 (Heise, Geyer, and Obermaisser, 2016), 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 (Heise, Geyer, and Obermaisser, 2016). Note that in this work we did not use space redundancy, as we focus on the validation of PTRF and including another redundancy mechanism could mask its behaviour.



(a) PTRF OMNeT++ module for transmission.

(b) PTRF OMNeT++ module for reception.

Figure 7.1: PTRF OMNeT++ modules.

Figure 7.1 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 also modified the TSN frame provided by TSimNet to convey the PTRF information described in the previous chapter.

Figure 7.1a 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[1] and sends it out.

Figure 7.1b 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.

7.2 Fault Scenarios Analyses

We must recall that we want to 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. We start by describing how we calculate the combinations of fault scenarios in a single link, and we then move to the analyses of the approaches.

[1]Note that in this context the word message refers to the message structure used in OMNeT++, not to the message transmitted by an application

7.2.1 Fault Scenarios in a single Link

As we have explained, in order for faults to be tolerated at least one replica must reach the destination. Therefore, when transmitting k replicas through a link, PTRF can only tolerate up to $k-1$ faults, i.e. we can only lose up to $k-1$ replicas. We must note that the order in which frames are affected by faults is irrelevant for PTRF as the result is the same, i.e. all the frames affected by faults are dropped upon reception, regardless of the order in which faults happen. Furthermore, a replica can only be affected by a fault once in a link, as we do not consider near-coincident faults in our analysis. Therefore, the number of scenarios corresponds to a combination without repetition.

Equation 7.1 shows how to calculate the number of different scenarios in which we can lose up to $k-1$ in a link:

$$\sum_{e=0}^{k-1} \binom{k}{e}, \tag{7.1}$$

where k is the number of replicas and e is the number of erroneous frames, i.e. frames affected by faults. e can be 0 as we count the scenario with no faults as a successful scenario for PTRF.

We next describe how this propagates in a multi-hop network that implements each one of the approaches.

7.2.2 Approach A

As we know, 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. This means that the combinations of fault scenarios in each link must account for the faults occurred in previous links. Equation 7.2 shows how to account for the errors occurred in the first link when we have two links:

$$\sum_{e_1=0}^{k-1} \left(\binom{k}{e_1} \cdot \sum_{e_2=0}^{k-e_1-1} \binom{k-e_1}{e_2} \right), \tag{7.2}$$

k is the number of replicas transmitted by the talker, e_1 is the number of errors in the first link and e_2 is the number of errors in the second link.

As we have said, only the frames that have not been affected by an error in the first link are forwarded through the second link, that is $k - e_1$ replicas. Moreover, since we need to guarantee that at least one replica reaches the destination, the maximum number of errors that can be tolerated in the second link is $k - e_1 - 1$. Thus, if we lose $k - 1$ frames in the first link, the maximum number of tolerated errors in the second one is $k - (k - 1) - 1 = 0$.

We now want to generalize Equation 7.2 to any number of links. To that, we use the distributive property of the summation. The distributive property tell us that:

$$a \cdot \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} a \cdot x_i \tag{7.3}$$

Therefore, if we apply the distributive property to 7.2 we get:

$$\sum_{e_1=0}^{k-1} \sum_{e_2=0}^{k-e_1-1} \binom{k}{e_1} \binom{k-e_1}{e_2} \tag{7.4}$$

We can generalize this as showed in Equation 7.5, which provides us with the number of fault scenarios that approach A can tolerate in a network with l links:

$$\sum_{e_1=0}^{k-1} \cdots \sum_{e_l=0}^{(k-e_1-\cdots e_{l-1})-1} \left(\prod_{m=1}^{l} \binom{k - \sum_{i=1}^{m-1} e_i}{e_m} \right) \tag{7.5}$$

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.

7.2.3 Approach B

As we already know, 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 each bridge. Thus, approach B can tolerate all fault scenarios where up to $k' - 1$ replicas are lost in each link. Therefore, each fault scenario of a link must be in turn combined with all the scenarios in the rest of the links.

This is expressed in Equation 7.6:

$$\left(\sum_{e=0}^{k'-1}\binom{k'}{e}\right)\cdot\left(\sum_{e=0}^{k'-1}\binom{k'}{e}\right)\ldots \tag{7.6}$$

where k' is the number of replicas transmitted by each component and e' is the number of faults that happen in each link. This can be generalised as shown in Equation 7.7, which calculates the number of scenarios that approach B can tolerate in a network with l links:

$$\left(\sum_{e'=0}^{k'-1}\binom{k'}{e'}\right)^{l} \tag{7.7}$$

7.2.4 Approach C

Approach C follows the same replication scheme as approach B, in the sense that receiving one replica is enough for a bridge to generate a new subset of replicas. The difference is that end-stations and bridges may send a different number of replicas k''_m through each link m. Therefore, approach C can tolerate all fault scenarios where up to $k''_m - 1$ replicas are lost in each link. Since the scenarios in each link are combined in the same way as in approach B, we move directly to the generalisation of the formula. Equation 7.8 shows the number of scenarios that approach C can tolerate in a network with l links:

$$\prod_{m=1}^{l}\sum_{e''=0}^{k''_m-1}\binom{k''_m}{e''} \tag{7.8}$$

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.

7.3 Simulation Results

We used the model described in Section 7.1 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 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 (802.1, 2020b), 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 7.1 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.

Table 7.1: 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 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.

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 we present in chapter 9.

Chapter 8

Implementation and Experimental Evaluation of PTRF

As we have stated in Chapter 7, in order to prove our thesis statement we need to ensure that PTRF properly operates in TSN networks without interfering with the real-time and operational flexibility that the standards provide. On top of that, we must ensure that this operation is transparent for COTS TSN devices if we want to take full advantage of the standards.

Chapter 7 presented a first assessment of the adequacy of PTRF to operate in TSN networks. Nonetheless, simulation is not enough. To actually prove that PTRF does not interfere with the real-time guarantees of the network, as well as to ensure the correct integration with TSN an implementation on a real prototype is required.

In this Chapter we present the implementation of PTRF on a commercial TSN bridge of the company SoC-e. On top of that, we also describe the implementation of a custom fault injection device that we have designed to evaluate the PTRF mechanism in the presence of faults. After that, we present the experiments we carry out and we discuss the results that we have obtained and how they help us reach our thesis statement.

8.1 PTRF Implementation

In this section we describe the implementation of a PTRF-enabled device which can be used as an end-system and as a bridge indistinctly. Furthermore, we describe the implementation of a fault injection device which has been designed and developed to support the validation and evaluation of the PTRF mechanism.

8.1.1 The PTRF-enabled device

The PTRF mechanism has been implemented on an already existing TSN bridge developed by the company System on Chip (SoC-e) (*System on Chip & FPGA IP Core Development.*). Specifically, the mechanism is currently implemented as a firmware that operates on the Multiport TSN (MTSN) Kit (*MTSN Kit: a Comprehensive Multiport TSN Setup.*) developed by the company. The MTSN Kits used for this work had a version of the firmware (SoC-e continously upgrades this firmware) which implemented a series of TSN standards fully or partially, namely IEEE Std 802.1AS , IEEE Std 802.1Qav-2009, IEEE Std 802.1Qbv-2015 and IEEE Std 802.1Qcc-2018.

The PTRF mechanism is implemented on a Zynq UltraScale+ MPSoC (*Zynq UltraScale+ MPSoC.*). This MPSoC can be roughly divided between a processing system and a programmable logic. In our experiments, the processing system executes the end-system application and it is in charge of producing the messages to be transmitted; while the programmable logic implements the MTSN switch IP with PTRF support. This allows us to use the MPSoC both as an end-system and as a bridge, both compatible with PTRF. Moreover, the implementation allows us to disable PTRF and use the MPSoCs as standard TSN devices.

Figure 8.1 depicts the architecture of the MPSoC with the Linux OS implemented in the processing system and the MTSN switch IP implemented in the programmable logic. The PTRF components described in Section 6.4.3 are highlighted in blue. Solid lines represent the path followed by the frames, whereas dashed lines represent management or configuration interactions. As we have mentioned, the MPSoC can be configured to operate as an end-system or a bridge.

When the MPSoC acts as an end-system, the application is executed in the Linux OS and the communication between the processing system and the programmable logic is activated. Messages created by the application are

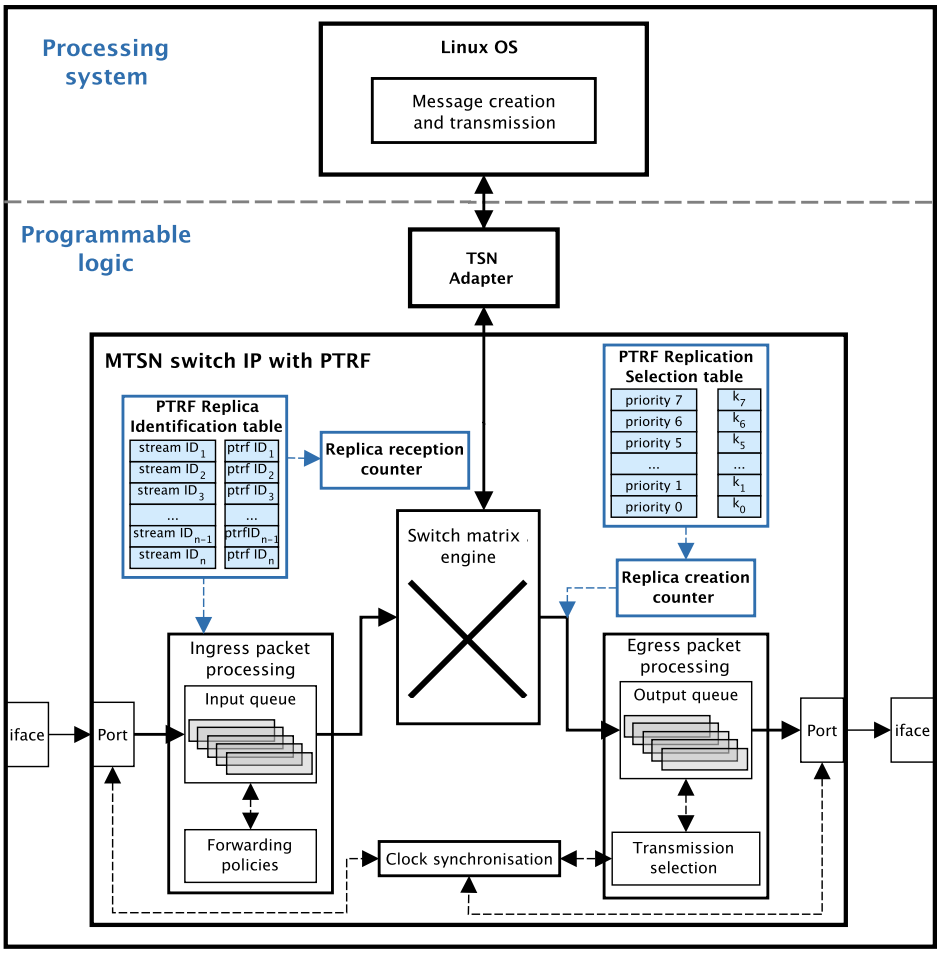


Figure 8.1: Basic architecture of the MPSoC that implements the PTRF mechanism. The PTRF components are highlighted in blue. Solid lines represent the path which frames follow whereas dashed lines represent management or configuration interactions.

sent to the TSN adapter, which is implemented in the programmable logic and acts as a TSN network interface. Afterwards, the frame is forwarded to the adequate egress port using the switching logic and it is processed and transmitted as we describe next. On the other hand, when a frame is received through any of the ingress ports of the switch, the frame is passed to the application.

When the MPSoC acts as a bridge, the processing system is disabled and

only the MTSN switch IP is active. Whenever a frame is received through a port, the PTRF replica identification table is consulted in order to detect replicas and decide whether it must be discarded or forwarded. If the replica must be forwarded, the bridge consults the forwarding policies and the replica is forwarded to the designated egress port.

Once a replica is forwarded to the egress port and before it is stored in the output queue, the PTRF replication selection table is consulted to decide whether the frame must be replicated or not. If the frame must be replicated, all replicas are created and stored in the corresponding output queue in a consecutive manner. No frames can be processed while the replicas are created, thus guaranteeing that all replicas are stored and transmitted subsequently. Once the transmission selection algorithm selects the corresponding queue, the replicas are transmitted.

Finally, if PTRF is deactivated in the MPSoC, the programmable logic simply ignores the PTRF components and logic. Therefore, the MPSoC acts as a standard end-system or bridge, depending on the configuration. This is useful when implementing approach A, as only end-systems must be PTRF-devices, while bridges must be standard TSN.

8.1.2 The fault injection device

As we have already explained, the PTRF mechanism is designed to tolerate temporary faults in the links. Therefore, to properly validate and evaluate PTRF we must study its behaviour in the presence of faults. We rely on prototype-based fault injection at the software level to carry out our evaluation (Mei-Chen Hsueh, Tsai, and Iyer, 1997). This technique consists in using fault injection in a real prototype of the system to study its behaviour in the presence of faults. To do that, we use a device specifically designed to corrupt the desired frames (whether they are replicated or not) while they are being transmitted through the links in such a way that the receiver drops them thanks to the frames' CRC.

This device has been developed by SoC-e on a Rely-RB time-aware redbox switch and it allows inspecting frames on-the-fly to (i) corrupt frames that match a predefined pattern, (ii) timestamp frames during their transmission through a link and (iii) measure the bandwidth consumption of a link in real time. This device has been developed ad hoc by the company and, thus, there is no public documentation or specification.

The Rely-RB is connected to the network as depicted in Figure 1.1a

(LabTool). Basically, we place the Rely-RB between two devices (end-systems or bridges) that exchange information. Specifically, we connect one of the devices to port 0 of the Rely-RB and the other one to port 1. The Rely-RB carries out the designated operations (corrupt frames, timestamp or metering) on the frames that arrive through port 0 and forwards them through port 1. All these operations are done on-the-fly with nanosecond resolution and the overhead introduced by the device is negligible. In this way, the Rely-RB is transparent to the devices in the network.

Furthermore, the Rely-RB allows inspecting two different links simultaneously, that is, frames that arrive through port 2 are processed and forwarded through port 3. This is useful in our experiments as all the interfaces share a common clock. Therefore, we can use the Rely-RB as a common time reference to measure the ed-to-end delay in our experiments.

We next describe the setups used to evaluate the PTRF mechanism, the experiments done and the results obtained.

8.2 Experimental Setup Characterisation

In this section we describe the hardware and software setup we use to carry out the experiments described in Section 8.3. In these experiments, we use up to four MTSN switches, two fault injection devices and a PC. One MTSN switch acts as an end-system that generates frames, while the rest of the MTSN switches act as pure TSN or PTRF-enabled TSN bridges. The fault injection devices are used to record the frames' transmission or reception instants and to corrupt particular frames. We use the recorded timestamps to measure the end-to-end delay and jitter as described in Section 8.3. Finally, the PC acts as an interface to control all the other devices and it also captures the received frames with Wireshark (*Wireshark.*).

It is important to note that in all of our experiments we use a line topology with a single path. This is so because we study the impact of time redundancy in the absence of space redundancy. Moreover, Ethernet eliminates loops in the network, creating a logical line topology even in mesh networks. On top of that, we only use a single transmitter and a single receiver, each of them placed in a different end of the line topology to maximise the number of hops between transmission and reception. Finally, even though TSN provides timing guarantees for up to six hops, we limit our experiments to four hops due to hardware restrictions.

Regarding the traffic, we transmit a single stream because we want to study

the impact of PTRF and using interfering traffic could mask the impact of the mechanism and complicate the analysis unnecessarily. On the other hand, we carry out our experiments using TT traffic with priority 5. We do this because, many of the novel applications supported by TSN, such as Industry 4.0, autonomous driving or energy management; strongly rely on TT traffic. Furthermore, TT traffic is usually subject to the highest timing and reliability requirements. Thus, even though PTRF can be used to replicate other types of traffic, we have decided to evaluate the mechanism using the most demanding type.

Finally, we use a 100Mbps network with different TAS configurations, to support different schedules for the TT traffic. We must note that we only use TAS as it is the only mechanism needed to transmit TT traffic. Furthermore, we use three different frame lengths (64, 782 and 1500 bytes) to be able to observe the impact of frame processing in the bridges. We use two, three and four replicas to study how the transmission of replicas impacts on the communications. We only use up to four replicas as previous works have shown this to be a sufficiently high number of replicas in other Ethernet-based reliable networks (Barranco, Derasevic, and Proenza, 2020). We run each experiment 1000 times, as we consider this to be a sufficiently large sample. We next describe the different experiments in detail and we discuss their results.

8.3 PTRF evaluation and results

As any other fault tolerance mechanism, PTRF has an impact on the network performance. This is because the transmission of replicas implies an increase in the resources required to exchange a single message, e.g. time or bandwidth. As we have already mentioned, we have evaluated PTRF from three different points of view, namely (i) impact on the end-to-end delay, (ii) introduction of jitter in the transmission and (iii) impact on the bandwidth consumption. We carry out all of our measurements in the layer 2 of the network architecture for two reasons. First, all the mechanisms proposed by the TSN TG operate at the layer 2 and PTRF is also designed to operate in such layer. Second, we carry out a sensitivity analysis to study the behaviour of PTRF when operating in networks with different characteristics. Therefore, our study must be application independent and, thus, we abstract the operation of the application.

It is important to note that we do not measure the impact of using

time redundancy in a specific application. Instead, we study the overhead introduced by the devices that implement PTRF and we carry out a sensitivity analysis. This is so because the actual impact on the temporal behaviour and the bandwidth depends on the redundancy level and the network utilization of each application. At any rate, PTRF is only to be used in systems that require a certain level of reliability, even if that means reducing the response time or the bandwidth available for non-critical traffic.

We have carried out each of the following experiments for 3 different types of networks: a standard TSN network, a network implementing approach A and a network implementing approach B. As the implementation of approaches B and C is the same, we do not carry out any specific experiments to measure the difference between these approaches. Furthermore, a study on the gains of using approach C over approach B in the networks of adaptive systems is out of the scope of this dissertation and is left as future work.

It is important to stress the fact that these experiments have been executed by Ignasi Furió, *Profesor Titular* of the University of the Balearic Islands. The author of the present dissertation has designed all the experiments, as well as she has analysed all the results. On top of that, she has taken special care to guarantee that the experiments were properly executed.

We next describe the experiments done and the results obtained.

8.3.1 End-to-end delay

We carry out two different sets of experiments to study the impact of PTRF on the end-to-end delay. The target of the first set of experiments is to measure the overhead that using PTRF causes on the end-to-end delay of a single frame in the absence of faults. To that, we transmit a single frame in each one of the networks with no replication, we measure the end-to-end delay and we compare the results obtained with approaches A and B to those obtained with standard TSN to measure the overhead.

We must note that we define the end-to-end delay of a set of replicas as the time elapsed since the first replica starts being transmitted until the first correct replica reaches the receiver. Since in this set of experiments we transmit a single frame in the absence of faults, the end-to-end delay is the time elapsed since the frame is transmitted until it is received.

Figure 8.2 shows the topology. The first MTSN switch (Sw 1) acts as an end-system and transmits the frames, which are forwarded on the other three MTSN switches until they are captured with Wireshark on the right-hand PC.

The fault injection device (LabTool) is used to record the frames' timestamps. The difference between the time recorded on port 3, when the frame reaches the PC, and port 0, when the frame leaves the end-system, is the end-to-end delay of the frames. TAS is configured in each device to allow all priority 5 TT traffic pass and block the rest of the traffic, preventing interference. Furthermore, frames are transmitted with an inter-frame gap of 2 ms to avoid queuing delays.

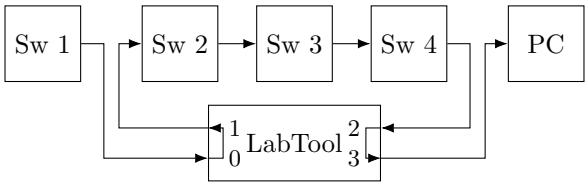


Figure 8.2: Topology used to measure the end-to-end delay in the absence of faults.

As we have mentioned, we study standard TSN, approach A and approach B and we use three different frame lengths (64, 782 and 1500 Bytes), producing a total of nine experiments, in each of which 1000 consecutive non-replicated frames are exchanged. Table 8.1 shows the mean, the standard deviation, the maximum and the minimum end-to-end delay. As expected, we can see that the frame length highly impacts the end-to-end delay, as the time required to transmit and process a large frame is significantly higher than for a short one. On the contrary, we observe that the impact of the PTRF mechanism in the end-to-end delay is actually low.

Specifically, we see that the difference in the end-to-end delay between PTRF (both following approach A or B) and TSN is lower than 162 ns, even for the minimum end-to-end delay, i.e. the overhead caused by PTRF is lower than 162 ns. Moreover, we see that PTRF does not cause significant variations in the end-to-end delay in the absence of faults, as we can see observing the standard deviation. Thus, we can conclude that the overhead introduced when the PTRF mechanism is implemented does not pose a threat to the timeliness of the system.

Now that we have measured the impact that the PTRF mechanism has on the end-to-end delay, we want to measure the impact of sending replicas. As we have anticipated at the beginning of this section, we have carried out a second set of experiments. Specifically, in this set we measure the maximum end-to-end delay when we transmit several replicas in the presence of faults.

Table 8.1: End-to-end delay when transmitting a single frame using TSN, approach A and approach B through a 4-hop network.

Network	**Frame Length** (B)	**End-to-end delay** (ns)			
		mean	std	max	min
TSN	64	6487.600	15.704	6534	6435
	782	23743.979	15.704	23787	23697
	1500	40975.512	15.503	41022	40923
App A	64	6631.830	15.989	6687	6588
	782	23863.720	15.222	23904	23814
	1500	41095.400	16.498	41157	41049
App B	64	6630.509	16.538	6696	6570
	782	23863.204	16.520	23913	23823
	1500	41095.432	15.581	41139	41049

Figure 8.3 shows the end-to-end delay of four replicas transmitted through a single link, more specifically the link between Sw1 and Sw2 in Figure 8.2. As we have explained, when we talk about a set of replicas, we measure the end-to-end delay as the time elapsed since the first replica starts being transmitted until the first correct replica is completely received. Therefore, the maximum end-to-end delay corresponds to only receiving the last replica ($r_{1,4}$ in Figure 8.3), i.e. losing all replicas but the last one due to faults.

We use the network configuration shown in Figure 8.4 to measure the maximum end-to-end delay. As in the previous experiment, the first MTSN switch is used as an end-system to transmit frames and the other three switches are used as TSN or PTRF bridges. LabTool 1 is used to timestamp frames and to inject errors on predefined frames, while LabTool 2 is used only as error injection device. As we can see, LabTool 1 is connected to Sw1 and to the PC to have a common time reference when measuring the end-to-end delay.

In this case, we only study approach A and approach B, as TSN does not provide time redundancy. Thus, comparing the approaches to TSN would result in an unfair comparison, as TSN would always be faster but, in contrast, cannot provide the same level of fault tolerance.

As in the previous experiment, we use three different frame lengths (64, 782 and 1500 Bytes) and we transmit 1000 different frames. The end-system (Sw 1) transmits two, three and four replicas of each frame, resulting in 9

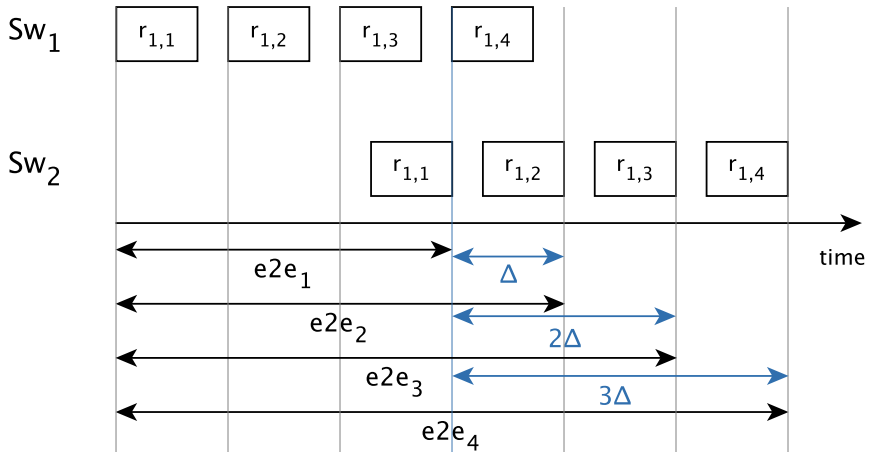


Figure 8.3: This figure shows the end-to-end delay of four replicas transmitted through a link between two bridges (Sw1 and Sw2). Specifically, the figure shows the transmission of four replicas with their respective maximum end-to-end delays marked with black arrows, i.e. e2e1 is the maximum end-to-end delay when transmitting a single replica, e2e2 is the maximum end-to-end delay when transmitting two replicas, etc. Moreover, the blue arrows show the variation between the end-to-end delays, a.k.a. jitter.

experiments for each network. In approach A only the end-system creates replicas and LabTool 1 is configured to introduce errors in all replicas but the last one, only in the first link. The bridges Sw 2, Sw 3 and Sw 4 simply forward the correct frames or replicas and, thus, LabTool 2 is configured to pass frames without errors.

On the other hand, in approach B all the switches use PTRF and all of them are configured to replicate two, three and four times depending on the experiment. In this case, Labtool 1 and LabTool 2 are configured to introduce errors in all replicas but the last one in all links. Each bridge receives the only correct replica and produces a new set of replicas. We introduce errors in all replicas but the last one in all links to achieve the maximum end-to-end delay in the multi-hop network.

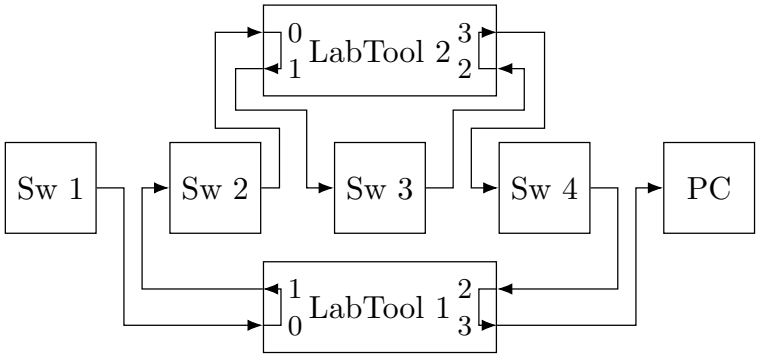
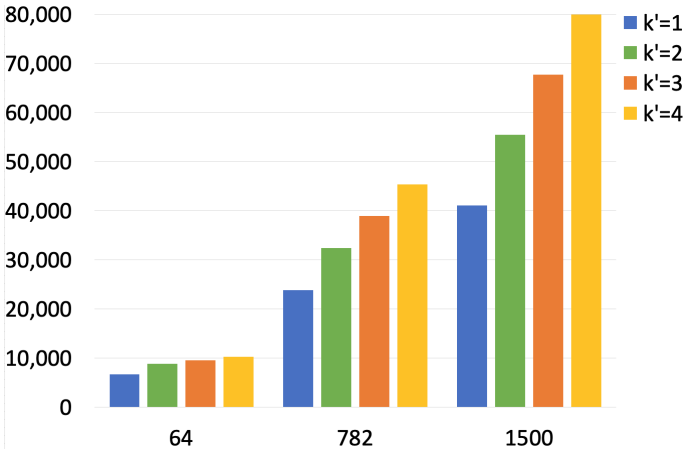


Figure 8.4: Topology used to measure the end-to-end delay and the jitter in the presence of faults.

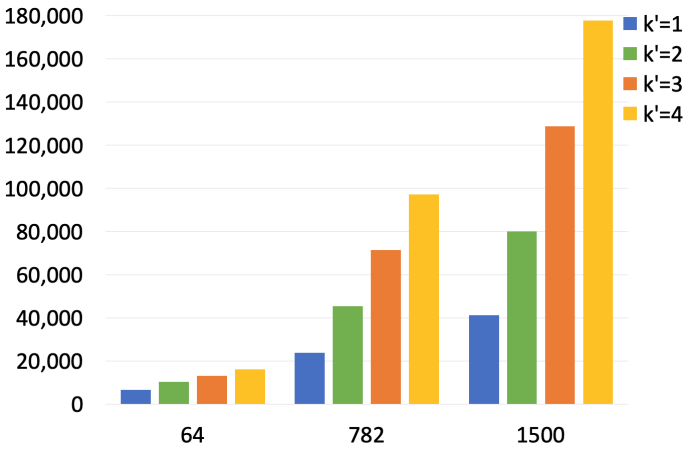
Table 8.2: Maximum end-to-end delay in nanoseconds when transmitting two, three and four replicas using approach A and approach B.

Network	**Frame length** (B)	$k = k' = 2$			$k = k' = 3$			$k = k' = 4$		
		mean	std	max	mean	std	max	mean	std	max
App A	64	8778.74	18.99	8829	9513.73	18.48	9576	10250.45	19.31	10314
	782	32412.10	705.81	32544	38912.56	612.81	39033	45392.19	611.56	45522
	1500	55454.69	386.92	55521	67679.37	387.06	67752	79916.40	19.86	79965
App B	64	10250.16	18.03	10314	13195.39	18.63	13248	16138.32	17.58	16191
	782	45409.83	507.82	45495	71350.47	354.20	71424	97250.73	521.15	97353
	1500	79890.84	547.79	79974	128799.66	387.34	128871	177681.99	583.65	177768

Table 8.2 shows the mean, the standard deviation and the maximum values for the maximum end-to-end delay when transmitting two, three and four replicas using approaches A and B. As expected, the maximum end-to-end delay increases with the number of replicas transmitted and it is higher in approach B than in approach A. Nevertheless, it is important to note that the number of fault scenarios that can be tolerated also increases with the number of replicas and it is also higher using approach B than using approach A, as we showed in (Álvarez et al., 2019).



(a) Mean e2e delay approach A for k = 1, 2, 3 and 4.



(b) Mean e2e delay approach B for k' = 1, 2, 3 and 4.

Figure 8.5: Mean value for the maximum end-to-end delay for approaches A and B with k = k' = 1, 2, 3 and 4. The X axis represent the frame length in bytes, while the Y axis represent the end-to-end delay in nanoseconds.

Moreover, we can observe that the standard deviation is really low when the frame length is short, but it significantly increases for medium and large frames, which means that the deviation in the end-to-end delay is higher when using time redundancy with larger frames.

We now compare these results to the ones obtained in the previous experiment (Table 8.1). Figure 8.5 shows the mean value for the maximum end-to-end delay for approaches A and B when transmitting one, two, three and four replicas. Observing the results of approach A we see that the difference in the end-to-end delay of $k = 1$ with $k = 2$ is around 2 μs higher than the difference of $k = 2$ with $k = 3$ and $k = 3$ with $k = 4$, regardless of the frame length. We can thus conclude that there are two different contributions to the overhead introduced when transmitting replicas: (i) one contribution is common to all the cases and could be caused by the regular forwarding and error detection mechanisms, (ii) another contribution that varies depending on the number of replicas transmitted and their length. In any case, we can see that the end-to-end delay is always under 80 μs.

Regarding approach B, we see that maximum the end-to-end delay increases considerably with the number of replicas transmitted regardless of the frame length. This is because in approach B the whole set of replicas is transmitted in every link, even though some replicas are corrupted during transmission. Furthermore, since in this experiment we corrupt all replicas but the last one, each bridge must wait for the last replica in order to generate the new set. Since this is repeated in each hop, the impact is considerably higher than in approach A. In any case, we see that the end-to-end delay is under 80 μs for frames of small and medium length, and under 180 μs for large frames.

8.3.2 Transmission jitter

Timeliness is crucial to guarantee the correct operation of hard real-time systems. In fact, a factor that can negatively impact the correct behaviour of a system is the jitter in the communication. Jitter can roughly be defined as the variation on the end-to-end delay between the transmission of several frames through a path or stream. This variations can be caused by differences in the queuing and processing times. In PTRF, we have to also consider the variations in the queuing and processing times that replication introduces, as well as the time required for the creation of replicas.

We carry out a first set of experiments to measure the jitter. In the first set of experiments we use the simple topology depicted in Figure 8.6 with a single end-system (Sw 1) and a single bridge (Sw 2). In this way we can

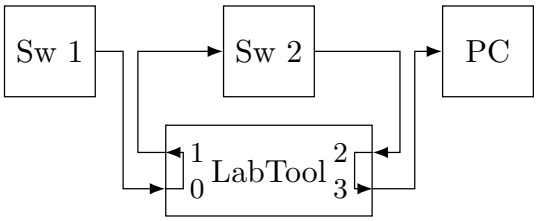


Figure 8.6: Topology used to measure the jitter in the absence of faults.

isolate the impact of PTRF from other queuing and processing variations. The end-system transmits 2, 3 and 4 replicas. We then compare the time required to transmit the last replica to the time required to transmit the first one to obtain the jitter. We do this for frame lengths of 64, 782 and 1500 bytes. We carry out this experiment in the absence of faults as we want to compare the jitter introduced by the creation and processing of replicas.

Table 8.3 shows the results of this first experiment. The first thing we notice observing the results is that the jitter measured is 0 when k = k' = 2 and the frame length is 64B, probably because the variation in the delay could not be measured by our equipment. Since we only have one hop and the cable and frame lengths are short, the jitter is clearly negligible in this case for both approaches.

Observing the rest of our results, we clearly see that the jitter increases with the frame length and the number of replicas. Moreover, we can see that the mean jitter measured for approach B is slightly higher than for approach A in most cases, probably due to the fact that the bridge (Sw 2) must eliminate and create the new set of replicas. This is also reflected in the standard deviation, which is mostly higher for approach B than approach A, which means that the jitter suffers a higher variation in the former.

Nevertheless, in PTRF we have to consider an additional level of jitter, as the end-to-end delay of a frame varies depending on which correct replica reaches its destination first in the presence of faults. More specifically, we measure this level of jitter as the difference between the minimum and the maximum end-to-end delay, i.e. the difference in the end-to-end delay when the first replica is correctly received and when only the last replica is correctly received. We carry out these experiments in the presence of faults. to force the loss of al replicas but the last one.

Table 8.3: Mean, standard deviation and maximum jitter when transmitting two, three and four replicas using approach A and approach B in a one-hop network. All the results are in nanoseconds.

Network	**Frame length** (B)	$k = k' = 2$			$k = k' = 3$			$k = k' = 4$		
		mean	std	max	mean	std	max	mean	std	max
App A	64	0	0	0	736.030	4.058	756	1472.332	4.834	1494
	782	6479.855	3.522	6498	12894.142	682.271	15129	19354.956	1156.110	25920
	1500	12224.751	5.803	12249	24412.945	701.358	26631	36555.708	1410.492	36702
App B	64	0	0	0	736.164	3.824	756	1472.058	4.627	1485
	782	6460.380	354.751	6489	12920.900	500.816	12969	19404.496	475.906	19449
	1500	12211.308	386.568	12240	24447.141	7.804	24462	36633.248	687.231	36702

Figure 8.3 depicts the jitter introduced by PTRF when transmitting a different number of replicas in the presence of faults. Let us assume that Sw1 only transmits two replicas, $r_{1,1}$ and $r_{1,2}$. Let us also assume that in one transmission $r_{1,1}$ reaches Sw2 with end-to-end delay $e2e_1$. Let us now assume that in another transmission, $r_{1,1}$ does not reach Sw2, but $r_{1,2}$ does. In this case, the end-to-end delay is $e2e_2$. The difference between both end-to-end delays is the jitter, indicated with Δ in Figure 8.3. We also see in Figure 8.3 the jitter when transmitting three replicas, 2Δ, and four replicas, 3Δ.

Thus, we carry out a second set of experiments. We now measure the jitter in the presence of faults using the network topology depicted in Figure 8.4. In this way, we can study the impact of using a larger network and we can measure this new level of jitter introduced by PTRF.

Table 8.4 shows the measured jitter for each approach, with the different frame lengths and number of replicas. If we take a close look to the results obtained for $k = k' = 2$ replicas and frame length of 64 B, we see that the maximum jitter is 2.2 μs for approach A and 3.7 μs for approach B. Therefore, we can corroborate that the jitter for a single hop was too low to be measured by our equipment.

Observing the overall results, we can see that, just like the end-to-end delay, the jitter is greatly affected by the frame length and number of replicas. Moreover, we also see that the jitter introduced by approach B is significantly higher than by approach A. This is because in approach B there is a contribution to the jitter in each link, while in approach A there is only the contribution in the first link.

In any case, we can see that the maximum measured jitter for a 4-hop network and 4 replicas using approach A is under 40 μs, even for the largest frames. On the other hand, we see that the maximum jitter for approach B is considerably higher, around 137 μs. Again, it is important to note that the number of fault scenarios tolerated by approach B using the same number of replicas is also considerably higher than by approach A. Nevertheless, these results show that when the frame length is large and the number of replicas must be high, approach A is a more suitable solution than approach B for jitter-sensitive applications.

Table 8.4: Mean, standard deviation and maximum jitter when transmitting two, three and four replicas using approach A and approach B in a four-hop network In the presence of faults. All the results are in nanoseconds.

Network	**Frame length (B)**	$k = k' = 2$			$k = k' = 3$			$k = k' = 4$		
		mean	std	max	mean	std	max	mean	std	max
App A	64	2146.914	24.570	2232	2882.125	25.096	2961	3618.641	25.084	3708
	782	8543.211	728.160	8694	15051.944	596.498	15174	21524.725	630.428	21681
	1500	14358.351	398.471	14445	26583.137	399.794	26667	38821.610	389.461	38889
App B	64	3618.333	23.894	3690	6563.565	24.271	6642	9506.493	23.673	9576
	782	21546.346	507.302	21663	47485.637	366.145	47574	73387.060	521.007	73512
	1500	38794.171	563.057	38898	87703.380	398.456	87804	136586.633	583.833	136692

8.3.3 Bandwidth overhead

Bandwidth consumption is one of the main concerns in the development of novel converged networks for the applications targeted by TSN. For this reason, it is important to minimise the impact that fault tolerance mechanisms have on the bandwidth consumption. Moreover, it is of utmost importance to quantify this impact. We carry out two different experiments to measure the impact that PTRF has on bandwidth.

To measure how PTRF impacts bandwidth, we measure the overhead introduced by creating replicas. To that, we carried out a first experiment to compare the time required to transmit a set of frames to the time required to transmit a set of replicas. That is, we measured the time required to transmit 1000 frames to the time required to transmit 1000 replicas of a single frame. Nevertheless, during this experiment we observed that the time required to produce a replica is significantly shorter than the time required to produce a new frame. This is due to the fact that frames are produced by the application and processed by the software stack, while replicas are created directly in the network level. Therefore, this experiment did not allow us to quantify the bandwidth overhead.

Therefore, we carried out a second set of experiments in which, instead of sending replicas of a single frame, we replicated different frames using the network depicted in Figure 8.2. In approach A the end-system (Sw 1) transmits two, three and four replicas while the TSN bridges (Sw 2, 3, 4) only forward them. In Approach B, the end-system and the PTRF bridges produce the same number of replicas, which go from two to four. Moreover, we carry out each experiment with frame lengths of 64, 782 and 1500 Bytes to take into account the different processing times, resulting in 21 experiments.

In order to measure the impact on the bandwidth, we count how many frames (with their corresponding replicas) can be transmitted within a time slot using TSN, approach A and approach B. The lower the number of frames within a slot, the higher the impact on bandwidth of the mechanism. We believe that measuring the impact on bandwidth in a per-slot basis is a good approach for TSN networks, because the communication is divided in time slots which means that the schedule is also calculated in a per-slot basis.

We use the Time-Aware Shaper to configure the time slots for all priorities. Specifically, we use three different configurations for priority 5 (i) 1 ms open and 5 ms closed, (ii) 10 ms open and 5 ms closed and (iii) 20 ms open and 5 ms closed; where open means that frames can be transmitted while closed means that frames cannot be transmitted. The time during which the gate is

closed allows us to delimit the different slots and count the number of frames that the network is going to be able to effectively exchange within each slot. The gates of the rest of priorities are always configured to be closed so no frames from a different priority can be transmitted during the experiments.

We decided to use slot sizes of 1, 10 and 20 ms, as we consider that these sizes are diverse enough to study the behaviour of PTRF under different circumstances, while still being realistic, e.g. 1 ms for the transmission of control traffic and 20 ms for the transmission of multimedia traffic.

Table 8.5: Mean number of number of frames that the network can effectively exchange within a slot using TSN, approach A and approach B, with different slot sizes and frame lengths.

Network	**Frame**	**Number of frames per slot**		
	Length (B)	1ms—5ms	10ms—5ms	20ms—5ms
TSN	64	23.827	163.669	322.587
	782	23.607	162.750	318.478
	1500	20.246	161.166	317.321
App A	64	23.534	162.287	321.682
	782	23.798	161.574	316.459
	1500	19.978	160.197	316.957
App B	64	23.937	162.820	321.550
	782	23.821	161.554	316.462
	1500	20.139	160.567	314.226

Table 8.5 shows the mean number of frames that the network can effectively exchange within a time slot for TSN, approach A and approach B. Note that since we have calculated the mean, we show the results with 3 decimals, even though the number of frames exchanged within a slot is, obviously, an integer number. As expected, the number of frames that can be exchanged within a slot increases with the size of the slot and decreases with the frame length. Regarding the impact of PTRF, we can see that the overhead when the time slot is small is barely noticeable, regardless of the frame size. With a medium-size slot TSN can convey 1 more frame per slot than approaches A and B and with large slots TSN can convey up to two frames more per slot. Thus, we can see that the impact on bandwidth of using PTRF is low. Furthermore, we can see that the difference between approaches A and B is negligible for all slot sizes and frame lengths studied.

8.3.4 Summary

The design of a fault-tolerant system or mechanism is always a trade-off between reliability and performance. Nevertheless, in certain scenarios the use of such mechanisms is essential, and even mandatory, to guarantee the correct operation of the system. In these cases, the target is to design fault tolerance mechanisms that have the lowest impact on the systems' performance. Observing the results obtained, we can conclude that the PTRF mechanism is a suitable solution to provide time redundancy to TSN networks.

On the one hand, we have seen that the impact of PTRF on the end-to-end delay is negligible in the absence of faults, always under 125 ns. Furthermore, we have also studied the maximum end-to-end delay in the presence of faults, which corresponds to losing all replicas except for the last one during transmission. The maximum end-to-end delay was always under 80 μs for approach A and under 180 μs for approach B, even when transmitting four replicas of 1500 B. Thus, we think that both approaches provide acceptable values for the maximum end-to-end delay, with approach A providing the lowest upper bound of the two.

Regarding jitter, we have observed that the jitter introduced by the creation and processing of replicas in the absence of faults is under 40 μs even for the largest frames and highest number of replicas. Furthermore, the difference between approaches A and B is negligible. On the other hand, we have studied the jitter introduced by the transmission of replicas in the presence of faults. In this scenario we observe that the jitter for approach A was always under 40 μs, and the jitter for approach B was under 140 μs, even when transmitting four replicas of 1500 B. Observing all the results, we can conclude that both approaches are adequate when the frame length is short or the number o replicas is low. Nevertheless, for larger frames or higher number of replicas, approach A is a more adequate solution than approach B, specially for time-sensitive applications.

We have also seen that the impact of PTRF in the bandwidth is low. In fact, we have seen that generating replicas requires considerable less time than generating frames, as replicas are created in the second layer of the communication stack, while frames are generated in the application layer. On the other hand, we have seen that the overhead introduced by the process of creating replicas is negligible for short frames and small time slots. Furthermore, the difference on the number of frames that can be exchanged in a slot is also low when the frame length or the slot size increase, i.e. up

to two more frames exchanged per slot in TSN than in PTRF.

For all these reasons, we think that PTRF is an adequate solution to tolerate temporary faults in the links of TSN-based networks, even for time-sensitive applications with tight timing requirements. We think that the impact that approaches A and B have on the performance of the system is acceptable for applications that require a high level of reliability, helping us to achieve our thesis statement.

Chapter 9

Reliability Evaluation of PTRF

To finally proof our thesis, we have carried out a parametric sensitivity analysis that quantifies the reliability benefits of tolerating temporary faults in TSN networks by means of the time redundancy mechanisms of PTRF. To that aim, we built three models based on the PRISM probabilistic model checker. These models include parameters that represent several aspects of the network and PTRF that may influence the reliability.

As we have already mentioned, despite the interest in providing TSN networks with fault-tolerance mechanisms, there are no previous works that quantify the reliability that these mechanisms can yield. In fact, most of the works that treat space redundancy in TSN focus on evaluating the impact on the performance, bandwidth or schedulability of the network (Nasrallah et al., 2019). Furthermore, even the works that do focus on dependability aspects do not provide a quantitative measure for the reliability. Instead, they analyse it in a qualitative manner or focus on specific gains in the fault tolerance capabilities, e.g. counting the number of simultaneous faults that can be tolerated.

More specifically, we can find in the literature other works related to the study of TSN and its fault tolerance mechanisms. In (Hofmann, Nikolić, and Ernst, 2020) the authors discuss in a qualitative manner the challenges and limitations of the IEEE Std 802.1CB standard for frame replication and elimination that supports space redundancy. In (Atallah, Hamad, and Mohamed, 2019) the authors study the capability of tolerating the faults caused by single event upsets using a duplicated communication path, showing

that an adequate schedulability analysis is required to tolerate all faults even with redundant paths. Finally, in (Pahlevan and Obermaisser, 2018) the authors present a simulation model for redundancy management and validate the design of the redundancy mechanism using fault injection. However, as we mention, none of these works provide a quantitative evaluation of the reliability.

In this chapter we present the three formal models that we propose to carry out our parameterised sensitivity analysis of the reliability. These models are based on the PRISM probabilistic model checker which has extensively been used to evaluate the reliability of systems, protocols and networks. Some examples of this are the following : (Rosset et al., 2012), where the authors use PRISM to model and evaluate the reliability of a group membership protocol for dual-scheduled time division multiple access networks; or (Barranco, Derasevic, and Proenza, 2020), where the authors study the reliability achievable when using fault tolerance mechanisms in an architecture specifically designed for highly-reliable adaptive distributed embedded systems.

First we describe how we designed and built our models and then we move to the results of the analyses that we have carried out.

9.1 Modeling rationale

In this section we describe the most relevant aspects of the PRISM models for TSN and PTRF, namely the reliability metric we use, the fault and the failure models of the PRISM models, the modeling assumptions, the modeling strategy, the model limitations and the model validation. We must indicate that we only develop models for approach A and B, but not C as it is a specific case of approach B.

We need to keep in mind during the following discussion that, even though PTRF is designed to tolerate faults in the links of the network, we cannot actually quantify the reliability of links as isolated components. Instead, we need to evaluate the reliability that the whole network can reach when implementing PTRF. This is because frames are not only transmitted through links, they also traverse end-systems and bridges. Thus, in order to properly quantify the reliability achievable when implementing PTRF we need to take into account how PTRF is affected by all the devices of the network.

Another important aspect to keep in mind during the description of the models is that we have developed our models in such a way that, in the future,

we can evaluate the reliability achievable when using PTRF together with TSN's fault tolerance mechanisms: space redundancy 802.1, 2017c; 802.1, 2016b and error containment 802.1, 2017a. Thus, our models can be tuned to include temporary and permanent faults, as well as time redundancy, space redundancy and error containment. Therefore, even though in this article we focus on temporary faults and time redundancy, we discuss all these aspects of our models.

We next start by defining our reliability metric.

9.1.1 Reliability metric

As we have already mentioned, the general definition of reliability stands as the probability with which a system or subsystem provides a correct service in a continuous manner for a specific amount of time, called mission time. However, in order to build up a model that quantifies the reliability, first we need to particularize this general definition into a reliability metric which the model must calculate. In this sense, the reliability metric must be aligned with the objective of the model. That is, we must decide and state clearly what we are measuring.

As we have mentioned, our models allow us to quantify the reliability benefits that TSN networks can obtain thanks to several fault-tolerance mechanisms of TSN standards and the fault-tolerance mechanisms of PTRF. It is important to note that we do not model the TSN network of any specific system. Instead, we want to quantify the reliability achievable by *any* TSN network. This implies that the reliability metric has to be defined in such a way that it is general enough to be applicable to the network of any system, regardless of its specific characteristics. Thus, we propose a reliability metric to quantify the reliability of streams.

We must note that in this work we only consider time-triggered traffic. Therefore, in our work a stream is a virtual communication channel which consists of a talker sending a message edition every T units of time to one or several listeners through a given number of paths; where T is the stream period. In case of using TSN, the talker only sends one copy of each message edition per period (one copy through each path in case of having multiple paths). In the case of PTRF, the talker sends several replicas of each message edition through each path as explained in Chapter 6.

On top of that, we can specify in the model the number of listeners that must receive each message edition in order for the system to work correctly.

In this way, we can take into account systems that only require a subset of the intended end-systems to communicate correctly to provide a correct service, e.g. systems that use node redundancy to tolerate faults and thus can operate with a subset of end-systems only.

At this point we can define our reliability metric as the probability that for each stream a subset of the subscribed listeners receives at least one copy of each message edition during a given interval of time called mission time.

In the particular case of the analyses that we show in this article, we do not consider space redundancy. This is so because in this article we want to prevent the reliability benefits of using space redundancy from masking the reliability benefits of the time redundancy provided by PTRF. Thus, our reliability metric is calculated taking into account one path. Furthermore, we only analyse the impact of the reliability for one talker that communicates with a single listener, as this allows us to draw significant conclusions while minimising the factors that can interfere with our analysis. In the future we plan to evaluate networks with several streams, each with several listeners.

Finally, it is important to note that this metric alone does not quantify the reliability of a whole distributed system based on TSN. To achieve such a quantification, it would be necessary to model the particularities of a given system, as well as to include all the reliability mechanisms the system is provided with, e.g. the mechanisms to replicate the talker and listener themselves. Thus, our metric and models have to be understood as a relevant piece that can help in quantifying the reliability of a whole system, but not as the only piece that is necessary for doing so.

9.1.2 Fault model of the modelled system

As we have already explained, the fault model specifies the fault types that are assumed that may happen during the operation of the system. In Chapter 6 we specified the fault model with which PTRF deals, i.e. temporary non-malicious hardware faults that affect the links. However, as we have mentioned we cannot model links as isolated devices, instead we need to model all the components of the communications subsystem, i.e. links, bridges and the communication controllers of end-systems. Thus, we need to extend the fault model to consider faults affecting the bridges and the communication controllers of end-systems too.

On top of that, we need to take into account that we developed our model taking into consideration TSN's space redundancy and error contention

mechanisms to be able to extend our analyses in the future. Therefore, we need to also include permanent non-malicious hardware faults that affect any device of the network. Therefore, the fault model of the modelled system is *permanent and temporary non-malicious hardware faults that affect the links, the bridges and the communication controllers of end-systems.* This extension of the fault model has resulted in an extension of the failure modes of the system too, which is thoroughly explained in Section 9.1.3.

Particularly, in the analyses presented in this article we have tuned the models to only consider temporary faults that affect links, bridges and the communication controllers of end-systems. As said before, we do not include permanent faults nor space redundancy in the analyses presented in this article as we do not want to mask the benefits that PTRF can yield to the network. Nonetheless, we must note that all the networks that we analyse here include the error-containment mechanisms of TSN. Specifically, our networks include Ethernet's CRC and TSN's filtering and policing, which allow the network devices to drop frames that are incorrect in the time and the value domains, as long as these errors are produced at the network level, e.g. a bit flip in the output queue of an end-system or bridge.

Finally, it is important to note that our models do not include faults happening in parts of the system other than the communication subsystem, e.g. software faults, hardware faults affecting the end-system's main memory, etc. These faults affecting other parts should be considered in case of modeling a complete system that may include mechanisms to tolerate them; which is not the case of this article.

9.1.3 Modeling assumptions

Every model relies on a set of assumptions; some of them can be reflected as model parameters, whereas others determine the structure of the model in itself. The model parameters allow us to carry out parametric sensitivity analyses, such as the ones we present in Section 9.2. As we explain there, first we establish a case of reference, in which we assign to each parameter a value that can be considered as realistic but, at the same time, pessimistic for PTRF. This allows us not to bias this case towards PTRF. Then we carry out a set of analyses; where each one of them consists in quantifying the reliability when varying one of the parameters with respect to the case of reference.

Table 9.3 shows the parameters that we have considered in our models, e.g., the period of the stream, the frame size, the number of replicas, etc.;

the range of values we have considered for each one of these parameters in the analyses; their values for the case of reference; and their meaning.

In the rest of this section we describe the assumptions that determine the structure of our models; and we further comment on the ranges we have considered for some of the parameters. All the assumptions that determine the structure of the model are pessimistic in order to obtain an lower bound to the reliability achievable when using PTRF.

Constituent components and failure rates

To obtain a model that is computationally solvable, it is necessary to abstract away the constituent components of the system to some extent. In our models we differentiate between links, bridges and the communication controllers of the end-systems. For the sake of succinctness, we will refer to the communication controllers as end-systems.

Each component has a different failure rate which varies depending on its complexity and quality. We must note that, since in this work we only consider temporary faults, we only specify the components' failure rates caused by these faults.

We characterise the rate of temporary faults affecting the links by means of the *bit error rate* (BER)[1], i.e. the ratio between the number of erroneous bits and the total number of transmitted bits. Specifically, we consider BER values that range from 1E-4, usually considered in highly critical applications, to 1E-12, usually considered in non-critical applications. In this way we cover BER values considered in relevant applications with a wide range of reliability requirements, e.g., aerospace or automotive Smirnov et al., 2016.

Regarding temporary faults affecting bridges and end-systems, we have considered different failure rates depending on whether they are standard TSN or PTRF-enabled. This is a reasonable assumption since PTRF-enabled devices are slightly more complex, as they count with extra hardware and logic to implement the time redundancy mechanisms, i.e. the ones that support frame replication. This extra hardware is highlighted in blue in Figure 8.1.

In order to estimate the failure rate of PTRF-enabled devices we use the implementation of PTRF on a real TSN bridge, which has been implemented using an FPGA as we explain in detail in Section 8.1. Unfortunately, we do

[1]Sometimes also referred to as Bit Error Ratio.

not count with the exact information regarding the occupation of this extra hardware in the FPGA. Thus, we make a pessimistic assumption and we estimate that PTRF implies an increase of the 50% in the hardware of the devices and, thus, an increase of a 50% in the failure rate when compared to TSN-enabled devices.

We have considered a failure rate of 1E-4 per hour for standard TSN bridges and end-systems Peti et al., 2005. On the other hand, we have considered a failure rate of 1.5E-4 per hour for these devices when they are PTRF-enabled.

Failure model and fault-tolerance coverages of the modelled system

The failure model of a system includes the failure modes of each one of its components. Therefore, we need to specify the failure modes of all the devices included in the system, i.e. in this case the devices of the network. The failure modes that constitute the failure model, as well as the proportions with which they manifest, have a strong influence on the system reliability. Thus, we take both these aspects into account while defining our models. To better understand the influence of the failure modes, we can recall the failure mode hierarchy presented in Chapter 2.

As we have already explained in Chapter 6, temporary faults affecting a link are assumed to provoke an omission, i.e. a frame that is corrupted due to a temporarily faulty link is dropped by the communication port that receives it. This means that we will assume that the only failure mode that links exhibit is omission.

The consequence of a frame omission on the reliability metric depends on whether the network is provided with space or time redundancy. A frame omission in a given path prevents the listener from receiving the corresponding message edition through that path, unless the frame is replicated in the time domain in that path.

In the particular case of the analyses presented here, we are considering that the network does not have space redundancy, i.e. the stream traverses a single path, both if TSN or PTRF is considered. In the case of TSN, since it does not count with any time redundancy mechanisms, only one copy of each message edition is transmitted through the path. As a consequence, if a temporary fault in a link corrupts a frame, the conveyed message edition does not reach its destination, provoking the failure of the stream.

Table 9.1: Unreliability of each device, failure mode proportions and their impact on the stream.

Device	Unreliability	Faulty port	Prop.	Faulty component	Failure mode	Prop.	Stream failure
TSN Talker	$1 - e^{-(Tx \times 1E-4)}$	Output	50%	Port/queue	Byzantine	100%	**Y**
		Input	50%	-	-	-	**N**
TSN Bridge	$1 - e^{-((Rx+Fw+Tx) \times 1E-4)}$	-	100%	-	Byzantine	100%	**Y**
TSN Listener	$1 - e^{-(Rx \times 1E-4)}$	Output	50%	-	-	-	**N**
		Input	50%	Port/queue	-	100%	**Y**
PTRF Talker	$1 - e^{-((Tx \times 1.5E-4)}$	Output	50%	Port/queue	TSN + buffer	83.33%	**Y**
				PTRF transmission	counter > k	8.33%	**Y**
					counter < k	8.33%	$(\frac{k-1}{k})\mathbf{N}/(\frac{1}{k})\mathbf{Y}$
		Input	50%	-	-	-	**N**
PTRF Bridge	$1 - e^{-((Rx+Fw+Tx) \times 1.5E-4)}$	Output	50%	Port/queue	TSN + buffer	83.33%	**Y**
				PTRF transmission	counter > k	8.33%	**Y**
					counter < k	8.33%	$(\frac{k-1}{k})\mathbf{N}/(\frac{1}{k})\mathbf{Y}$
		Input	50%	PTRF reception	Fail to eliminate	100%	**Y**
PTRF Listener	$1 - e^{-(Rx \times 1.5E-4)}$	Output	50%	-	-	-	**N**
		Input	50%	PTRF reception	Fail to eliminate	100%	**Y**

Tx: time required for the port to transmit the frame.
Rx: time required for the port to receive the frame.
Fw: time required for the bridge to forward a frame from the input port to the output port.
-: the value is irrelevant as it does not affect the result.

Conversely, PTRF transmits through the path several replicas of the frames that convey each message edition. Thus, the omission of a replicated frame does not provoke the failure of the stream, as long as the listener receives at least one replica of that replicated frame.

As regards bridges and end-systems, they are assumed to exhibit byzantine failures, when faulty. The impact of such failures on the exchange of frames through the stream depends on the following aspects: (i) whether or not space redundancy is used; (ii) whether the faulty device is TSN or PTRF-enabled; (iii) whether the faulty hardware is involved in the transmission or the reception capabilities of the device, i.e. whether it affects an output or an input port; and (iv) how the faulty hardware can affect the transmission/reception operations of the device.

Table 9.1 specifies the expression used to model the unreliability of bridges and end-systems; the proportions of their different failure modes; and the impact of each failure mode on the reliability metric that we have defined in the previous subsection when using TSN and PTRF with a single path. For instance, we can see that the unreliability of a TSN standard talker is expressed using the equation $1 - e^{-(Tx \times 1E-4)}$, where Tx represents the time required for a port to transmit a frame. We can also see that, whenever a fault affects the talker, there is a 50% probability that this fault affects its output port and a 50% probability that it affects its input port. If the fault affects the output port, we see that it can affect the whole port or just the queues and, regardless of the affected component, the fault manifests as byzantine and, with a 100% probability it results in the failure of the stream, as we explain in detail.

In the specific case of TSN, when a temporary fault affects the input or the output port of a TSN bridge, the bridge transmits, through one or several output ports, a byzantine version of the frame it is forwarding. Fortunately, each one of these byzantine versions is dropped by means of the error-containment mechanisms of the next bridge or listener that receives it; thereby transforming each one of these byzantine frames into an omitted one. In this sense, the impact of a faulty bridge resembles the one of a faulty link, i.e. it prevents the listener from receiving the affected frames through the corresponding path.

In the particular case of using a single path, this results in the failure of the stream. Conversely, the fault can be tolerated when using multiple paths, as long as the byzantine failure does not compromise the schedule of other streams transmitted or forwarded by the same device, e.g. by starving the

bandwidth of the links connected to the output ports through which the talker or bridge sends byzantine frames. For the sake of simplicity, we have decided to model this scenario by means of a parameterised probability of affecting the schedule when a byzantine fault affects a bridge.

A temporary fault affecting a TSN end-system only results in a frame being incorrectly transmitted or received if the fault affects the components involved in the operation of the end-system. Specifically, if the end-system acts as a talker, then the fault can only lead it to transmit a byzantine version of the frame through a given output port if the fault affects that port. Likewise, in the case of the listener, a fault can only lead it to receive a byzantine version of the frame through a given input port, if the fault affects that port.

The impact of a byzantine frame being transmitted by a TSN talker is the same as if it was transmitted by a bridge, i.e. the frame is dropped by the next bridge, causing the stream failure when using a single path or in case the corrupted frame affects the schedule of other streams when using multiple paths. In contrast, we assume that the impact of a fault affecting any input port of a TSN listener always is the failure of the stream. This is because, since the fault happens within the listener itself, it may affect the error-containment mechanisms that supervise the correct operation of the port.

In the analyses we present here, which assume a single path, we consider that each end-system has one output port and one input port. Since the complexity of both ports is similar, we consider that when a fault happens in an end-system, it affects equiprobably its output and input ports. This is shown in the column *Prop.* of the left side of Table 9.1, where the proportion with which each port is affected is 50%. In any case, these proportions are parameters of our model, and different values for them will be considered in our experiments.

The analysis of the negative impact of temporary faults affecting PTRF-enabled bridges and end-systems is more complex. Unlike TSN, PTRF does provide time redundancy. Hence, as said before, faults can be tolerated in the corresponding path, as long as the listener receives at least one copy of each message edition through that path. To simplify the explanation from hereon, let us consider a single path. In order to model how faults can be tolerated when using PTRF, we need to differentiate which failure modes PTRF-enabled devices may exhibit on top of the ones exhibited by TSN devices. This is so because certain failure modes may exhaust the time

redundancy provided by PTRF and, thus, exceed its fault-tolerance capacity. Table 9.2 describes the failure modes of PTRF.

For this purpose, let us revisit Figure 8.1, which shows coloured in blue the extra components of a PTRF-enabled bridge. As already explained in Chapter 8, the blue components represent the extra hardware required for PTRF to create replicas during transmission and to eliminate surplus replicas upon reception. More specifically, the left side of the Figure shows the components that eliminate surplus replicas upon reception. Let us refer to them as *PTRF reception components*. The right side of the figure presents the components that create and then queue in the output queues the frame replicas. Let us refer to them as *PTRF transmission components*.

A fault affecting the PTRF transmission components may manifest as the two following failure modes in addition to the byzantine failure mode of TSN transmission devices: (i) queuing corrupted frame replicas; or (ii) queuing an incorrect number of frame replicas. Table 9.2 also explains these two failure modes and their impact. The first one provokes the failure of the stream. In contrast, the second one only provokes such a failure when it creates more replicas than expected ($n > k$), as having more frames in the network would cause the schedule to be violated, or when it creates no replicas ($n = 0$). Otherwise, this second failure mode simply reduces the ability to tolerate further faults affecting the message edition ($0 < n < k$).

The proportion of these PTRF-specific failure modes can be parameterised in the model, but Table 9.1 shows the proportions selected for our sensitivity analysis. We can see in the Table that the proportion of the TSN byzantine and the buffered failure modes in the output port of PTRF talkers and bridges is the 83.33% (labelled as "TSN + buffer" in the table). Let us explain how we obtain this value. As we have previously said, we assume that the failure rate of PTRF devices is a 50% higher than for TSN ones due to the extra hardware required by PTRF, i.e. TSN's logic occupies 2/3 of the hardware while PTRF's logic occupies 1/3. We then assume that TSN's byzantine failure mode has a proportion of 2/3 (66.67%), while PTRF's failure modes have a proportion of 1/3 (33.33%).

On top of that, we assume that the failure modes of PTRF's transmission are equiprobable, i.e. the probability of frame replicas being corrupted in the buffer and the probability of creating the wrong number of replicas are the same (50%). The 50% of a proportion of 1/3 is 16.66%.

Table 9.2: Failure modes of PTRF-enabled devices and how they affect the overall system.

Component failure	**Description**	**Stream failure**
Frames corrupted in the replication buffer	Temporary faults can result in the corruption of frames before they are replicated, e.g. due to a bit flip. If the frame that is going to be replicated is corrupted, all the replicas will be corrupted too and the receiver will drop them when checking the CRC. If this happens the intended listeners do not receive the frame and, thus, they cannot carry out their operation properly.	Yes
Frame replication counter corrupted, creating n≠k replicas.	n>k. If a device creates more replicas than expected their transmission may affect the transmission of correct scheduled frames.	Yes
	0<n<k. If the device creates less replicas than expected, then it provokes redundancy attrition, i.e. it reduces the number of forwarded frame replicas and thus the capacity of tolerating faults of the corresponding message edition in the hop, but does not provoke the failure of the stream.	No
	n=0. If the device does not send any replicas of the frame, the intended listeners do not receive the frame and, thus, they cannot carry out their operation properly.	Yes
Failing to eliminate surplus replicas upon reception	A PTRF device can fail to eliminate all surplus replicas upon reception. If this happens in a bridge, the surplus replica will be also forwarded and transmitted, violating the schedule. Let us illustrate this with an example. A PTRF bridge receives the first replica of a frame i $r_{1,i}$, it correctly forwards replica $r_{1,i}$ and uses it to create a new set of replicas. Later on, the same bridge receives replica $r_{3,i}$, and it fails to eliminate it. Thus, the bridge forwards replica $r_{3,i}$ and uses it to create a new set of replicas again, resulting in the transmission of twice the number of expected replicas and interfering in the transmission of correct frames. On the other hand, if this happens in the listener, PTRF passes the same frame several times to the application, causing the failure of the system.	Yes
Eliminating all replicas of a frame upon reception	Temporary faults may affect the identification of replicas upon reception in such a way that all the replicas of a specific frame are eliminated. If this happens the intended listeners do not receive the frame and, thus, they cannot carry out their operation properly.	Yes

Since the corruption of frames in the replication buffer always causes the failure of the stream, just like TSN byzantine failures, we can group these two failure modes into a single failure mode with a proportion of 66.67% + 16.66% = 83.33% (TSN + buffer). On the other hand, since the faults that affect the replica creation counter not always result in the failure of the stream, we cannot group this failure mode with the ones previously discussed.Since the proportion of faults that affect the replication counter is the 16.66% and we assume that the probability of creating more replicas than expected is the same as creating less, we obtain a failure mode proportion of 8.33% for $n > k$ and 8.33% for $n < k$.

On the other hand, a fault affecting the PTRF reception components may manifest as in TSN, but also in the following two ways: (i) by failing to eliminate all surplus replicas of a message edition upon reception; or (ii) by eliminating all replicas of a message edition upon reception. Table 9.2 further explains these failure modes and why both of them cause the failure of the stream. As we can see in Table 9.1, faults in the reception of bridges and listeners always cause the stream to fail. Thus, we group all the failure modes into one which we call fail to eliminate that has a proportion of the 100%.

Finally, another important aspect that has an important impact on dependability in general, and on reliability in particular, is the value of the *coverage* associated to the FT mechanisms. Coverage can be roughly defined as the probability with which a system or fault-tolerance mechanism behaves as intended. We find different types of coverage. First we find the assumption coverage, which refers to the probability with which faults manifest as stated in the failure model. Second, we find the error containment coverage, which refers to the probability with which an error containment mechanism effectively contains faults.

We have included a coverage parameter for the assumption coverage, but in this dissertation we have established an assumption coverage of 100% for two reasons. First, the assumption coverage of fault-tolerant systems is usually high, as these systems are conscientiously designed to ensure that they exhibit the failure semantics that it is assumed for them. Second, we tend to assume byzantine failure modes for most devices, which is the least restricted mode and thus corresponds to 100% assumption coverage. Third, we assume that most failure modes cause the failure of the stream, which is a sufficiently pessimistic assumption.

Regarding the coverage of the error containment mechanisms, we must note that the error containment mechanism used to drop erroneous frames is

Ethernet's CRC. As we have discussed in Chapter 6, the CRC can detect virtually all erroneous frames and, thus, we assume a coverage of 100%. We must highlight that assuming a lower coverage would result in an unfair evaluation of the reliability achievable not only by PTRF, but also by TSN.

Topology assumptions

As we have already explained, we want to evaluate the reliability achievable by TSN and PTRF. Moreover, we want this evaluation to be as general as possible and, thus, we do not model any specific system nor any specific network. Instead, we model the network as a set of streams. Thus, we abstract the network topology away and we only keep in our model those aspects that have a significant impact on the reliability of the stream, e.g. the number of hops that the frames of the stream need to traverse.

We should recall that in Chapter 5 we defined a stream as a *virtual communication channel* which communicates one talker to one or several listeners. Now, we introduce the concept of *virtual path*, which communicates the talker of a stream to one of the listeners through a number n of bridges. Thus, each stream has one virtual path for each listener composed of a talker, a listener and n bridges. For simplicity, from now on we will refer to virtual paths simply as *paths*.

In our models all paths are disjunctive, i.e. the paths do not share any network component. This assumption could be considered optimistic, as it implies that all faults are independent e.g. a fault in an output port of the talker only affects one path of one stream. Nonetheless, we can overcome this limitation by tuning the maximum number of paths that can fail before the stream fails. Thus, we can establish that the failure of one path causes the failure of the whole stream, which is a pessimistic assumption as it means that a fault in one component in one path of a stream affects all the paths of that stream.

In the future we plan to quantify the reliability of specific network topologies, modifying our models to express their particularities. In any case, such quantification is out of the scope of this dissertation.

Traffic scheduling assumptions

Since we want to quantify the reliability of TSN and PTRF networks in general, we also want to abstract most details related to the traffic scheduling.

Nonetheless, certain aspects related to the traffic are relevant for the design and evaluation of the models. The most important aspect is that we only consider time-triggered periodic messages since in many applications this type of traffic is usually subject to the most stringent reliability requirements. Thus, all the traffic in our model is transmitted periodically and the period is defined in a per-stream manner.

Another important aspect related to the transmission of frames is the passage of time. In Chapter 5 we propose to use the cyclic queuing and forwarding standard (802.1, 2017b) to schedule frames. As we explain in Chapter 5, CQF schedules frames so that they traverse one and only one hop in each TAS window. Therefore, we have decided to use the TAS window duration as the main time reference for the models. Since the duration of the TAS window in a device depends on the traffic that traverses it, in our model each stream can have a different TAS window duration. Nonetheless, to reduce the complexity of the models we have decided that the TAS window duration must be the same in all the paths of a stream.

Finally, we make two more assumptions related to the traffic. We assume that all the messages fit within one frame, which is a reasonable assumption for most time-triggered traffic with high reliability requirements. Second, the period must always be higher than the end-to-end delay of a frame. All these assumptions prevent us from modeling networks with certain characteristics regarding the traffic scheduling. Nonetheless, since we want to measure the reliability of TSN and PTRF networks from a general point of view, these limitations do not reduce the relevance of the results of our comparison. We will address these limitations in future works.

All of these assumptions allow us to model TSN and PTRF approaches A and B using *discrete-time Markov chains* (DTMC), where the passage of time is associated to the TAS window.

9.1.4 Modeling strategy

We follow a modular strategy to build our models. Specifically, first we develop a model of the link reliability when transmitting one frame or several replicas of the same frame. The link model allows us to measure the probability of losing a message edition depending on the BER and the number of replicas. Besides the link model, we have developed three different models to evaluate the reliability of TSN, PTRF approach A and PTRF approach B, to which we refer as general models. We use the results of executing the link model to feed the general models. In this way, we simplify

the general models and reduce both the state space (preventing it from exploding) and the computation time.

We use PRISM to develop our models. PRISM is a probabilistic model checker that can be used to evaluate the reliability of systems (Kwiatkowska, Norman, and Parker, 2011). PRISM allows modelling the system using different stochastic formalisms, among which we find DTMCs. In PRISM the models are developed using a modeling language that allows building a DTMC as a set of *modules*, which are the main composition units. On top of modules, PRISM allows using *variables* to define the *local state* of the modules. The *global state* of the model is determined by the local state of all the modules of the model. Moreover, PRISM provides *commands* to specify the behaviour of the modules. PRISM transforms the modules that conform a model into a Markov chain and solves it analytically.

PRISM also provides a property specification language, which allows specifying which properties of the model must be checked. This language can be used to calculate the probability with which the model is in certain state and, thus, it allows expressing reliability metrics such as the one we have defined in Subsection 9.1.1.

For the sake of clarity and succinctness, we do not describe the models in detail. Instead, we next describe the most important characteristics of each model.

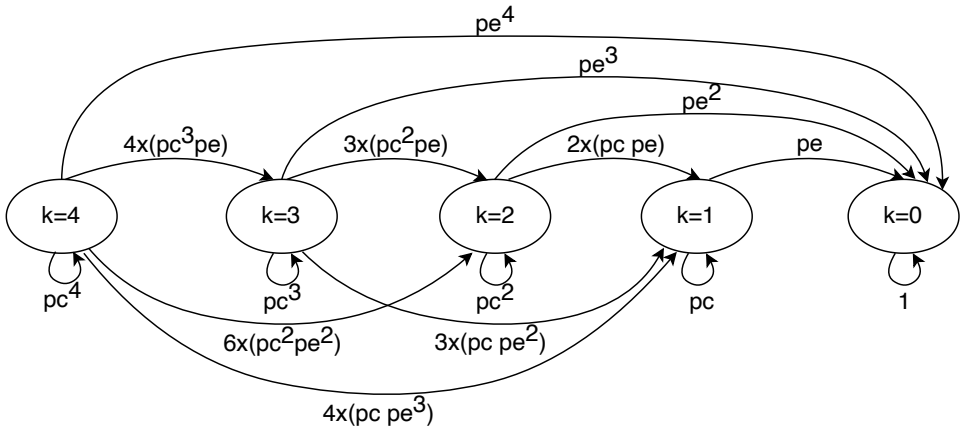


Figure 9.1: State diagram for the model of a link. The diagram represents the probabilities of receiving n out of k replicas in a link. The probability of a frame being erroneous is represented by pe, while the probability of a frame being correct is represented by pc.

Link model

As we have mentioned, we use the link model to measure the probability of receiving a certain number of replicas n out of a total of k replicas transmitted through a single link with a specific frame size and BER. We must note that, currently, our link model only allows modeling the transmission of k replicas, where $1 \leq k \leq 4$. Figure 9.1 shows the state diagram of the complete link model. As we can see, the ellipses represent 5 different states which correspond to the number of correct replicas that can be transmitted through the link in the model. The transitions between states represent the probability of receiving n replicas out of k, where $n \leq k$, while they are transmitted through a link.

The probability of receiving a subset n from a set k of replicas depends on the number of replicas that are affected by a fault and can be expressed as follows:

$$p = \binom{k}{k-n} \times (1 - pe)^n \times pe^{(k-n)} \tag{9.1}$$

where pe is the probability of a frame being affected by a fault. For further details on how we obtain this formula, the reader can refer to Chapter 7. Probability pe is calculated as follows:

$$pe = 1 - e^{-\frac{BER * framesize}{bandwidth}} \tag{9.2}$$

As we have mentioned, we use the link model to feed our general models. Thus, in order to be able to model the transmission of any number of replicas (up to four) and the loss of any combination of replicas, we have to execute this model for every value of k and n, where $0 \leq n \leq k$. On top of that, since we carry out a sensitivity analysis, we execute this model for each value of the BER and frame size that we have considered (Table 9.3). The input parameters of the link model are the following:

- Number of replicas transmitted, k. In our model, the minimum value for k is 1 and the maximum value for k is 4.

- Number of replicas received correctly, n. The maximum value of n depends on the k we define, as we cannot receive more replicas than we transmit. Thus $0 \leq n \leq k$.

- Frame size. Necessary to calculate the probability of a frame being affected by a fault, as depicted in Equation 9.2. We use the frame size values set for our sensitivity analyses: 64, 782 and 1500 bytes.

- BER. Necessary to calculate the probability of a frame being affected by a fault, as depicted in Equation 9.2. We evaluate the BER values set for our sensitivity analyses: from 1E-4 to 1E-12.

General models

As we have mentioned, we have developed three different general models, one for standard TSN, one for approach A and one for approach B. Nonetheless, all the models share the same structure presented here. Figure 9.2 shows the different modules that make up the models and how they interact. Specifically, each model has a system evaluation module, a path module, a phase module and a period module. The orange dashed lines represent time information, while the black lines represent information related to the state of the model. We next describe each module in more detail.

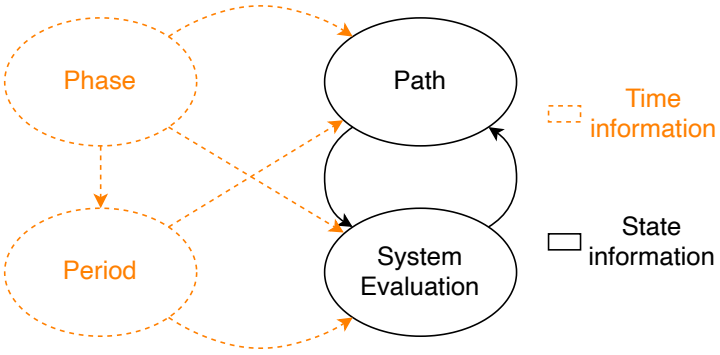


Figure 9.2: Modules in the general model. The module *system evaluation* decides whether the system has failed, the module *path* models the transmission of frames, and the modules *phase* and *period* manage the pass of time in the model.

- Module *system evaluation*. This module decides whether the system has failed according to a set of predefined rules which are applied to the local variables of the path, phase and period modules. Specifically, the system fails if *too many* streams fail; a stream fails if *too many* paths fail; and a path fails if all the replicas of a message edition are lost in the

path. The result of this module is shared with the path modules, so that the affected path modules stop evolving (i.e. making calculations) when the system has already failed. Whenever the number of failures in any of these modules reaches the specified threshold, the whole model stops.

- Module *path*. This module models the transmission of frames through a given path of a stream. Specifically, for each message edition this module models whether the transmission, forwarding and reception of each replica is correct or not for all the hops of the path. If a stream has L intended listeners there must be L path modules for said stream. This module is further divided into three blocks, namely talker, bridge and listener. Figure 9.3 shows the different blocks that constitute the path module and their relationships.

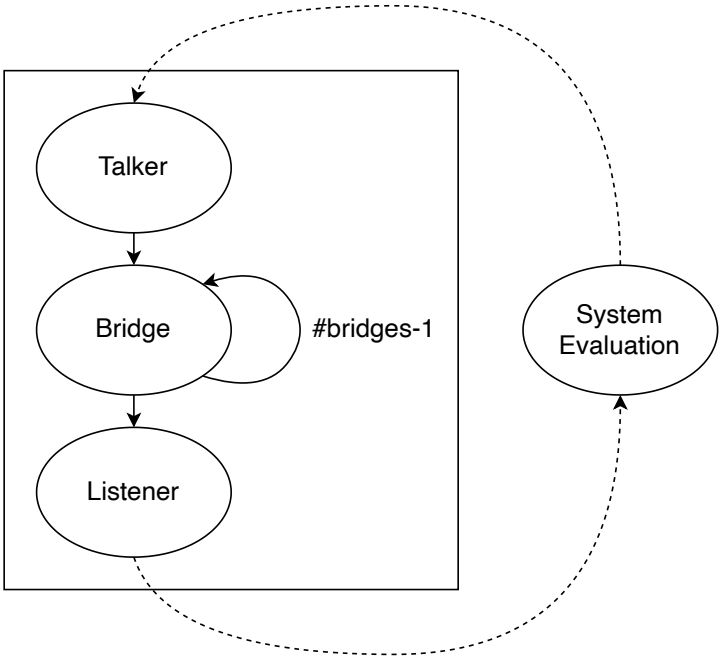


Figure 9.3: Blocks that constitute the path module and how they relate to each other and to the system evaluation module. The continuous lines represent the transmission of frames, while the dashed lines represent the exchange of status information.

As we can see in Figure 9.3, the execution of the path module starts with the talker block, which transmits a frame or the specified number

of replicas of a frame every period.

The bridge block evaluates how many frames transmitted by the talker or the preceding bridge successfully reach the destination bridge of the current hop. Specifically, the bridge uses the loss probabilities calculates using the link model previously described to carry out this evaluation. Then it evaluates if the bridge correctly forwards the adequate number of frame replicas (or a single frame in TSN) following the corresponding approach depending on the model: TSN, PTRF approach A, or PTRF approach B. In case of deciding that the bridge fails in the fowarding, the bridge block models the specific way in which this failure manifests, e.g. by forwarding more replicas than expected. The bridge block is executed as many times as bridges are in the path.

The listener block evaluates how many frames transmitted by the last bridge of the path successfully reach the listener. Then, it decides if the listener correctly eliminates the surplus replicas in case of PTRF and if it correctly delivers the frame to the upper layer. Once the path is completed, the *system evaluation* module determines whether the stream is faulty or not and, if not, the path module is executed again when dictated by the *period* module.

- Module *phase*. As we have just explained, the path module is divided intro three different blocks, namely the talker, the bridge and the listener block. Additionally, each block executes several steps, e.g., the bridge block receives, forwards and transmits frames. We must recall that we use CQF for the scheduling of frames, which means that all the steps of one block are executed within the same TAS window. The phase module dictates when the different steps of a block must be done within the TAS window.

  We must note that not all blocks execute the same number of steps. More concretely, the bridge is the block with the highest number of steps. Nonetheless, we have assumed that all the TAS windows have the same duration for a specific stream and, thus, the number of phases is the same for all the blocks of the stream. When the talker and the listener complete all the steps, they remain idle until all the phases pass, just like a real TSN device would do.

- Module *period*. This module dictates when each block of the path module is executed, i.e. it dictates the pass of time in a stream. Since we use CQF to schedule frames, each block of the path module is executed during one TAS window. The module period also counts

the number of TAS windows in a period of a stream and triggers the transmission of a frame in the talker when required.

Figure 9.4 shows the relationship between the phase and period modules. Specifically, the execution of the complete model starts initialising the period and phase modules to 0. The phase module is increased as many times as steps executes the bridge (as previously explained). Once all the steps of a block are executed, the period is increased and the phase module is reset.

If the period is higher than the number of hops, the execution of the path module is completed before the period module reaches the specified period T. In that case, the path module remains idle until the reaches the value specified for the stream period, then the period is reset and the talker creates a new frame.

9.1.5 Model testing

We check the models have been correctly implemented using two different strategies. On the one hand, we use the simulation tool available in PRISM, which allows tracing the execution of the model. First, we use the simulation tool to execute the model step by step. This allows us to detect problems and correct them so that the models operate as intended. Second, we use the simulation tool to execute several steps of the model at once to detect interlocks, i.e., we execute the models in sets of 1,000 steps at a time, which allows exploring the model faster.

On the other hand, we have executed the models using different configurations. Specifically, we have assessed the proper execution of the model using (i) a single stream with a single path; (ii) a single stream with up to five paths; (iii) up to three different streams, each with up to three paths and (iv) replicated streams with one path each, allowing us to model space redundancy.

The results of the aforementioned tests allowed us to refine the implementation of the models to eliminate mistakes, improve their efficiency in terms of execution time and validate their correctness.

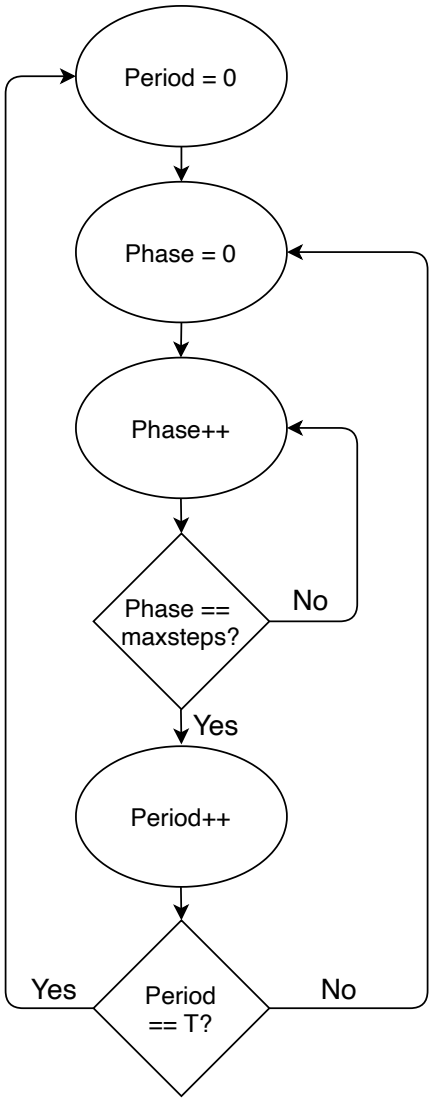


Figure 9.4: Execution of the phase and period modules to dictate the pass of time in the models.

9.2 Sensitivity analyses

As we have mentioned, the aim of this work is to prove that we can increase the reliability of TSN networks by using PTRF to tolerate temporary faults. To that we compare the reliability achievable by TSN and PTRF consider-

ing the possibility oftemporary faults under different operating conditions. Specifically, we carry out a parametric sensitivity analysis where we vary a series of parameters and study their impact on the reliability. Specifically, we compare standard TSN, approach A and approach B of PTRF. We do not evaluate the reliability achievable by approach C as it is a specific case of approach B that aims at improving the efficiency of PTRF, not its reliability.

As we have said, reliability is defined as the probability of a system or subsystem to provide a correct continuous service for a certain amount of time called mission time. Furthermore, as we explain in 9.1.1, our reliability metric is the probability that for each stream a subset of listeners receives at least one copy of each message edition during a given interval of time called mission time. As we have also explained, in this section we only evaluate the reliability of a single stream with a single listener.

In this section we describe the parameters of the models, how we configure them and we discuss the results of the analyses.

9.2.1 Parameters of the models

In order to carry out the comparison between TSN, approach A and approach B, we must first define a case of reference that serves as the baseline to compare the results of the models, as well as to study the impact of the different parameters on reliability. To that, we assign a reference value to each parameter and we define the range of values that each parameter takes. We must note that the case of reference must have reasonably realistic

Table 9.3: Parameters tuned in the the sensitivity analysis. For each parameter we specify the range of values that we have evaluated, its value in the case of reference and its meaning.

Parameter	**Range**	**Reference**	**Meaning**
Mission time	10h	**10h**	The time that the system is expected to operate in a continuous manner. We use 10 hours in all the experiments as it is typically assumed for evaluating the reliability of critical applications such as throttle-by-wire (Morris and Koopman, 2005).
TAS window	1.3ms	**1.3ms**	The duration of the TAS window in milliseconds. Since we assume that frames are scheduled using CQF, each TAS window must last enough time for a frame to be transmitted, propagated, received and forwarded. Thus, to decide on the TAS window size we have taken into account the transmission and reception time, the propagation time and the forwarding time of a frame through a link and a bridge. These values have been mathematically estimated or measured using a real TSN bridge (*MTSN Kit: a Comprehensive Multiport TSN Setup.*). This parameter is fix in all the experiments.
Period	15 TAS windows	**15**	The number of TAS windows that elapse between the transmission of frames. This parameter is fix in all the experiments.
BER	{1E-4 .. 1E-12}	**1E-10**	The bit error rate of the links. We consider values from 1E-4 considered in highly critical applications such as space missions, to 1E-12 considered for utility communications. We use 1E-10 as reference as it is the most pessimistic assumption while still considering critical applications such as automotive.
Frame size	{64, 782, 1500}B	**782**	The size of the frames transmitted through the stream in bytes, being 64 the minimum size, 782 the medium size and 1500 the maximum size for Ethernet frames. We use 782 as reference to compare the impact when using smaller and larger frames.
# of replicas	{1 .. 4}	**2**	The number of replicas transmitted by PTRF-enabled components. This value is always 1 in TSN and we consider up to 4 replicas as our results show that reliability converges with this number of replicas. Our case of reference is 1 replica for TSN and 2 replicas for PTRF as this is the lowest number of replicas that PTRF can transmit while still providing time redundancy.
# of bridges	{2 .. 10}	**6**	The number of bridges in the path between the talker and the listener. We consider from 2 to 10 bridges. We select 6 bridges for our case of reference as this is the maximum number of bridges for which TSN ensures tight synchronisations.

values for the parameters, while still being pessimistic for PTRF. In this way, we do not bias the results in favor of PTRF.

Table 9.3 shows the parameters that we have tuned during the analysis, the range of values that we have evaluated as part of the sensitivity analysis, the value we have assigned for the case of reference and a brief description of the parameter. The case of reference is highlighted in bold.

As we can see, our models have 7 different parameters that can be tuned during the analysis. We have fixed 3 parameters, namely the mission time, the duration of the TAS window and the period of the stream. On the other hand, we vary and thus evaluate the impact of four different parameters, namely the BER, the frame size, the number of replicas and the number of bridges.

9.2.2 Results

We have executed an experiment for each combination of parameters, which has resulted in 2,146 experiments. In each one of these experiments, we have modelled the transmission of 1,800,000 frames. We next analyse the most relevant results obtained, starting by the case of reference and moving to the analysis of the impact that each parameter has on the reliability.

Case of reference

As we have explained at the beginning of this Section, we have established a case of reference to compare the different solutions and the impact that the different parameters have on the reliability. We must note that, as we can see in Table 9.3, the number of replicas transmitted by PTRF in the case of reference is 2, as this is the lowest number of replicas that PTRF can use while still providing time redundancy. Later on, we show how transmitting a single frame when using PTRF affects the reliability.

Table 9.4 shows the results obtained in the case of reference for TSN, approach A and approach B. As we can see, the results obtained with all the solutions are in the same order of magnitude. However, TSN can achieve a reliability of 85.84%, while approaches A and B both reach a reliability of 99.99% when transmitting just two replicas.

If we now focus on PTRF, we can see that approach A is more reliable than approach B. This is because in approach A only the talker and the listener are PTRF-enabled devices and bridges are standard TSN devices with lower

Table 9.4: Reliability of TSN, approach A and approach B in the case of reference.

TSN	Approach A	Approach B
0.858467899141365	**0.9999**58504043796	**0.9999**43599679439

failure rate; unlike approach B where all the devices are PTRF-enabled. Since the BER in the case of reference is low, the contribution of the failure rate of the devices in the unreliability is higher than in networks with larger BERs and, thus, the unreliability of bridges is more noticeable.

From this first set of results we can extract several conclusions. First, we can conclude that the use of time redundancy has a positive impact on the reliability achievable in networks that suffer temporary faults. Second, we see that the benefits of time redundancy are even noticeable in networks with a relatively low BER, only 1E-10, and with a relatively small network, only 6 bridges. Finally, we can see that approach A is a more adequate solution than approach B in networks with low BER, as the contribution of the failure rate of the components increases the unreliability in the later.

In the following subsection we analyse how the BER affects reliability and we pay special attention to higher BER values.

Bit error rate

Since we are evaluating the impact of temporary faults in the network, the first parameter we want to study is the BER of the links. The BER of the links severely affects the reliability of the communication subsystem, so we have selected a wide range of BER values to carry out our analysis, which go from 1E-12 to 1E-4. Figures 9.5 and 9.6 show the evolution of the reliability achievable by TSN, approach A and approach B when modifying the BER.

As expected, we see that the reliability is severely affected by the increase in the BER. We can also see that the impact is significantly higher in TSN than in approaches A and B when the BER is the same. This is because TSN does not count with any time redundancy mechanism to tolerate the faults in the links, while approaches A and B do. Furthermore, these results also show that approaches A and B can reach reliability values over 99.9% for BER values up to 1E-8 and over 99.8% for a BER of 1E-7 when transmitting just two replicas. As the BER increases, we see that the reliability drops dramatically even when using PTRF. This is because 2 replicas are no longer

enough to tolerate the great number of faults that affect the links in those cases.

If we pay close attention to the results of approach A and B, we see that the increase in the BER has a higher impact in approach A than in approach B. This is because in approach A there is a single set of replicas that traverse the network, so whenever a replica is affected by a fault in a link it is lost in the whole path. On the contrary, in approach B bridges create a new set of replicas in each hop, so even if a replica is affected by a fault in a link the following bridge transmits two replicas again. Therefore, we can conclude that in networks with high BER values, approach B is a more adequate solution as it can provide higher reliability values than approach A using the same number of replicas and, thus, the same bandwidth.

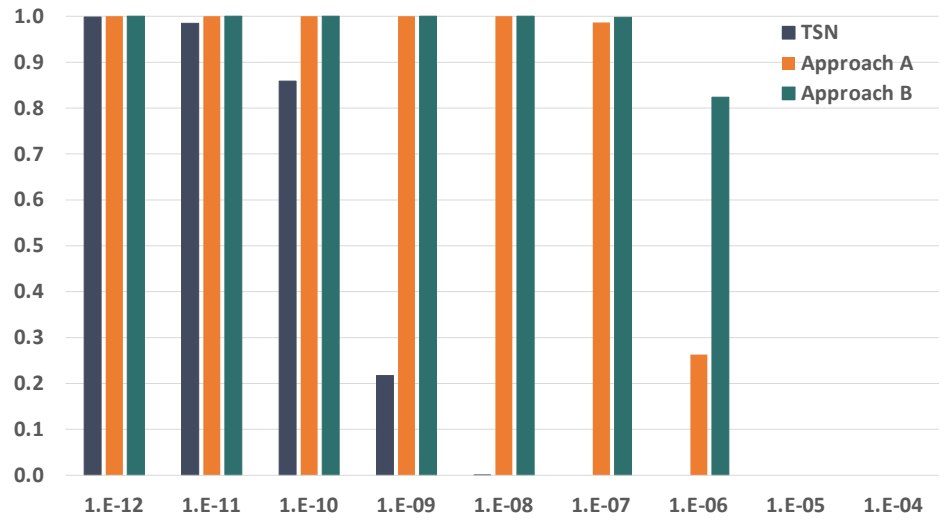
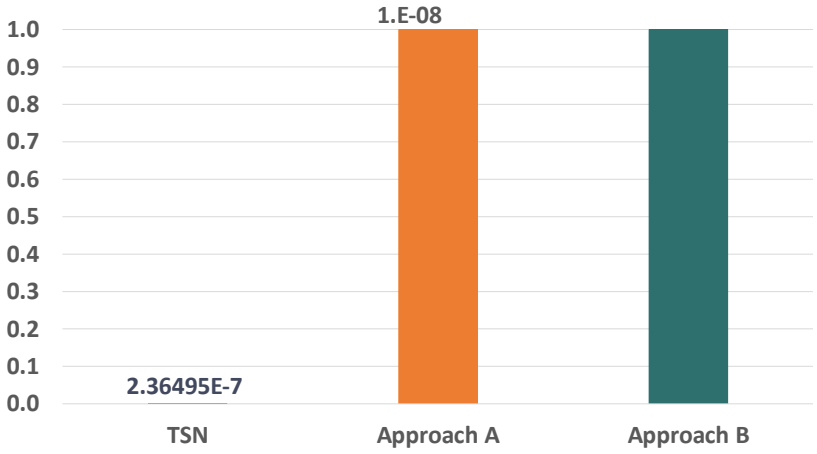
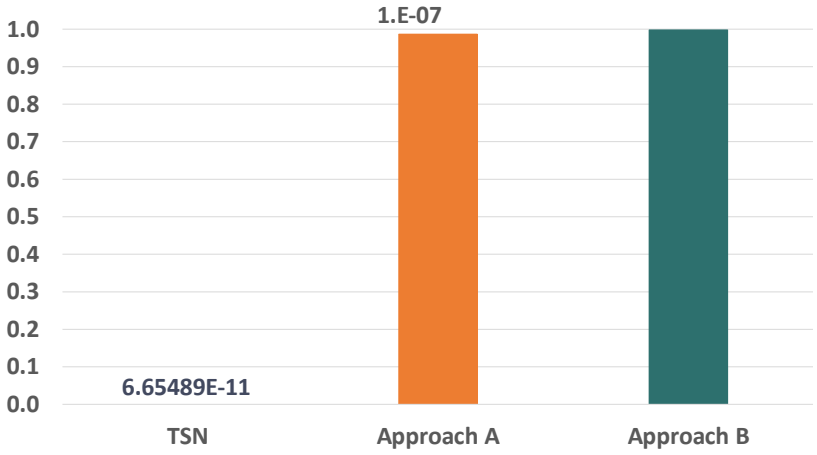
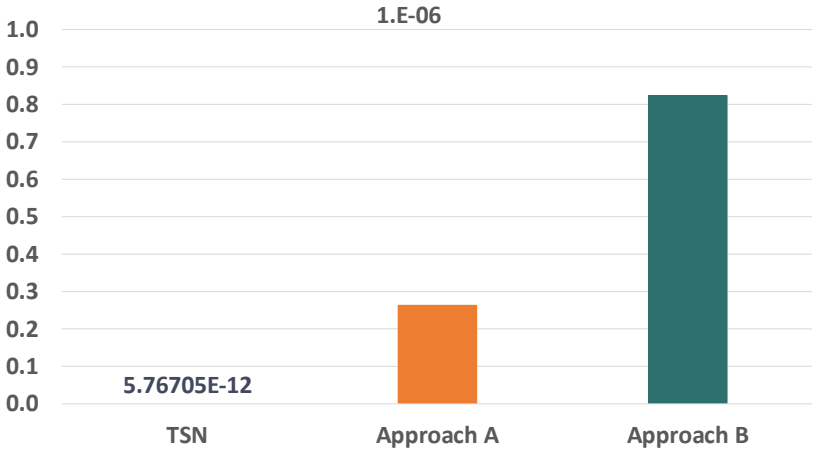


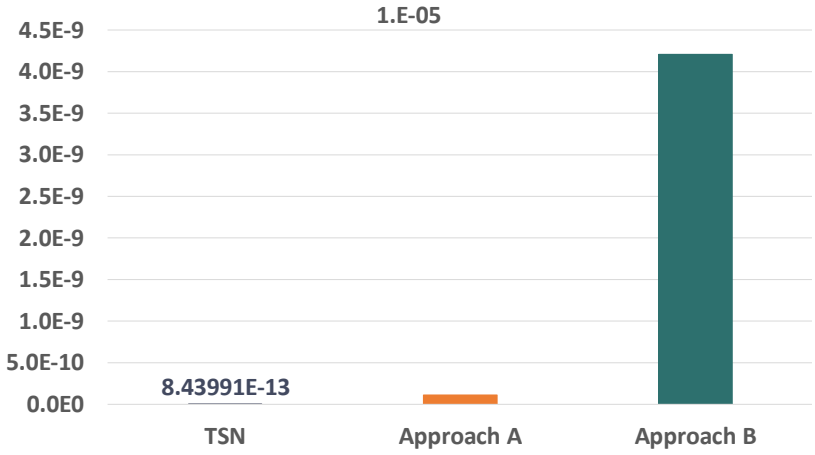
Figure 9.5: Reliability achievable when varying the BER from 1E-12 to 1E-4. TSN is represented in gray, approach A in orange and approach B in green. The X axis represent the BER values, while the Y axis represent the reliability.

These results confirm the benefits of using time redundancy to tolerate temporary faults and show that in many cases a low level of redundancy can provide benefits. Nonetheless, we can see that two replicas are not enough


1.E-08
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
2.36495E-7
TSN
Approach A
Approach B


1.E-07
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
6.65489E-11
TSN
Approach A
Approach B


1.E-06
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
5.76705E-12
TSN
Approach A
Approach B


1.E-05
4.5E-9
4.0E-9
3.5E-9
3.0E-9
2.5E-9
2.0E-9
1.5E-9
1.0E-9
5.0E-10
0.0E0
8.43991E-13
TSN
Approach A
Approach B

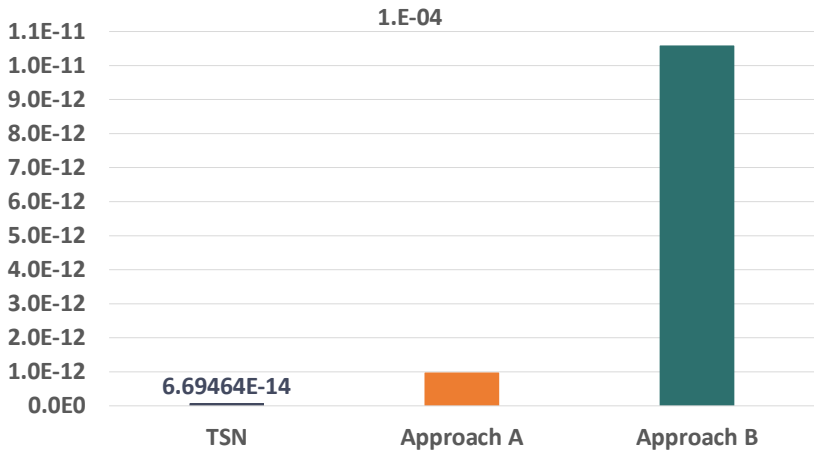


Figure 9.6: Zoom in the reliability achievable by TSN, approach A and approach B for BERs from 1E-8 to 1E-4. TSN is represented in gray, approach A in orange and approach B in green. The X axis represent the network type, while the Y axis represent the reliability.

to reach the reliability levels required in highly critical applications, e.g. 99.999% for throttle-by-wire applciations (Morris and Koopman, 2005).

Finally, the huge difference in the reliability achievable when moving from a BER of 1E-7 to 1E-6 can let us envisage that the BER value can impact the results obtained when studying how the frame size, the number of replicas and the number of bridges affect reliability. Thus, if we only study the behaviour of PTRF in scenarios with BER of 1E-10, we could be masking important interactions between parameters. Therefore, we have expanded the case of reference to include a BER value of 1E-6, which is high enough to see how PTRF behaves in harsh environments.

Frame size

A temporary fault in the communication subsystem only results in an error if a frame is being transmitted while the fault happens. Therefore, we can assume that the size of the frame impacts the reliability, as larger frames occupies the communication subsystem for a longer time than smaller frames. The following experiments allow us to quantify this impact. Figure 9.7

shows the reliability achievable by TSN, approach A and approach B when transmitting frames of 64, 782 and 1500 bytes in two cases, namely, our case of reference with BER equal to 1E-10 and another case with a BER of 1E-6.

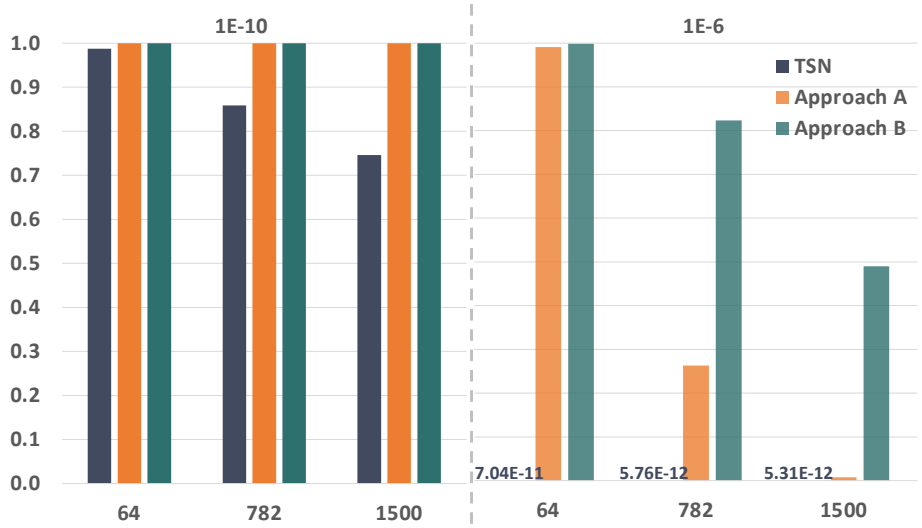


Figure 9.7: Reliability when varying the frame size. The left-hand side of the figure shows the reliability for a BER of 1E-10, while the right-hand shows the reliability for a BER of 1E-6. In both cases, TSN is represented in gray, approach A in orange and approach B in green. The X axis represent the frame sizes in bytes, while the Y axis represent the reliability.

If we look at the results obtained when the BER is 1E-10, we can see that the frame size has an important impact on the reliability achievable by TSN, which drops more than a 20% with larger frames. In contrast, we can see that in this case the impact for approaches A and B is negligible. This shows that, when the BER is low, using the minimum level of temporal redundancy (number of proactive replicas, k, equal to 2) is enough to compensate the negative impact of using large frames. Moreover, the results of approach A further indicate that bridges do not need to regenerate replicas at their output ports (as done in B) to compensate this negative impact; at least when the path does not include more than 6 bridges.

This benefit of temporal redundancy is corroborated, to some extent, when

we consider a BER of 1E-6. However, in this case, the reliability of A and B drops around a 98% and a 50% respectively when the frame size is 1500 bytes instead of 64. This indicates that when the BER is high, it is necessary to increase the number of time replicas to keep a high level of reliability; specially when using approach A instead of B. Later on we demonstrate this assertion (for the reference frame size of 782 B) when analyzing the impact of the number of replicas.

As a general recommendation, we can say that in critical applications that operate in harsh electromagnetic environments, it may be convenient to reduce the size of frames whenever possible. Nonetheless, this is not always possible. In those cases, approach B is a more adequate solution than approach A, but even with approach B the reliability decreases dramatically. Thus, in such environments larger frames should have a higher level of redundancy than shorter ones to provide adequate levels of reliability.

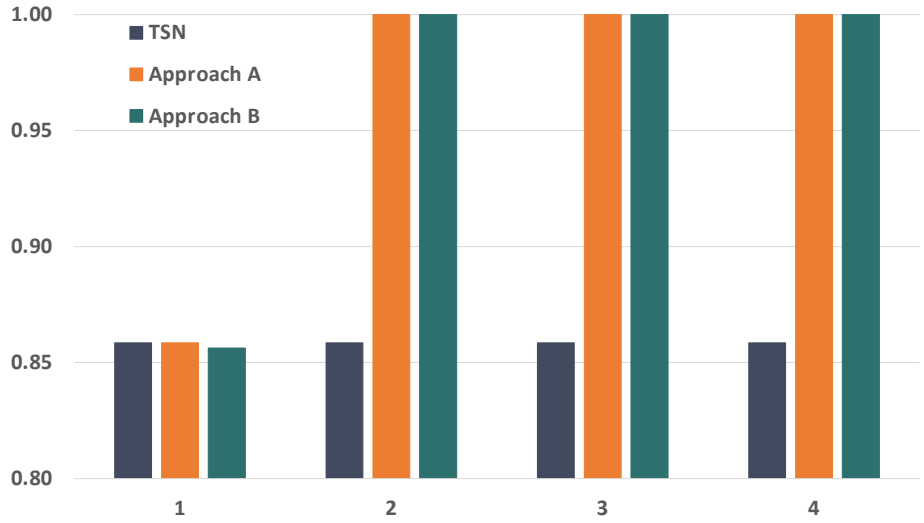
Number of replicas

It is expected that the number of replicas transmitted for each frame impacts the reliability achievable by the communications subsystem. Nonetheless, we need to note that the level of time redundancy can negatively impact other aspects of the system, such as the end-to-end delay, jitter and bandwidth consumption, as we show in Chapter 8. Therefore, choosing a level of temporal redundancy that provides the network with the intended reliability, while ensuring an adequate quality of service is of utmost importance.

Figure 9.8 shows how the number of replicas transmitted impacts the reliability achievable by TSN, approach A and approach B when the BER of the network is 1E-10. Note that the number of replicas in TSN is always one, as it does not count with any time redundancy mechanism.

If we focus on Figure 9.8a what we first see is that the reliability is improved by almost a 15% when using time redundancy in this scenario. However, if we focus on the first three columns, when replication is not used, we can

see that the reliability actually drops when using approaches A and B to transmit non-replicated frames. This is because PTRF-enabled devices have a higher temporary failure rate (1.5E-4) than standard TSN devices (1E-4). Thus, when no redundancy is used the probability of losing frames due to temporary faults affecting the bridges or end-devices is higher in PTRF than in TSN. Furthermore, this is more noticeable in approach B than in approach A, as the number of PTRF-enabled devices is higher in the former.

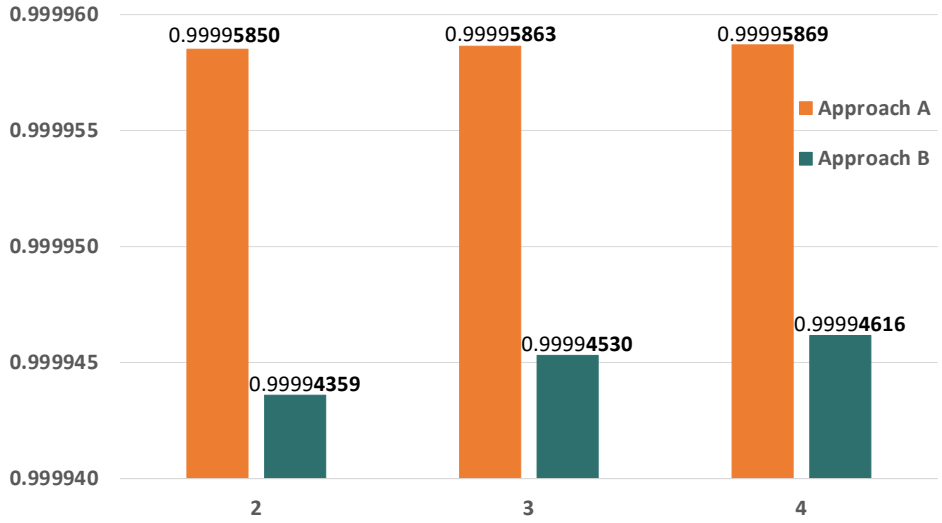


(a) Reliability achievable by TSN, approach A and approach B when varying the number of replicas. In TSN the number of replicas is always 1.

In fact, if we focus on Figure 9.8b we can see that the increased failure rate of PTRF-enabled devices limits the reliability benefits provided by B when compared to A. This is because PTRF is designed to tolerate link faults only and, thus, it cannot tolerate most faults that occur in end-systems or bridges. Therefore, in networks where the BER is low, approach A is a more adequate solution than approach B to tolerate temporary faults. In any case, we see that the reliability is always above 99.99% for both approaches.

On the other hand, Figure 9.9 depicts the reliability achievable by approach A and approach B when transmitting different number of replicas in a network with a BER of 1E-6. Please note that we do not include $k = 1$ in this plot as the reliability achievable by the three solutions in the absence of redundancy is around 5.76E-12. As we can see, the reliability increases significantly with the number of replicas transmitted. As indicated in the section devoted to analyze the frame size, we also see that transmitting only two replicas is no longer enough to achieve high reliability, specially in approach A, where the reliability is only a 26.31%, compared to the 82.36% of approach B.

Furthermore, we can see that the comparison of the reliability benefits



(b) Zoom in the reliability achievable by approach A and approach B when varying the number of replicas.

Figure 9.8: Reliability achievable when varying the number of replicas in a network with a BER of 1E-10. TSN is represented in gray, approach A in orange and approach B in green. The X axis represent the number of replicas, while the Y axis represent the reliability.

provided by A versus B is the opposite of what we have just described when considering a low BER of 1E-10. Specifically, we can see that when the BER is high approach B provides higher reliability than approach A, even with a low number of replicas. This is because when the BER is high, its contribution to the unreliability is higher than the contribution of the failure rate of devices, i.e. it is more likely that a frame is affected by a fault in the link than in a device. Thus, even though approach B has more PTRF-enabled devices with higher failure rate than approach A, approach B is a more adequate solution than approach A in networks where the BER is high.

Finally, please recall that these results for a BER of 1E-6 demonstrate (for a frame size of $\leq$ 782 B) what we anticipated in the section that analyzes the

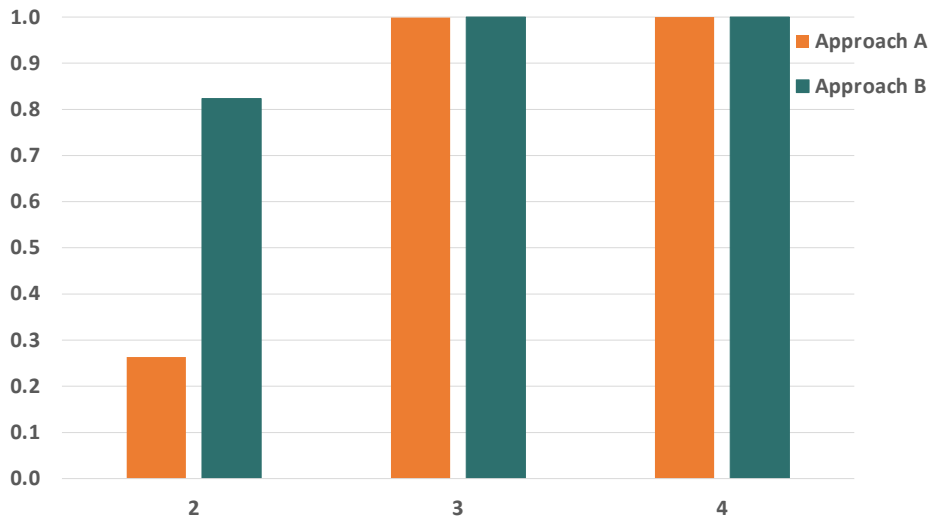


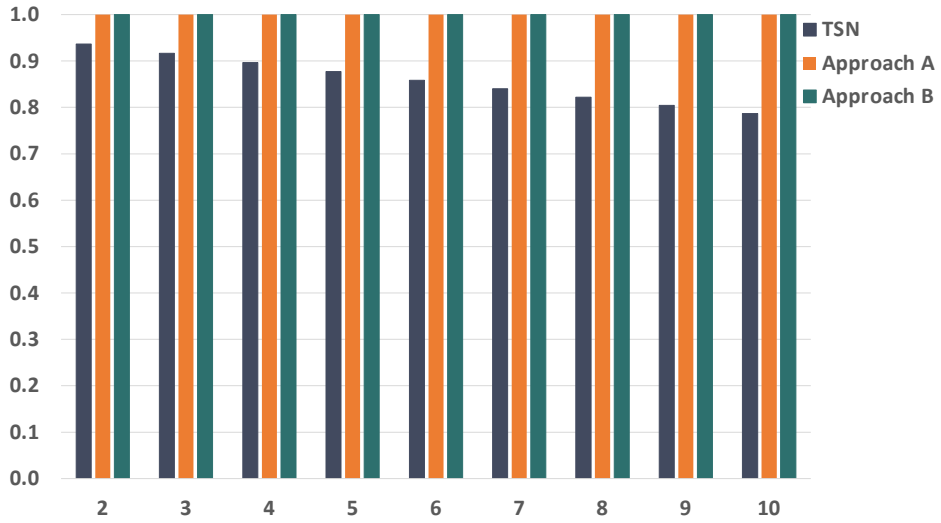
Figure 9.9: Reliability achievable when varying the number of replicas in a network with a BER of 1E-6. Approach A is represented in orange and approach B in green. The X axis represent the number of replicas, while the Y axis represent the reliability.

impact of the frame size, i.e. that to keep a high level of reliability when the BER is high, it is necessary to increase the number of time replicas; specially when using approach A instead of B.

Number of bridges

The size of the network is expected to impact the reliability of streams, as in a larger path there are more chances for a frame to be affected by a fault. Figure 9.10 shows the reliability achievable by TSN, approach A and approach B in networks of different sizes and a BER of 1E-10.

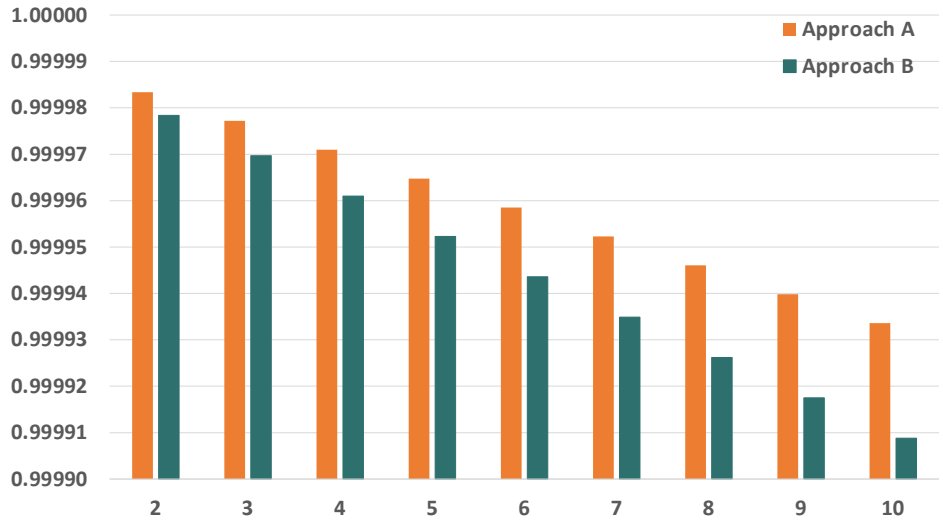
As expected, we can see that the reliability decreases as the network size increases. If we look at Figure 9.10a we see that the impact is higher in TSN than in approaches A and B, as TSN does not count with time redundancy. In any case, we can see that the reliability drops around a 15% for TSN



(a) Reliability achievable by TSN, approach A and approach B when varying the number of bridges.

when moving from 2 to 10 bridges. If we compare this results to the ones obtained previously, we can conclude that in TSN networks with low BERs the impact of the network size is lower than the impact of other parameters.

If we now focus on Figure 9.10b we see that the reliability using approach B is lower than using approach A. We also see that the difference in terms of this negative impact between both approaches increases with the number of bridges. To understand this effect please recall that the bridges of approach B are PTRF-enabled devices and, thus, they have a higher failure rate than the bridges of approach A. Also recall that this does result in approach A being more reliable than approach B when the BER is low. In this sense, what we observe in Figure 9.10b indicates that, when the BER is low, the negative impact of using PTRF-enabled bridges instead of TSN ones increases with the size of the path, as a larger path includes more bridges. In any case, we must note that the reliability achievable by PTRF is above the 99.99%, and higher than in TSN, when the BER is 1E-10.



(b) Zoom in the reliability achievable by approach A and approach B when varying the number of bridges.

Figure 9.10: Reliability achievable when varying the number of bridges in a network with a BER of 1E-10. TSN is represented in gray, approach A in orange and approach B in green. The X axis represent the number of bridges, while the Y axis represent the reliability.

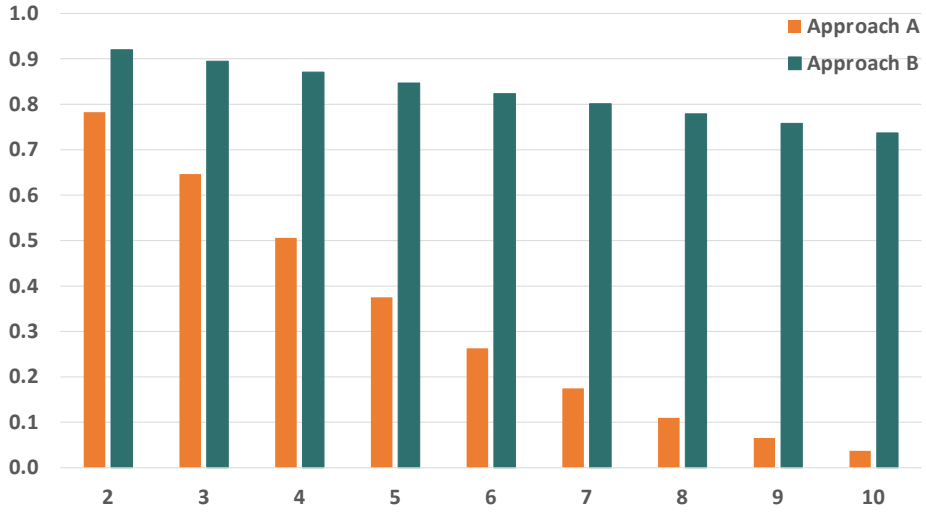


Figure 9.11: Reliability achievable when varying the number of bridges in a network with a BER of 1E-6. Approach A is represented in orange and approach B in green. The X axis represent the number of bridges, while the Y axis represent the reliability.

Finally, Figure 9.11 shows the reliability achievable by approach A and approach B in networks of different sizes with a BER of 1E-6. We must note that we do not plot the reliability of TSN in this scenario as it is too low to be depicted together with approach A and B (under 5.8E-12). Again, we can see that the reliability is affected by the size of the network. Furthermore, we see that the higher the BER is, the higher the impact will be.

However, conversely to what we observe in Figure 9.10b, we see that approach B is a more adequate solution than approach A in networks with higher BER values. In fact, the difference of reliability of approach B when compared to approach A increases with the size of the path. This does not only reinforce the result we showed before about the higher reliability of B when compared to A when the BER is high; but it also shows that, when the BER is high, the positive impact of the additional degree of time redundancy provided by PTRF-enabled bridges further counterbalances the negative impact of their higher failure rate as the path size increases. Specifically, we

see that the reliability achievable by approach B drops an 18% while the reliability achievable by approach A drops as much as a 74%.

Summary

These analyses prove the final part of our thesis statement: *We can increase the reliability of multi-hop networks based on TSN standards, which support real-time and operational flexibility, by using proactive time redundancy to tolerate temporary faults in the links in a way that is suitable for the RT response of the network.* Indeed, we can conclude that time redundancy alone can significantly increase the reliability of the communications in the presence of temporary faults.

In any case, we have also seen that the reliability achievable by any system or subsystem is affected by a myriad of aspects. Therefore, there is not a single solution that can fulfill the reliability needs of any system. Instead, an adequate specific solution must be used for each system. This solution must be configured appropriately in order to provide the required level of reliability, specially in critical systems that operate in harsh environments. The analyses herein presented can help designers in making better decisions when building the communication subsystem of TSN networks.

On the one hand, we have corroborated the idea that the BER is determinant in the reliability achievable by TSN, approach A and approach B. Furthermore, we have seen that approach A is a more adequate solution than approach B to provide high reliability in networks with low BERs. Not only approach A provides a higher reliability level, but it is also a cheaper solution as it uses COTS TSN bridges. On the contrary, approach B can reach significantly higher levels of reliability than approach A in networks with high BERs. Thus, in these cases using approach B is more adequate than using approach A, even if the cost of the network is higher.

Furthermore, we have seen that the frame size has an important impact on reliability, which increases significantly as the BER increases. Therefore, the size of the frames should be reduced whenever possible, e.g., preprocessing the information in the source. If this is not possible, streams that convey larger frames should have a higher level of redundancy than those that convey smaller frames when high reliability is required.

Finally, we have studied how the network size impacts reliability. Specifically, we have seen that the reliability decreases as the network size increases. We have also seen that this negative impact increases with the BER.

Chapter 10

Conclusions and Future Work

This Chapter presents the conclusions of this dissertation and a series of possible lines of work for the future.

10.1 Conclusions

In the last years industry and academia have shown a growing interest in developing novel and complex applications such as Industry 4.0, autonomous vehicles or efficient energy management. These applications create new challenges in the design of the infrastructures that support them. More concretely, these infrastructures must be distributed, reliable, real-time and flexible, which means that their communication networks must be *multi-hop* and must provide *reliability*, *real-time flexibility* and *operational flexibility*.

Ethernet is the network technology that has drawn the most attention over the last 20 years, because of its benefits in terms of bandwidth, scalability, widespread know-how and IP compatibility. Nonetheless, Ethernet was not designed to fulfil the requirements that we have previously discussed and, thus, it presents a series of limitations. As a result, a myriad of Ethernet-based solutions have been proposed to overcome these limitations. Nonetheless, most of these solutions addressed only one or a subset of Ethernet's limitations. On top of that, these solutions present their own serious limitations in terms of interoperability.

The TSN TG from IEEE has been been develop a series of Ethernet

standards which can be used to build communication networks that fulfil all the requirements previously discussed. Nevertheless, TSN still presents limitations in terms of the fault tolerance capabilities that it can provide to the network, which are critical in order to reach a high reliability while, at the same time, preventing an unnecessary cost increase and preserving the real-time response. In particular, TSN does not provide any mechanisms specifically designed to tolerate temporary faults in the links of the communication infrastructure, even though this is the most common type of fault. Thus, temporary faults in TSN can only be tolerated using space redundancy or higher layer protocols for time redundancy.

Nevertheless, these solutions are not the most adequate for a series of reasons (see Section 1.2). On the one hand, using space redundancy to tolerate temporary faults is not a cost-effective solution as the number of redundant independent paths increases with the number of simultaneous temporary and permanent faults that want to be tolerated. Moreover, when temporary faults occur the system cannot take full advantage from spatial redundancy to tolerate permanent faults. On top of that, when all the redundant paths except one are affected by permanent faults it is no longer possible to tolerate temporary faults due to the redundancy attrition.

On the other hand, higher layer protocols for time redundancy have traditionally been based on automatic repeat request (ARQ) techniques. ARQ techniques rely on the transmission of acknowledgement (ACK) or negative ACK (NACK) messages and timeouts to trigger the retransmission of lost frames. Nonetheless, ARQ-based solutions are non-deterministic in terms of end-to-end delay and bandwidth consumption as temporary faults are unpredictable. This non-determinism also leads to a high jitter, as the end-to-end delay varies depending on the number of retransmissions required to successfully deliver frames. Furthermore, in the worst-case scenario ARQ solutions worsen the utilization of the network, as additional bandwidth is required to transmit ACK and NACK messages. Finally, since ACK and NACK messages can also be affected by faults, ARQ-based solutions introduce new complex scenarios involving faults, which are harder to tolerate.

For all the previous reasons, we propose to use proactive time redundancy to tolerate temporary faults in the links. Specifically, we have proposed the Proactive Transmission of Replicated Frames (PTRF) mechanism, which consists in transmitting several copies of each frame (i.e. frame replicas) in a preventive manner, in order to increase the probability of at least one copy being delivered to its intended receiver even in the presence of temporary faults.

The target of this dissertation is to prove the following thesis statement:

> *We can increase the reliability of multi-hop networks based on TSN standards, which support real-time and operational flexibility, by using proactive time redundancy to tolerate temporary faults in the links in a way that is suitable for the RT response of the network.*

To prove our thesis we have designed, validated, implemented and evaluated PTRF. In fact, we have designed three different approaches of PTRF which differ from each other in two aspects, (i) which are the devices that carry out the replication and the later elimination of replicas and (ii) how to calculate the number of frame replicas that must be transmitted. During the design of PTRF we have established the following requirements to ensure that the thesis statement can be proved (see Section 6.1):

- R1: the mechanism must be fully compatible with standard devices. That is, PTRF-enabled devices and standard TSN devices must be able to coexist in the same network.
- R2: the mechanism must be easily integrable with existing standards.
- R3: the mechanism must not imply significant modifications of standard devices.
- R4: the mechanism must not have a high memory consumption as bridges have a limited amount of memory.
- R5: the mechanism must be flexible enough to be used in virtually any network, even those for adaptive systems.

Requirements R1 to R4 are key to ensure that we can use PTRF in TSN networks, while keeping the real-time and operational flexibility that they provide. Thus, all the approaches proposed in this dissertation are designed to meet these requirements. Regarding requirement R5, only one of our approaches is designed to meet it. Nonetheless, this requirement is not necessary to prove the thesis statement, as it is intended to *increase* the operational flexibility of the network.

In order to properly design the approaches of PTRF we have also defined the fault types that we want to tolerate in the links and the failure modes that the links would exhibit in the presence of such faults (see Section 6.2).

Specifically, our fault model covers temporary non-malicious hardware faults in the links and failures manifest as omissions. We then designed our three approaches of the PTRF mechanism, to which we refer as approach A, B and C (see Section 6.3). Approach A implements PTRF only in the end-systems, meaning that only talkers replicate frames and only listeners eliminate surplus replicas. Conversely, approaches B and C implement PTRF in all the devices, including bridges. The difference between approach B and C lies in the number of replicas sent by each device. Specifically, in B all devices send the exact same number of replicas through all their ports, while in C each device can send a different number of replicas through each port, providing an extra degree of flexibility (requirement R5).

In order to validate the design of the three approaches and check the feasibility of PTRF, we have developed a simulation model of PTRF (see Section 7.1). We have developed this model using OMNeT++ on top of an already existing preliminary TSN simulation model called TSimNet. We developed a series of modules that allowed us to simulate the three approaches. On top of that, we have also analysed the number of fault scenarios that each approach can tolerate and we have compared the results of our analysis to the results obtained with the simulation to validate the (see Section 7.2).

Once we had validated our design and checked the feasibility of the approaches, we could start the implementation of PTRF in a real TSN prototype (see Section 8.1). Specifically, PTRF is implemented on an already existing TSN bridge developed by the company SoC-e. This implementation was carried out by the company and is currently implemented as a firmware that can be executed in the company's products to implement both TSN end-systems and bridges. On top of that, we have also colaborated with SoC-e to develop a custom tool to inject faults in TSN networks, which allows us to evaluate PTRF in the presence of faults.

We have built a real prototype of a network using the PTRF-enabled bridges and the fault injection devices previously discussed (see Section 8.2). We have used this prototype to evaluate the impact that implementing PTRF has on performance from three different perspectives: the end-to-end delay, the jitter and the bandwidth consumption. To do so we have carried out a sensitivity analysis to study the behaviour of PTRF when operating in networks with different characteristics (see Section 8.3). We have also done this sensitivity analysis for TSN (without PTRF), as this allows us to establish the baseline to which we compare PTRF. Regarding PTRF, we only analyse approach A and approach B, as the implementation of approaches B and C is identical.

These experiments have allowed us to conclude that PTRF is a suitable solution to provide time redundancy to real-time TSN networks for the following reasons:

- The impact of PTRF in the end-to-end delay is negligible in the absence of faults. Furthermore, the maximum end-to-end delay in the presence of faults is also acceptable for real-time networks, with approach A providing the lowest upper bound.
- The jitter introduced by implementing PTRF is always low for both approaches, while the jitter experienced when losing frames increases for approach B when frames are large and the number of replicas is high.
- The impact of implementing PTRF in bandwidth is low compared to the impact that creating new frames has, as replicating frames requires low processing times.

These experiments have also allowed us to prove part of our thesis statement, as they show that PTRF can be implemented while keeping the real-time guarantees of TSN networks. On the other hand, the implementation of approach A is a proof of the compatibility of PTRF with standard TSN devices, as the PTRF traffic produced by talkers traverses COTS TSN bridges.

Finally, in order to prove that PTRF can increase the reliability of TSN networks we have modelled TSN and PTRF using the PRISM probabilistic model checker (see Section 9.1). Moreover, we have used the models to carry out a parametric sensitivity analysis that allowed us to quantify how several dependability-related aspects of PTRF affect the reliability when the possibility of temporary faults is considered (see Section 9.2). Specifically, we have developed one model for TSN, one for approach A and one for approach B. We did not develop any specific model for approach C as it is actually a specific case of approach B that is devised to yield benefits in terms of bandwidth.

The parameters for which we have considered a range of realistic values in our models in order to study their impact on reliability are: the bit error rate (BER) of links (from 1E-4 to 1E-12), the size of frames (64, 782 and 1500 bytes), the number of replicas transmitted (from 1 to 4) and the number of bridges (from 2 to 10).

The analyses that we have carried out allowed us to draw the following conclusions:

- PTRF increases the reliability achievable by TSN networks when the possibility of temporary faults is considered, proving our thesis. This applies no matter the values of the different parameters specified above (meaning that for any combination of parameters, TSN+PTRF is more reliable than TSN alone).

- The BER is decesive in the reliability achievable by the network in the presence of temporary faults. Furthermore, it affects the behaviour of PTRF, as approach A is a more adequate solution than approach B for low BERs, and approach B is better than approach A for high BERs.

- Transmitting larger frames reduces the reliability achievable by the network, specially when the BER is high. In order to keep the same level of reliability, the level of time redundancy when transmitting large frames must be higher than for shorter frames.

- Increasing the size of the network decreases the reliability. Furthermore, the impact is higher using approach A than approach B.

For all of the above, we can conclude that the work presented in this dissertation has proved our thesis, by means of an adequate design, simulation, implementation, experimentation, modeling and reliability evaluation.

10.2 Lines for future work

Next we list a series of potential tasks identified during the development of this dissertation that represent lines of future work.

- Evaluate how approach C can increase the reliability of TSN networks while minimising the impact of PTRF on performance. Note that approach C is designed so each network device can send a different number of replicas through each port. In this way the level of redundancy can be adapted to the unreliability of the link, in such a way that the consumption of bandwidth is optimised in each link while still providing an adequate level of reliability.

- Develop mechanisms to support the on-line configuration of PTRF in order to adapt the number of replicas transmitted to the changes in the unreliability of the environment.

- Integrate PTRF with the space redundancy mechanisms of TSN in a real prototype to experimentally evaluate their interaction and impact on performance.
- Study how reliability can benefit from combining PTRF with the space redundancy mechanisms of TSN using probabilistic models. Furthermore, we plan to include permanent faults in our models to see how they impact the behaviour of TSN and PTRF.
- Design, implement and evaluate a complete fault-tolerant network architecture with very high reliability based on TSN.

Bibliography

802.1, ANSI/IEEE (1998). "IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems- Local and Metropolitan Area Networks- Common Specifications Part 3: Media Access Control (MAC) Bridges". In: *ANSI/IEEE Std 802.1D, 1998 Edition*, pp. i–355. DOI: 10.1109/IEEESTD.1998.95619.

802.1, IEEE. *Time-Sensitive Networking (TSN) Task Group.* URL: https://1.ieee802.org/tsn/.

— (2004). "IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges". In: *IEEE Std 802.1D-2004 (Revision of IEEE Std 802.1D-1998)*, pp. 1–281. DOI: 10.1109/IEEESTD.2004.94569.

— (2006). "IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks". In: *IEEE Std 802.1Q-2005 (Incorporates IEEE Std 802.1Q1998, IEEE Std 802.1u-2001, IEEE Std 802.1v-2001, and IEEE Std 802.1s-2002)*, pp. 1–300. DOI: 10.1109/IEEESTD.2006.216285.

— (2007). "IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks Amendment 5: Connectivity Fault Management". In: *IEEE Std 802.1ag - 2007 (Amendment to IEEE Std 802.1Q - 2005 as amended by IEEE Std 802.1ad - 2005 and IEEE Std 802.1ak - 2007)*, pp. 1–260. DOI: 10.1109/IEEESTD.2007.4431836.

— (2009). "IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams". In: *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*, pp. C1–72. DOI: 10.1109/IEEESTD.2009.5375704.

— (2010). "IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks Amendment 14: Stream Reservation

Protocol (SRP)". In: *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)*. DOI: 10.1109/IEEESTD.2010.5594972.

802.1, IEEE (2011a). "IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks". In: *IEEE Std 802.1AS-2011*, pp. 1–292. DOI: 10.1109/IEEESTD.2011.5741898.

— (2011b). "IEEE Standard for Local and Metropolitan Area Networks–Audio Video Bridging (AVB) Systems". In: *IEEE Std 802.1BA-2011*, pp. 1–45. DOI: 10.1109/IEEESTD.2011.6032690.

— (2012). "IEEE Standard for Local and Metropolitan Area Networks–Media Access Control (MAC) Bridges and Virtual Bridges". In: *IEEE Std 802.1Q, 2012 Edition, (Incorporating IEEE Std 802.1Q-2011, IEEE Std 802.1Qbe-2011, IEEE Std 802.1Qbc-2011,IEEE Std 802.1Qbb-2011, IEEE Std 802.1Qaz-2011, IEEE Std 802.1Qbf-2011,IEEE Std 802.1Qbg-2012, IEEE Std 802.1aq-2012, IEEE Std 802.1Q-2012*, pp. 1–1782. DOI: 10.1109/IEEESTD.2012.6606799.

— (2016a). "IEEE Standard for Local and Metropolitan Area Networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic". In: *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q— as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q—/Cor 1-2015)*. DOI: 10.1109/IEEESTD.2016.7572858.

— (2016b). "IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks - Amendment 24: Path Control and Reservation". In: *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. DOI: 10.1109/IEEESTD.2016.7434544.

— (2017a). "IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks–Amendment 28: Per-Stream Filtering and Policing". In: *IEEE Std 802.1Qci-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, and IEEE Std 802.1Qbz-2016)*. DOI: 10.1109/IEEESTD.2017.8064221.

— (2017b). "IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks–Amendment 29: Cyclic Queuing and Forwarding". In: *IEEE 802.1Qch-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd(TM)-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015,*

IEEE Std 802.1Qbu-2016, IEEE Std 802.1Qbz-2016, and IEEE Std 802.1Qci-2017), pp. 1–30. DOI: 10.1109/IEEESTD.2017.7961303.

— (2017c). "IEEE Standard for Local and Metropolitan Area Networks–Frame Replication and Elimination for Reliability". In: *IEEE Std 802.1CB -2017*, pp. 1–102. DOI: 10.1109/IEEESTD.2017.8091139.

— (2018). "IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks – Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements". In: *IEEE Std 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018)*, pp. 1–208. ISSN: null. DOI: 10.1109/IEEESTD.2018.8514112.

— (2020a). "IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks - Amendment 34:Asynchronous Traffic Shaping". In: *IEEE Std 802.1Qcr-2020 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018, IEEE Std 802.1Qcc-2018, IEEE Std 802.1Qcy-2019, and IEEE Std 802.1Qcx-2020)*, pp. 1–151. DOI: 10.1109/IEEESTD.2020.9253013.

— (2020b). "IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications". In: *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, pp. 1–421. DOI: 10.1109/IEEESTD.2020.9121845.

— (2021). "IEEE Standard for Local and Metropolitan Area Networks–Link-local Registration Protocol". In: *IEEE Std 802.1CS-2020*, pp. 1–151. DOI: 10.1109/IEEESTD.2021.9416320.

802.3, IEEE. *Error Rates and Testability.* URL: https://www.ieee802.org/3/efm/public/may03/optics/dawe_optics_1_0503.pdf.

802.3ah, IEEE. *BER Requirements.* URL: https://www.ieee802.org/3/efm/public/nov01/khermosh_3_1101.pdf.

Airbus (2009). "Aircraft Data Network-Part 7, Avionics Full Duplex Switched Ethernet Network (AFDX)". In: *ARINC 664P7-1*.

Álvarez, I., J. Proenza, and M. Barranco (2018). "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)*, pp. 63–64. DOI: 10.1109/DSN-W.2018.00031.

Álvarez, I. et al. (2017). "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)*, pp. 1–4. DOI: 10.1109/ETFA.2017.8247721.

Álvarez, I. et al. (2019). "Fault Tolerance in Highly Reliable Ethernet-Based Industrial Systems". In: *Proceedings of the IEEE* 107.6, pp. 977–1010. ISSN: 1558-2256. DOI: 10.1109/JPROC.2019.2914589.

Álvarez, I. et al. (2019). "Simulation of the Proactive Transmission of Replicated Frames Mechanism over TSN". In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1375–1378. DOI: 10.1109/ETFA.2019.8868997.

Álvarez, I. et al. (2020). "Comparing Admission Control Architectures for Real-Time Ethernet". In: *IEEE Access* 8, pp. 105521–105534. DOI: 10.1109/ACCESS.2020.2999817.

Álvarez, I. et al. (2021). "Design and Experimental Evaluation of the Proactive Transmission of Replicated Frames Mechanism over Time-Sensitive Networking". In: *Sensors* 21.3. ISSN: 1424-8220. DOI: 10.3390/s21030756. URL: https://www.mdpi.com/1424-8220/21/3/756.

Amari, Ahmed et al. (Jan. 2016). "AeroRing: Avionics Full Duplex Ethernet Ring with High Availability and QoS Management". In: *Proceedings of the 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*.

Anderson, T. and P.A. Lee, eds. (1981). *Fault Tolerance – Principles and Practice*. Prentice Hall.

Ashjaei, Mohammad et al. (2014). "Reduced buffering solution for multi-hop HaRTES switched Ethernet networks". In: *2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 1–10. DOI: 10.1109/RTCSA.2014.6910504.

Ashjaei, Mohammad et al. (2017). "Schedulability analysis of Ethernet Audio Video Bridging networks with scheduled traffic support". In: *Real-Time Systems* 53.4, pp. 526–577. ISSN: 1573-1383. DOI: 10.1007/s11241-017-9268-5. URL: https://doi.org/10.1007/s11241-017-9268-5.

Atallah, A. A., G. B. Hamad, and O. A. Mohamed (2019). "Reliability Analysis of TSN Networks Under SEU Induced Soft Error Using Model Checking". In: *2019 IEEE Latin American Test Symposium (LATS)*, pp. 1–6. DOI: 10.1109/LATW.2019.8704633.

Avizienis, A. (1976). "Fault-Tolerant Systems". In: *IEEE Transactions on Computers* C-25.12, pp. 1304–1312. ISSN: 2326-3814. DOI: 10.1109/TC.1976.1674598.

Avižienis, Algirdas et al. (2004). "Basic Concepts and Taxonomy of Dependable and Secure Computing". In: *IEEE Transactions on Dependable and Secure Computing* 1.1, pp. 11–33. ISSN: 1545-5971. DOI: 10.1109/TDSC.2004.2.

Barranco, M., S. Derasevic, and J. Proenza (2020). "An Architecture for Highly Reliable Fault-Tolerant Adaptive Distributed Embedded Systems". In: *Computer* 53.3, pp. 38–46. DOI: 10.1109/MC.2019.2944337.

Bojthe, Zoltan et al. "The INET Framework—An Open-Source OMNeT++ Model Suite for Wired, Wireless and Mobile Networks." In: (). URL: https://inet.omnetpp.org/.

Bordoloi, U. D. et al. (2014). "Schedulability analysis of Ethernet AVB switches". In: *2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 1–10. DOI: 10.1109/RTCSA.2014.6910530.

Brand-Rex. "The Impact of Bit Error Rate on LAN Throughput White Paper". In: *The Impact of Bit Error Rate on LAN Throughput White Paper* (), pp. 1–3.

Burns, Alan and Andy Wellings (2009). *Real-Time Systems and Programming Languages: Ada, Real-Time Java and C/Real-Time POSIX*. 4th. USA: Addison-Wesley Educational Publishers Inc. ISBN: 0321417453, 9780321417459.

Cao, J. et al. (2016). "Tight worst-case response-time analysis for ethernet AVB using eligible intervals". In: *2016 IEEE World Conference on Factory Communication Systems (WFCS)*, pp. 1–8. DOI: 10.1109/WFCS.2016.7496507.

Cao, J. et al. (2018). "An independent yet efficient analysis of bandwidth reservation for credit-based shaping". In: *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, pp. 1–10. DOI: 10.1109/WFCS.2018.8402345.

Cisco. *Understanding Rapid PVST+*. URL: https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus5000/sw/configuration/guide/cli/CLIConfigurationGuide/RPVSpanningTree.html#31778.

Finn, Norman (2018). "Introduction to Time-Sensitive Networking". In: *IEEE Communications Standards Magazine* 2.2, pp. 22–28. DOI: 10.1109/MCOMSTD.2018.1700076.

Fuchs, Christian (Jan. 2012). "The Evolution of Avionics Networks From ARINC 429 to AFDX". In: 65.

Fujiwara, T., T. Kasami, and S. Lin (1989). "Error detecting capabilities of the shortened Hamming codes adopted for error detection in IEEE Standard 802.3". In: *IEEE Transactions on Communications* 37.9, pp. 986–989. ISSN: 1558-0857. DOI: 10.1109/26.35380.

Gessner, D. et al. (2019). "A Fault-Tolerant Ethernet for Hard Real-Time Adaptive Systems". In: *IEEE Transactions on Industrial Informatics* 15.5, pp. 2980–2991. ISSN: 1551-3203. DOI: 10.1109/TII.2019.2895046.

Ha, N. V., T. T. T. Nguyen, and M. Tsuru (2019). "TCP with Network Coding Enhanced in Bi-directional Loss Tolerance". In: *IEEE Communications Letters*, pp. 1–1. ISSN: 2373-7891. DOI: 10.1109/LCOMM.2019.2961096.

Heise, P., F. Geyer, and R. Obermaisser (2016). "TSimNet: An Industrial Time Sensitive Networking Simulation Framework Based on OMNeT++". In: *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5. DOI: 10.1109/NTMS.2016.7792488.

Hofmann, R., B. Nikolić, and R. Ernst (2020). "Challenges and Limitations of IEEE 802.1CB-2017". In: *IEEE Embedded Systems Letters* 12.4, pp. 105–108. DOI: 10.1109/LES.2019.2960744.

IEC (2010a). "Industrial Communication Networks – High Availability Automation Networks – Part 2: Media Redundancy Protocol (MRP)". In: *IEC 62439-2*.

— (2010b). "Industrial communication networks – Profiles – Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3". In: *IEC 61784-2*.

— (2012). "Industrial Communication Networks – High Availability Automation Networks – Part 3: Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR)". In: *IEC 62439-3*.

— (2014). "Digital Data Communications for Measurement and Control – Fieldbus for Use in Industrial Control Systems (All Parts)". In: *IEC 61158/61784*.

— (2016). "Industrial Communication Networks – High Availability Automation Networks – Part 2: Media Redundancy Protocol (MRP)". In: *IEC 62439-2*.

ISO/IEC (1994). "Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model". In: *ISO/IEC 7498-1*.

Johnson, Barry W. (1988). *Design and Analysis of Fault Tolerant Digital Systems*. Addison-Wesley Publishing Company. URL: http://dl.acm.org/citation.cfm?id=SERIES9098.61654.

Kleineberg, O., P. Fröhlich, and D. Heffernan (2011). "Fault-Tolerant Ethernet Networks with Audio and Video Bridging". In: *ETFA2011*, pp. 1–8. DOI: 10.1109/ETFA.2011.6058994.

Kopetz, H. (2008). "The Rationale for Time-Triggered Ethernet". In: *2008 Real-Time Systems Symposium*, pp. 3–11. DOI: 10.1109/RTSS.2008.33.

Kopetz, H. et al. (2005). "The time-triggered Ethernet (TTE) design". In: *Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*, pp. 22–33. DOI: 10.1109/ISORC.2005.56.

Kopetz, Hermann (2011a). *Real-Time Systems*. Real-Time Systems Series. Boston, MA: Springer US. ISBN: 978-1-4419-8236-0. DOI: 10.1007/978-1-4419-8237-7. URL: http://link.springer.com/10.1007/978-1-4419-8237-7.

— (2011b). *Real-Time Systems*. Real-Time Systems Series. Boston, MA: Springer US. ISBN: 978-1-4419-8236-0. DOI: 10.1007/978-1-4419-8237-7. URL: http://link.springer.com/10.1007/978-1-4419-8237-7.

Kwiatkowska, Marta, Gethin Norman, and David Parker (2011). "PRISM 4.0: Verification of Probabilistic Real-Time Systems". In: *Computer Aided Verification*. Ed. by Ganesh Gopalakrishnan and Shaz Qadeer. Springer Berlin Heidelberg, pp. 585–591. ISBN: 978-3-642-22110-1.

Laprie, Jean-Claude (1992). *Dependability: Basic Concepts and Terminology*. Springer-Verlag. ISBN: 9780387822969. URL: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Dependability:+basic+concepts+and+terminology#2.

Mei-Chen Hsueh, T. K. Tsai, and R. K. Iyer (1997). "Fault injection techniques and tools". In: *Computer* 30.4, pp. 75–82. DOI: 10.1109/2.585157.

Morris, Jennifer and Philip Koopman (2005). "Representing design tradeoffs in safety-critical systems". In: *ACM SIGSOFT Software Engineering Notes*. Vol. 30. 4. ACM, pp. 1–5.

Nasrallah, Ahmed et al. (2019). "Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research". In: *IEEE Communications Surveys Tutorials* 21.1, pp. 88–145. DOI: 10.1109/COMST.2018.2869350.

Pahlevan, M. and R. Obermaisser (2018). "Redundancy Management for Safety-Critical Applications with Time Sensitive Networking". In: *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, pp. 1–7. DOI: 10.1109/ATNAC.2018.8615374.

Peti, P. et al. (2005). "A maintenance-oriented fault model for the DECOS integrated diagnostic architecture". In: *19th IEEE International Parallel and Distributed Processing Symposium*, 8 pp.–. DOI: 10.1109/IPDPS.2005.41.

Poledna, Stefan (1996). *Fault-Tolerant Real-Time Systems: The Problem of Replica Determinism*. Norwell, MA, USA: Kluwer Academic Publishers. ISBN: 079239657X.

POWERLINK Basics: System Overview. URL: https://www.ethernet-powerlink.org/uploads/media/POWERLINKBasics_brochure_e.pdf.

Rosset, Valério et al. (2012). "Modeling the reliability of a group membership protocol for dual-scheduled time division multiple access networks". In: *Computer Standards & Interfaces* 34.3, pp. 281 –291. ISSN: 0920-5489.

DOI: https://doi.org/10.1016/j.csi.2011.10.004. URL: http://www.sciencedirect.com/science/article/pii/S0920548911000912.

Salazar, R. et al. (2019). "Utility Applications of Time Sensitive Networking White Paper". In: *Utility Applications of Time Sensitive Networking White Paper*, pp. 1–19.

Samii, S. and H. Zinner (2018). "Level 5 by Layer 2: Time-Sensitive Networking for Autonomous Vehicles". In: *IEEE Communications Standards Magazine* 2.2, pp. 62–68. ISSN: 2471-2825. DOI: 10.1109/MCOMSTD.2018.1700079.

Shu Lin, D. J. Costello, and M. J. Miller (1984). "Automatic-repeat-request error-control schemes". In: *IEEE Communications Magazine* 22.12, pp. 5–17. ISSN: 1558-1896. DOI: 10.1109/MCOM.1984.1091865.

Smirnov, F. et al. (2016). "Formal reliability analysis of switched Ethernet automotive networks under transient transmission errors". In: *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6. DOI: 10.1145/2897937.2898026.

SoC-e. *MTSN Kit: a Comprehensive Multiport TSN Setup.* URL: https://soc-e.com/mtsn-kit-a-comprehensive-multiport-tsn-setup/.

— *System on Chip & FPGA IP Core Development.* URL: https://soc-e.com/.

Varga, András (2001). "The OMNeT++ Discrete Event Simulation System". In: *Proceedings of the European Simulation Multiconference (ESM)*.

Wilamowski, Bogdan M. and J. David Irwin (2011). *Industrial Communication Systems (The Industrial Electronics Handbook)*. Ed. by CRC Press. Second edi. ISBN: 9781439802816.

Wireshark. *Wireshark.* URL: https://www.wireshark.org/.

Wollschlaeger, M., T. Sauter, and J. Jasperneite (2017). "The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0". In: *IEEE Industrial Electronics Magazine* 11.1, pp. 17–27. ISSN: 1932-4529. DOI: 10.1109/MIE.2017.2649104.

Zurawski, Richard, ed. (2015). *Industrial Communication Technology Handbook*. 2nd ed. CRC Press.

Zynq UltraScale+ MPSoC. URL: https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html.

Åkerberg, Johan et al. (2021). "Future Industrial Networks in Process Automation: Goals, Challenges, and Future Directions". In: *Applied Sciences* 11.8. ISSN: 2076-3417. DOI: 10.3390/app11083345. URL: https://www.mdpi.com/2076-3417/11/8/3345.