# Towards a Fault-Tolerant Architecture based on Time Sensitive Networking

Inés Álvarez, Manuel Barranco, Julián Proenza
DMI - Universitat de les Illes Balears, Spain
{ines.alvarez, manuel.barranco, julian.proenza}@uib.es

***Abstract*—The Time Sensitive Networking (TSN) Task Group has been working on describing a set of standards that will provide enhanced capabilities to standard Ethernet. Specifically, they work to provide Ethernet with real-time, reliability and reconfiguration capacities. Nevertheless, this set of standards (commonly referred to as TSN) does not cover some reliability aspects that are relevant for the correct operation of critical distributed control systems. Thus, in this work we present a first proposal of a highly reliable architecture and a set of mechanisms based on TSN to support the real-time and reliability requirements of these critical systems.**

## I. Introduction

The Time Sensitive Networking (TSN) Task Group from the IEEE is developing a set of technical standards. These standards, commonly referred to as TSN, aim at providing Ethernet with real-time, reconfiguration and reliability services in the layer 2 of the network architecture. TSN is devised to provide adequate services to a great variety of systems. Among them, we can find critical distributed control systems. The interaction of these systems with the environment in which they are deployed usually imposes hard real-time and reliability requirements. Moreover, more and more these systems are expected to adapt to unpredictable changes in the environment. To this end, the network subsystem must support reconfiguration without jeopardising the correct system operation.

There has been a growing interest in using Ethernet as the network technology for critical distributed control systems, due to its high bandwidth, low cost and Internet compatibility. Nevertheless, Ethernet does not have adequate support for these systems. For this reason some Ethernet-based solutions, such as Time-Triggered Ethernet (TTE) [1], Flexible Time-Triggered (FTT) [2] or Fault Tolerance for FTT (FT4FTT) [3], have been proposed to provide these systems with real-time and highly reliable communication services. However, these solutions either do not fully fulfill the real-time and reliability requirements of critical systems, or present some limitations regarding their compatibility with Internet. Moreover, although all these solutions are interesting, they are not receiving enough support from the Industry, which in contrast is clearly investing in the development of the TSN standards in general and of TSN standards related to reliability in particular.

Specifically, TSN includes several mechanisms to increase the reliability of the network. First, they described mechanisms to support spatial redundancy. Amendment IEEE Std 802.1Qca Path Control and Reservation [4] enables the creation of multiple paths in the network; whereas standard IEEE Std 802.1CB Frame Replication and Elimination for Reliability [5] allows to replicate streams and deploy them through the paths created using Qca. Second, the standard IEEE 802.1Qci Per-Stream Filtering and Policing [6] enables error containment by discarding frames that are received out of time or that exceed the allocated bandwidth for a given stream.

Unfortunately TSN does not propose a complete highly reliable communication infrastructure and, in fact, it lacks additional mechanisms to build up such an infrastructure. Moreover, to the best of the author's knowledge no one has proposed any mechanism to provide nodes with high reliability on top of TSN. This last limitation is fundamental, as nodes are known to be the most unreliable elements of a system.

Therefore, our goal is to design a highly reliable architecture based on TSN networks, using the services provided by TSN and extending them when needed. We will start this work by focusing on a mono-hop network that could meet the needs of part of a federated architecture. Studying the vulnerabilities and finding solutions for the rest of the network architecture is left as future work.

The remainder of the document is structured as follows. Section II briefly describes TSN features that are relevant for this work. Section III presents the system architecture and the mechanisms we propose and Section IV further discusses some of the most fundamental of those mechanisms. Finally, Section V concludes the document.

## II. TSN overview

In this section we will provide an overview of the TSN mechanisms relevant for this work. On top of the aforementioned reliability standards, TSN includes standards to increase the real-time response of the network and to enable its reconfiguration.

TSN relies on clock synchronization, IEEE 802.1AS-rev [7], and the time-aware shaper, IEEE 802.1Qbv [8], to configure and coordinate the different elements of the network, e.g. the switches, to enforce a given schedule that guarantees the timeliness of the hard real-time traffic. Moreover, the IEEE Std 802.1Qbu Frame Preemption [9] allows high priority frames to interrupt the transmission of certain frames and later resume their transmission, increasing bandwidth efficiency.

The time aware shaper divides the communication in time windows, as shown in Figure 1. Hard real-time traffic is transmitted during the *Protected Window*, isolated from the rest of the traffic. To do so, Qbv defines the *Guard Band*, a

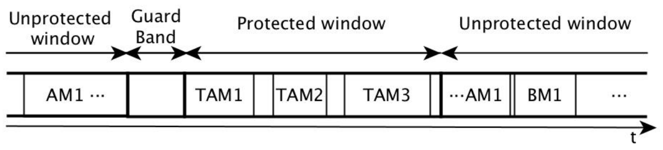

Fig. 1: Division of the communication in windows to separate Scheduled traffic from Class A, B and Best Effort traffic.

period of time during which no frames can be transmitted to prevent them from blocking hard real-time traffic. Finally, soft real-time is transmitted in the *Unprotected Window* and best effort traffic is transmitted at the end of this window if there is enough space. We used the cyclic nature of the communication to simplify the fault tolerance mechanisms we designed, as we will describe in Section IV.

Another key piece of TSN networks is the Stream Reservation Protocol (SRP), originally standardised in IEEE Std 802.1Qat [10] and amended in P802.1Qcc [11]. This protocol provides TSN with real-time and operational flexibility. First, SRP describes streams with different real-time and reliability guarantees (classes of traffic), enabling real-time flexibility. On the other hand, SRP allows for the on-line creation and elimination of the streams, enabling operational flexibility.

In order to allow for this flexibility while meeting the demanding requirements of critical applications, Qcc proposes to use a Centralised Network Configuration (CNC) approach [12]. This approach devises the use of a central element, to which we will refer as CNC element (CNCe). The CNCe has a complete view of the network that allows it to reconfigure the network in runtime. The specification of the CNCe is out of the scope of the standard but it is clear that it will be a key element to guarantee the correct operation of critical networks, specially in dynamic environments.

At the time of writing this work, it is still not clear which will be the configuration capacities of the CNCe. Nevertheless, we can deduce some of them by taking a look into the standards that are being developed to describe YANG data models for the different mechanisms proposed by TSN. P802.1Qcw describes YANG data models for Scheduled Traffic, Frame Preemption and Per-Stream Filtering and Policing, which will allow to modify the traffic in runtime; P802.1ABCU Link Layer Discovery Protocol YANG model for topology discovery and P802.1CBcv Frame Replication and Elimination for Reliability YANG data model [13] for spatial redundancy. Still, the CNCe potential reconfiguration capabilities are not just limited to the models previously described and could be extended to support new mechanisms.

## III. Architecture Proposal

To design our highly reliable architecture on top of TSN, we took advantage of the knowledge acquired by our group in the design of a highly dependable system based on the FTT protocol [14]. Next we describe the proposed architecture and the fault tolerance mechanisms used.

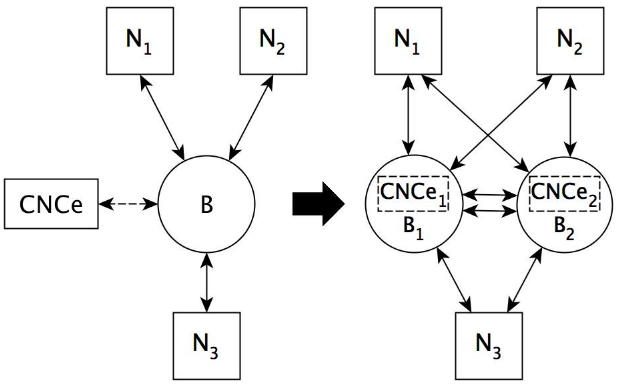

Fig. 2: From a simplex to a replicated star.

### A. Architecture

Figure 2 shows the proposed architecture. As we mentioned, in this work we will focus on a mono-hop architecture that could be a subnetwork of a larger federated architecture. On the left hand of Figure 2 we can see a mono-hop architecture, used as the starting point for our design. As seen, it is conformed by a bridge directly connected to each node through full-duplex links and to a CNCe. The dashed line between the CNCe and the bridge represents a logical connection, as the CNCe may be placed in the bridge or in a different component.

The right hand of the Figure shows our proposed topology. As we can see, it consists in a replicated star topology, where each bridge has a CNCe embedded. We will refer to these bridges as CNCe-bridges. Moreover, we can see that, by replicating the bridges we can easily replicate the links that connect them to the nodes, providing with replicated paths connecting any pair of nodes. Finally, note that we placed two links between the CNCe-bridges. We will refer to these links as *interlinks* and their main function is to establish a direct connection between the CNCes inside the bridges.

Next we will explain our design decisions and we will further describe additional mechanisms needed to guarantee the correct operation of our replicated architecture.

### B. Design Rationale

In order to understand the proposed architecture and the additional mechanisms we need to explain our fault model. The fault model describes the type of faults we want to tolerate with our solution. Specifically, our fault model covers permanent and temporary *non-malicious operational hardware faults* [15]. Moreover, we need to describe how the faults manifest in each component and how we prevent them from affecting the rest of the system. How faults manifest in a component is called the *failure semantic* of the component.

We see that in a star topology the bridge represents a *Single Point of Failure* (SPoF). That is, if the bridge fails the whole system fails as nodes will no longer be able to communicate. Thus, by duplicating the bridge we can tolerate one non-concurrent permanent or temporary fault. Furthermore, the CNCe is a SPoF too. Thus, we duplicated it and we placed one replica in each bridge. This way not only we eliminate the SPoF, but we increase the reliability of the communications between the CNCe and the bridge, as internal communications are less prone to faults than the communication channel. Moreover, we use active replication since we are considering hard real-time systems and we want the fail-over time of the network to be 0.

CNCe-bridges exhibit *arbitrary* failure semantics, which means that a fault can result in an arbitrary behaviour that could interfere with the correct operation of the network. One example of such an interference are *two-faced* behaviours; e.g. a fault in one of the ports could cause the bridge to forward a different message through different ports. Two-faced behaviours can also happen in the time domain, with a port transmitting a message in the correct time while other delays its transmission.

There are several ways to prevent a faulty bridge from interfering with the network operation. Tolerating arbitrary faults requires a higher number of components, exchanging messages and voting on the adequate action; which is costly and time consuming. Since CNCe-bridges are key for the communication and we want to keep the network architecture as simple as possible, we will choose to restrict their failure semantics. Thus, to be able to use a simplified topology, such as the one shown in the Fig. 2, we enforce *crash failure* semantics in the bridges; i.e. bridges produce the correct output or no output at all. This can be achieved with internal duplication and comparison.

Regarding permanent faults in the links that connect the nodes to the bridge, we can take advantage of the replication of the bridge to connect each node to both CNCe-bridges. This way, we can now tolerate the permanent faults of the links. Note that, as we mentioned in Section I, TSN already provides support for spatial redundancy by means of the IEEE Std 802.1Qca and IEEE Std 802.1CB standards. We use these standards to keep full compatibility and allow the extension of our proposal to multi-hop networks.

Moreover, the addition of the interlink allows to increase the number of permanent faults that can be tolerated. Let us consider the architecture depicted in Figure 2. Let us assume that the link that connects $N_3$ to $B_2$ suffers a permanent fault; $N_3$ can still communicate through $B_1$. Let us now assume that the link between $N_2$ and $B_1$ suffers a permanent fault; without interlink $N_3$ and $N_2$ can not communicate any more. Nonetheless, if bridges exchange data traffic through the interlink, $N_3$ and $N_2$ will be able to communicate through the path $N_3$–$B_1$–$B_2$–$N_2$.

Furthermore, as we explained, the interlink represents a direct communication channel between the CNCes. Thus, if the interlink suffered a permanent fault, CNCes would no longer be able to coordinate their actions, which would result in the partition of the network into two networks with uncoordinated CNCes. Uncoordinated CNCes could, in turn, result in inconsistencies among the nodes and the failure of the system. Therefore, we replicate the interlink to eliminate the SPoF it represents.

Regarding temporary faults in the links, we consider them to manifest as omissions. This is so as Ethernet uses *frame check sequence* to detect errors in frames upon reception. We assume that Ethernet's frame check sequence has a perfect error detection and, so, all erroneous frames are discarded.

It is important to note that we do not use the spatial redundancy provided by the architecture to tolerate temporary faults in the links. This is so because temporary faults in the links have a high probability compared to permanent ones. Thus, using spatial replication to tolerate both permanent and temporary faults would result in an inefficient use of spatial redundancy. Thus, we proposed a time redundancy mechanisms to tolerate temporary faults in the links.

The time redundancy mechanism is called Proactive Transmission of Replicated Frames (PTRF). PTRF consists in sending several copies of each critical frame in a preventive manner, to ensure that at least one copy reaches its destination even in the presence of temporary faults. PTRF was first presented in [16] and further details can be found there.

Regarding the nodes, just like CNCe-bridges nodes exhibit arbitrary failure semantics. Thus, we restrict the nodes failure semantics to *incorrect computation* using error-containment. That is, any node fault will manifest as a frame omission or as a frame with incorrect payload. To tolerate errors in the payload we use active replication. Critical nodes will be replicated using the mechanisms presented in [3].

TSN already devises several mechanisms to enforce error containment. As mentioned, the IEEE Std 802.1Qci standard allows to discard frames that are transmitted in an untimely manner and frames that exceed the bandwidth allocated for a given stream. Nevertheless, to the best of the author's knowledge there are no specific mechanisms devised to deal with two-faced behaviours or impersonations. Thus, we will further analyse the TSN standards to identify which mechanisms can support or help in supporting further error-containment mechanisms.

## IV. Replica Determinism

If we want our system to work correctly both CNCe-bridges must act as one, i.e. they must be *replica determinate* [17]. Otherwise, it could result in inconsistencies in both, the nodes and the network. Next we describe how to enforce replica determinism in our network.

Regarding the forwarding of data traffic in the bridges, we need to ensure that both bridges issue the same frames in the same window in each cycle. With the adequate scheduler, TSN already provides this guarantee for scheduled traffic, supported by clock synchronisation and the time-aware shaper.

As we mentioned, CNCes carry out the configuration of the network. The changes could be triggered by a node that requests the modification of part of the traffic using SRP or by a topological change in the network due to faults. To us, the *network status* is composed by the resources available in the network and the requests transmitted by nodes at a given point in time. The network status is used by CNCes to calculate an adequate configuration and to enforce it in the network.

As CNCes exhibit crash failure semantics, we can assume that as long as both CNCes have the same vision of the network status they produce the same configuration. Therefore, we have to ask ourselves, 1) how do we ensure that both CNCes have the same vision of the network status at a given point in time?, 2) how do CNCes decide when to start recalculating the network configuration?, and 3) how do CNCes know when to trigger the changes? Before answering these questions note that we assume that when the system starts, both CNCes are non-faulty and share a common view of the network status.

To answer question 1) we need to take into account that SRP requests are not replicated in time and, thus, can be lost due to temporary faults. Moreover, once a permanent fault affects the link between a node and a CNCe-bridge, said CNCe-bridge will no longer receive the node's requests. Thus, to ensure that both CNCes have the same vision of the network status, even in the presence of faults, CNCes have to exchange the SRP requests they receive. As we mentioned, the information is exchanged through the interlinks.

Now that we stated how CNCes achieve the same vision of the network status we can move to question 2). To decide when to start calculating a new configuration we can take advantage of the existence of a common knowledge of time and the division of the communication into cycles. Let us assume that the network reconfiguration is done every $n$ cycles.

During the first $n$ cycles the CNCes gather information from the network. In the $nth$ cycle, both CNCes exchange the new information of the network status through both interlinks and they compare the information received to their own information. If they have the same information, they start the reconfiguration; otherwise they both choose the state with the highest amount of information and discard the other one.

Now that both CNCes can calculate the new configuration we can move to question 3). To decide when CNCes finish calculating the new configuration we will use the worst computation time. This time, to which we will refer as $w$, is the longest time it can take for a CNCe to calculate the new configuration. Note that this time will depend on the reconfiguration approach, but we assume that $w < n$. Thus, CNCes transmit the configuration to all the components, including the other CNCe, in cycle $n$+$w$. They also send information about the instant of time when the components must actually perform the changes. This process will be repeated every $n$ cycles.

## V. Conclusions

TSN is an IEEE Task Group that is currently working to provide Ethernet with hard real-time, reliability and reconfiguration capacities. To that end, they developed a set of standards commonly referred to as TSN. TSN is attracting the interest of industry, as it represents an appealing technology for the networks of a great variety of systems. In this work we proposed a highly reliable network architecture based on TSN, that could be suitable for critical distributed control systems. We focused on a mono-hop network that could be part of a bigger federated architecture. We proposed to use a replicated star topology where the bridges have enhanced reconfiguration capacities. Moreover, we described mechanisms to support replication and to further increase the reliability of the network.

## Acknowledgements

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

## References

[1] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The Time-Triggered Ethernet (TTE) Design," in *8th IEEE International Symposium on Object-oriented Real-time distributed Computing (Seattle, Washington: TU Wien)*, May 2005, p. 2233.

[2] P. Pedreiras and L. Almeida, "The Flexible Time-Triggered (FTT) paradigm: an approach to QoS management in distributed real-time systems," in *Proc. Int. Parallel and Distributed Processing Symposium*. IEEE Computer Society, 2001.

[3] A. Ballesteros, S. Derasevic, D. Gessner, F. Font, I. Alvarez, M. Barranco, and J. Proenza, "First Implementation and Test of a Node Replication Scheme on top of the Flexible Time-Triggered Replicated Star for Ethernet," in *2016 IEEE World Conference on Factory Communication Systems (WFCS)*, May 2016.

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

[5] "IEEE Standard for Local and metropolitan area networks–Frame Replication and Elimination for Reliability," *IEEE Std 802.1CB-2017*, Oct 2017.

[6] "IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks–Amendment 28: Per-Stream Filtering and Policing," *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)*, Sept 2017.

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

[8] "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic," *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)*, March 2016.

[9] "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks – Amendment 26: Frame Preemption," *IEEE Std 802.1Qbu-2016 (Amendment to IEEE Std 802.1Q-2014)*, Aug 2016.

[10] "IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol (SRP)," *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)*, Sept 2010.

[11] "IEEE Draft Standard for Local and metropolitan area networks–Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks Amendment: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements," *IEEE P802.1Qcc/D2.2, March 2018*, Jan 2018.

[12] W. Steiner, P. G. Peon, M. Gutierrez, A. Mehmed, G. Rodriguez-Navas, E. Lisova, and F. Pozo, "Next generation real-time networks based on IT technologies," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sept 2016, pp. 1–8.

[13] IEEE802—Time-Sensitive Networking (TSN) Task Group. [Online]. Available: https://1.ieee802.org/tsn/

[14] D. Gessner, J. Proenza, and M. Barranco, "A proposal for managing the redundancy provided by the flexible time-triggered replicated star for ethernet," in *2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014)*, May 2014, pp. 1–4.

[15] A. Avižienis, J.-C. Laprie, B. Randell, and L. Carl, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

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

[17] S. Poledna, *Fault-tolerant Real-Time Systems.* The Springer International Series in Engineering and Computer Science, Springer US, 1996.