technologies

Article

Time Sensitive Networking Protocol Implementation for Linux End Equipment

Jesús Lázaro [1,*], Jimena Cabrejas [2], Aitzol Zuloaga [1], Leire Muguira [1] and Jaime Jiménez [1]

[1] Electronics Technology Department, Faculty of Engineering of Bilbao, University of the Basque Country, 48013 Bilbao, Spain; aitzol.zuloaga@ehu.eus (A.Z.); leire.muguira@ehu.eus (L.M.); jaime.jimenez@ehu.eus (J.J.)
[2] Bytek, 48002 Bilbao, Spain; jimena@bytek.info
* Correspondence: jesus.lazaro@ehu.eus

Abstract: By bringing industrial-grade robustness and reliability to Ethernet, Time Sensitive Networking (TSN) offers an IEEE standard communication technology that enables interoperability between standard-conformant industrial devices from any vendor. It also eliminates the need for physical separation of critical and non-critical communication networks, which allows a direct exchange of data between operation centers and companies, a concept at the heart of the Industrial Internet of Things (IIoT). This article describes creating an end-to-end TSN network using specialized PCI Express (PCIe) cards and two final Linux endpoints. For this purpose, the two primary standards of TSN, IEEE 802.1AS (regarding clock synchronization), and IEEE 802.1Qbv (regarding time scheduled traffic) have been implemented in Linux equipment as well as a configuration and monitoring system.

Keywords: TSN; PTP; PCIe; SCADA

Citation: Lázaro, J.; Cabrejas, J.; Zuloaga, A.; Muguira, L.; Jiménez, J. Time Sensitive Networking Protocol Implementation for Linux End Equipment. *Technologies* **2022**, *10*, 55. https://doi.org/10.3390/technologies10030055

Academic Editor: Vijayakumar Varadarajan

Received: 7 March 2022
Accepted: 20 April 2022
Published: 22 April 2022

1. Introduction

Since Ethernet was standardized in 1983 [1], it has become the de facto link protocol used in industrial fieldbuses and the aerospace, automotive, or transportation sectors. Most applications in these critical sectors need to operate in real-time: data must be received within tight deadlines. However, Ethernet does not work deterministically; i.e., it cannot guarantee network latency times. Therefore, over the last 35 years, various conventional Ethernet-based enhancements have been developed. Examples of these protocols are, for instance, Profinet [2], EtherCAT (industry) [3], AFDX (aerospace) [4], or Time-Triggered (TT) Ethernet (automotive) [5]. They are incompatible with each other and with conventional Ethernet because they incorporate different mechanisms to guarantee determinism. This heterogeneity means that the industry's market for real-time Ethernet solutions is currently highly fragmented.

In addition, the advent of Industry 4.0 and the digitalization process aims to obtain greater productivity and effectiveness by proposing the interconnection of Operational Technology (OT) with data networks or Information Technology (IT) (see Figure 1). This concept is called the IIoT and consists of the interconnection of all the elements of the factory (sensors, machinery, industrial computer) with external data centers. This interconnection allows it to collect many operational data from the factory to analyze them using services. Subsequently, a high level of automation is achieved to optimize factory control processes and improve productivity and efficiency. The interoperability of both sides is a crucial aspiration of IIoT.

Due to specific characteristics of each of the networks, the technologies in both worlds (OT and IT) have been very different and generally not interoperable. For this reason, with current technologies, this interconnection is not trivial since factory networks cannot directly access the Internet, as they are not compatible with conventional Ethernet. Therefore,

a standard Ethernet-based technology that can bridge the IT and OT worlds is needed. TSN is a proposal for such an Ethernet-only-based solution.

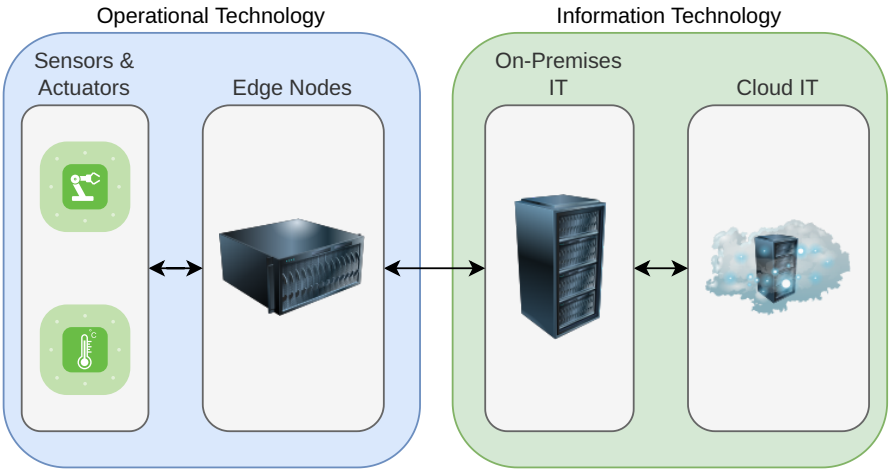

Figure 1. The boundary between OT world (factories) and IT (data centers).

This proposal is not without its own challenges which range from practical ones to technical ones. The initial high cost is one of the former, mainly because all switches and elements of the network must be TSN-aware. Another one is the rapidly changing unfinished standards that sometimes overlap. There is no current ability to verify conformance and interoperability. Cybersecurity and other security issues are also a challenge, mainly because TSN is strongly dependent on robust synchronization and configuration that can be targeted [6]. TSN configuration is, as well, another issue due to the lack of a standard data model.

In this direction, this paper describes the implementation of two of the primary standards of TSN, IEEE 802.1AS (regarding clock synchronization) and IEEE 802.1Qbv (regarding time Scheduled traffic (ST)), establishing an end-to-end TSN network. The main contributions can be summarized as: (a) Setup and configuration of the Linux kernel to extend the TSN slot configuration up to the Operating System (OS). (b) Setup of the dedicated hardware within Linux. (c) Implementation of a dedicated software tool to configure easily all different parameters. (d) A visualization tool that shows packets in their slots to check its correct behaviour.

The rest of the paper is organized as follows: Section 2 analyzes the TSN-related standards, Section 3 gives an overview of the related work, Section 4 describes the proposed solution, Section 5 shows the set-up results, and Section 6 summarizes the main conclusions of the work.

2. Related Standards

TSN is a set of standards developed by the IEEE Time-Sensitive Networking Task Group [7]. This task group was formed in 2012 from the existing Audio/Video Bridging (AVB) Task Group. AVB is a standard developed for synchronized audio and video data transmission over LANs. The idea of the TSN group is to migrate and adapt the technical solutions provided by the AVB initiative to other sectors and for sending all types of data. It should be noted that TSN is not a communication protocol per se but an evolution of Ethernet. All the TSN-related standards are part of the IEEE Ethernet standard. For example, advanced TSN features such as preemption are part of the 802.3.

With the advent of TSN, deterministic data transmissions can be achieved with conventional Ethernet. TSN enforces bandwidth and time slots, thus increasing the isolation. It allows critical data to be sent over the same communication link as the rest of the traffic without causing delays or disturbances, thus eliminating the need to create industrial networks independent of each other. These traffic classes facilitate data exchange between

production sites and enterprises by fully interoperating factory networks with the Internet [8]. All the switches must be TSN-aware to take advantage of TSN. The main reason behind this requirement is the needed advanced synchronization. On the other hand, cabling and Ethernet cards of non-real-time nodes will be unchanged. Since it is an open standard, different vendors can achieve interoperability without the problems of proprietary protocols. This technology can be used in almost all industrial applications thanks to its flexibility in meeting different latency, jitter, or error tolerance requirements.

As mentioned before, TSN is not a single standard but a set of standards to make Ethernet more deterministic. Every standard develops at different rates depending on the evolution of the market and its needs. Some of these standards are already thoroughly tested and implemented, and others are still in the early stages of development (draft versions) [9].

Two of the base standards are:

- IEEE 802.1ASrev: This standard defines the IEEE 802.1AS protocol, used for clock synchronization. Through this many, advanced features of TSN can be achieved.
- IEEE 802.1Qbv: This standard defines the IEEE 802.1Qbv protocol, used for ST. It leverages network synchronization to divide the bandwidth and time slots.

2.1. IEEE 802.1Qbv (Enhancements for ST)

With IEEE 802.1Qbv, packet transmission is scheduled end-to-end in a repeating cycle. Qbv allows packets' deterministic arrival, giving latency guarantees, extremely low jitter, and no packet loss. Three basic types of traffic are defined in TSN: ST, Best-Effort traffic (BE), and RE. ST is appropriate for critical messages with strict real-time requirements. BE is general Ethernet traffic that does not require any QoS. Moreover, Reserved Traffic (RT) is for frames that need to reserve specific bandwidth and have soft real-time requirements [10].

A Time-Aware Shaper (TAS), defined in IEEE 802.1Qbv [11], is a gate that enables or disables the frame transmission depending on the scheduling algorithm. TAS divides Ethernet communications into fixed-length, continuously repeating cycles. These cycles are divided into time slots, and in each time slot, one or more of the eight priorities are assigned.

The number of time-slots in each cycle, their duration, and which priorities can be transmitted in each one are fully configurable by the application. Thanks to this operation, ST can have dedicated time slots, which ensures the deterministic operation of this traffic over a conventional Ethernet network. On the other hand, the reserved and BE is accommodated in the remaining time slots in each cycle. RT is guaranteed a dedicated bandwidth, while BE can use the remaining bandwidth. An example 802.1Qbv configuration is shown in Figure 2.

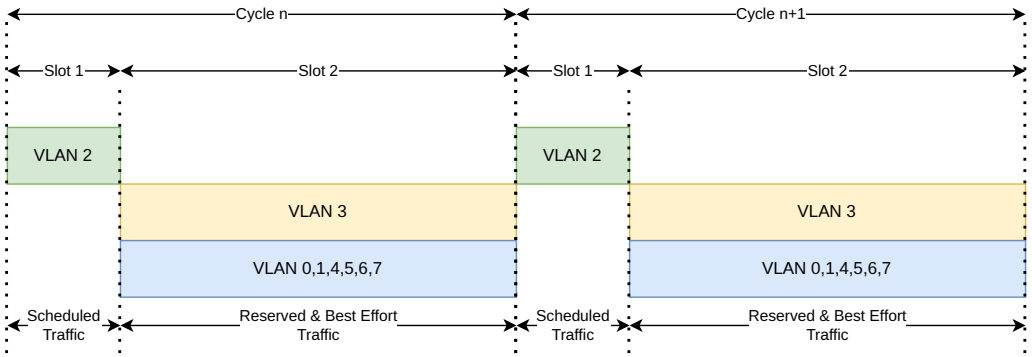

Figure 2. Time-slots division. Slot 1 is reserved for ST; no other traffic is present during the slot. Slot 2, on the other hand, is used by Reserved and Best Effort Traffic. The main difference is that RT is guaranteed minimum bandwidth.

Since the operation of TSN is based on sending different types of traffic at different time intervals, all network equipment must be synchronized in the nanosecond range.

2.2. IEEE 802.1ASrev (Timing & Synchronization)

It was the first standard to be published, making it the most widely implemented TSN standard today [12]. With IEEE 802.1ASrev, network end devices and switches have a common clock, allowing synchronization with an accuracy of less than 1 µs. Synchronization is possible over long distances without affecting the packet propagation delay.

The Precision Time Protocol (PTP) defined in IEEE 1588 [13] is used to synchronize the devices' clocks in a network with microsecond precision. The Generic Precision Time Protocol (gPTP), also known as IEEE 802.1ASrev—the successor of 802.1AS—, is a profile of PTP that includes features that significantly improve the accuracy of clock synchronization. gPTP has some changes that make the two protocols not compatible.

The synchronization and establishment of the clock domain in gPTP occurs in four phases: (a) Determine whether the other equipment on the link (peer) is capable of supporting gPTP. (b) Determine the link latency and the clock frequency of the peer. (c) Select the best clock in the network as the master (MasterClock). (d) Synchronize all nodes in the network to the MasterClock.

3. Related Work—TSN Implementation

TSN implementation has been extensively proposed and analyzed in different sectors and applications. Deterministic latency is accomplished through time synchronization and the application of a global schedule, corresponding to IEEE 802.1AS and 802.1Qbv TSN standards, respectively [14]. So, principally, critical traffic can be scheduled more deterministically using these two standards [15]. Experimental procedure in [16] demonstrates that if requirements of latency and jitter are very low, IEEE 802.1Qbv scheduling must be combined with clock synchronization mechanisms. Including IEEE 802.1ASrev clock synchronization with IEEE 802.1Qbv standard is a challenging case study [17]. For example, in networks containing many flows, it can be complex to decide how to schedule them [18]. This section reviews the implementation of these two TSN standards, comparing the research literature to the proposed solution.

In [14], some IEEE 802.1AS design decisions motivated by specific application requirements are explained. A survey of the synchronized time utilization by software and other applications running at network edges is performed. They emphasized the need to continue improving the PTP and related standards and precision time software support and OS for end stations running the synchronized time applications.

Regarding scheduled transmission defined in IEEE 802.1Qbv, standard alternative queuing algorithms defined by IEEE 802.1 are discussed in [19]. Besides, routing impact over TT traffic schedulability is also explored in [20]. They discussed the need for specialized routing algorithms for such traffic and proposed two Integer Lineal Programming (ILP)-based routing algorithms for improving TT traffic schedulability. Ref. [15] gives an improvement providing a combination of the TT and the priority-based scheduling, broadening the solution for the Gate Control Listss (GCLs). On the other hand, Ref. [21] proposes using a GCL synthesis approach based on a Greedy Randomized Adaptive Search Procedure to schedule TT flow.

The simulation framework has crucial importance, allowing extracting system performance at the development phase stages [15,22,23]. In order to evaluate the TSN-based system's global time base reliability, Ref. [22] presents a simulation framework for IEEE 802.1ASrev. In contrast to existing clock synchronization simulation frameworks, it considers faults and dynamic changes in real-time systems. For TSN network simulation and IEEE 802.1Qbv scheduling mechanism analysis, Ref. [23] presents TSN-specific extensions to OMNeT++/INET framework. Conversely, Ref. [15] presents a method based on network calculus for evaluating TSN critical communications where the GCLs have been previously generated.

Another significant challenge is the integration of initially non-TSN-capable devices in TSN networks [24]. For example, most of the analyzed implementations are based on Linux OS, such as [14,24]. Linux also has uncertainty sources for real-time performance,

such as preemption or interruptions [25], but it includes many mechanisms to achieve better reliability. A clear timeline in [26] shows the evolution of Linux regarding real-time performance.

4. Proposed Solution

As seen in previous sections, synchronization poses several challenges. These challenges are increased when we seek to use slots and shape the traffic in the wire. Our solution is based on specialized TSN capable PCIe network cards — RELY-TSN-PCIe. These cards are TSN capable and based on an Field Programmable Gate Array (FPGA) and an Intel i210 chip. The Intel i210 is TSN capable, while the FPGA provides multiple paths —embedded Ethernet switch— and real-time capabilities. This combination allows using standard drivers in the PC, which is crucial when the OS lacks TSN support. At the same time, it also allows advanced scheduling not supported by the Intel chip. The RELY-TSN-PCIe card is the first known solution for TSN that allows the deployment of a deterministic Ethernet network abstracting from the user's end equipment and the application for which it will be used. In other words, it can be used in different end equipment (Supervisory Control And Data Acquisition (SCADA), IoT gateway) and thus introduces TSN technology in the equipment and integrate it into the deterministic network [27]. The insertion is transparent from the OS perspective since it only detects a standard Ethernet Card. If the OS wants to use advanced scheduling, this introduction becomes less transparent, but the TSN network operation does not require such changes. The solution has been developed in an Ubuntu 20.04 LTS.

4.1. IEEE 802.1ASrev Implementation

For implementing the 802.1ASrev Timing and Synchronization for Time-Sensitive Applications standard, the network clocks that have to participate in the standard were first identified. Figure 3 shows the identified clocks and synchronization links. As can be seen, six different clocks can be distinguished.

It is necessary to distinguish between the way to synchronize all the clocks. There are two types of synchronization: (a) network synchronization (b) device-network synchronization The network synchronization is based on synchronizing the four PTP hardware clocks (PHC) of the I210 and the PCIe; for this, the ptp4l command included in the linuxptp [28] package will be used. linuxptp is an implementation of the PTP for Linux. ptp4l implements Boundary Clock (BC) and Ordinary Clock (OC). On the other hand, the system clock, which is software, gets its time from the Internet using NTP or GPS for the device-network synchronization. Nevertheless, in this case, the system clock will get its time from the TSN network through the phc2sys command included in the `linuxptp` package instead of directly using gPTP, which uses hardware timestamping.

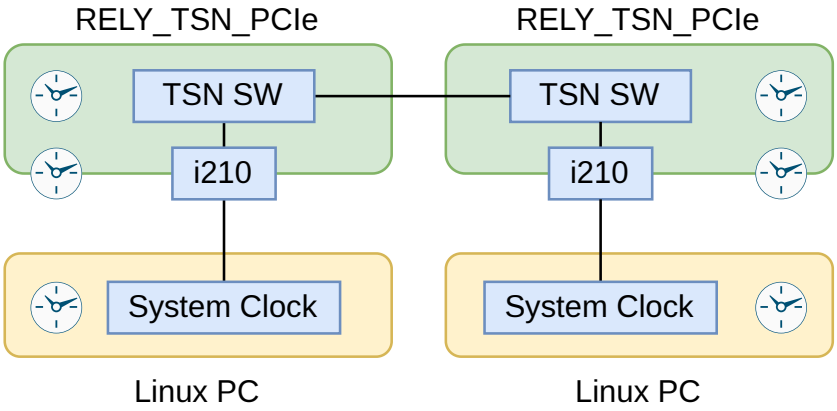

Figure 3. Network clocks that require to be synchronized.

4.2. IEEE 802.1Qbv Implementation

There are two options to build a network in which Qbv is implemented end-to-end. The first is to activate Qbv only on the output of the Linux endpoint that acts as a talker. In this way, the packets will leave the Linux Kernel orderly fashion and theoretically propagate unordered throughout the network until they reach the listener. However, this is not the best approach. From the kernel output to the wire, several layers inject jitter. This jitter may lead to non-compliance with the allocated slots.

The second option is to activate Qbv in two points: in the Kernel of the Linux talker and in the output port of the Ethernet card connected to the talker, as shown in Figure 4. This second approach is the one that has been used. The same Qbv is configured at both points. In this way, the packets follow the allocated slots at the wire. When all the network elements are also TSN-aware, these slots will be maintained up to the receiver.

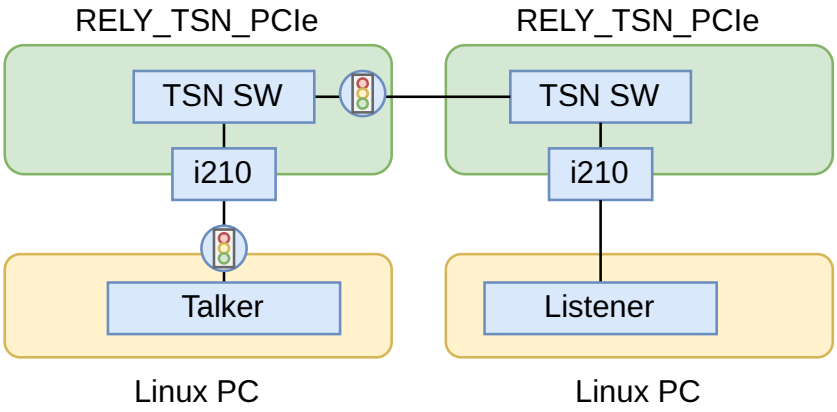

Figure 4. Qbv net structure.

Kernel patches have been created to provide Qbv functionality to Linux machines. In the following, we will explain how packet forwarding works on Linux machines, the two patches that have been created to work with time-slots, and their implementation on the final Linux machines.

The traffic forwarding on Linux systems is done through the Traffic Control (TC) subsystem of the Kernel [29]. TC subsystem code operates between the Intellectual Property (IP) and network interface drivers that transmit data to the network. This subsystem is responsible for constantly supplying packets to be sent to the driver.

TC is composed of queue disciplines (qdisc). The qdiscs represent the scheduling policies applied to the queue. It reorganizes the packets arriving in the queue according to the rules installed in that scheduler and sends them in that new order. By default, this scheduler maintains a First In, First Out (FIFO) queue. Therefore, what is needed is a qdisc that can reorganize packets based on time intervals and send traffic in an orderly fashion based on the 802.1Qbv standard.

The vanilla Linux kernel does not have this queuing discipline, so kernel patches were developed that introduce the necessary tools to implement 802.1Qbv on Linux systems [30,31]. These patches introduce two new qdiscs:

- Earliest TxTime First Qdisc (ETF) allows applications to control the exact instant a packet should be sent to the network card driver. ETF achieves that by buffering packets until a configurable time before their transmission time.
- Time-Aware Priority Shaper (TAPRIO) implements a simplified version of the state machines defined by the IEEE 802.1Qbv standard (see IEEE 802.1Qbv Standard) that allows the configuration of a sequence of gateway states where each state allows or disallows the egress of traffic for a subset of traffic classes.

The patches also introduce a new option to system sockets called SO_TXTIME to enable a socket for time-based transmission and thus configure its parameters.

5. Results

A high-level application has been created to configure both standards graphically. It enables the network administrators to configure and observe the network providing slot information similar to Wireshark's I/0 Graph shown in Figure 5 but in real-time. Proofs of concepts have been developed in a specially set up network composed of two PCs (talker and listener) connected through a TSN-aware network composed of a single switch.

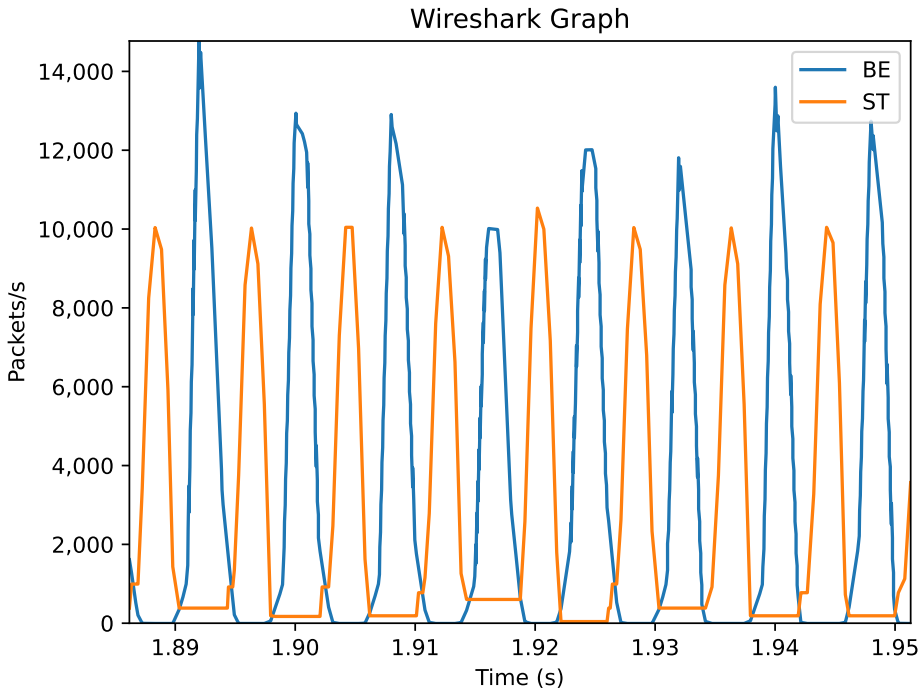

Figure 5. Wireshark capture demonstrating slots.

The first step is to check the synchronization among all the different systems. Checking the Pulse-Per-Second output of the PCIe boards and the information provided by the PTP daemons, the Ethernet cards are synchronized within 10 ns, while the systems are in the 100 ns range (see Figure 6).

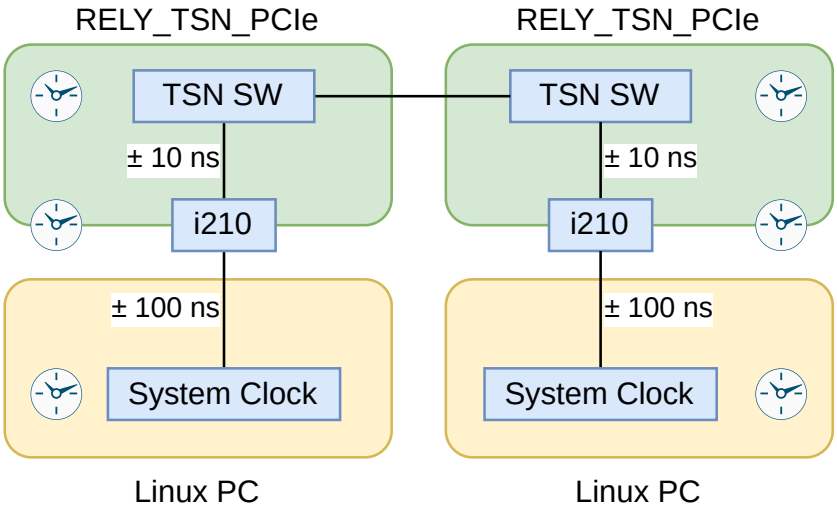

Figure 6. Fully synchronized system. gPTP relationships have been created among the internal switches of the cards, between the internal switch and the i210 chip, and between the chip and the PC.

A demo has been created to test the operation of the patches on the endpoint. The demo configures the Qbv patches in the talker to send ST and Best Effort traffic. The listener receives this data and shows graphically in real-time how each one arrives in its slot.

The resulting traffic can be seen in Figure 7, as shown by the real-time window of the developed application. These demos show the correct operation of the standards and their integration with the PCIe card. The result is ordered and shaped traffic.

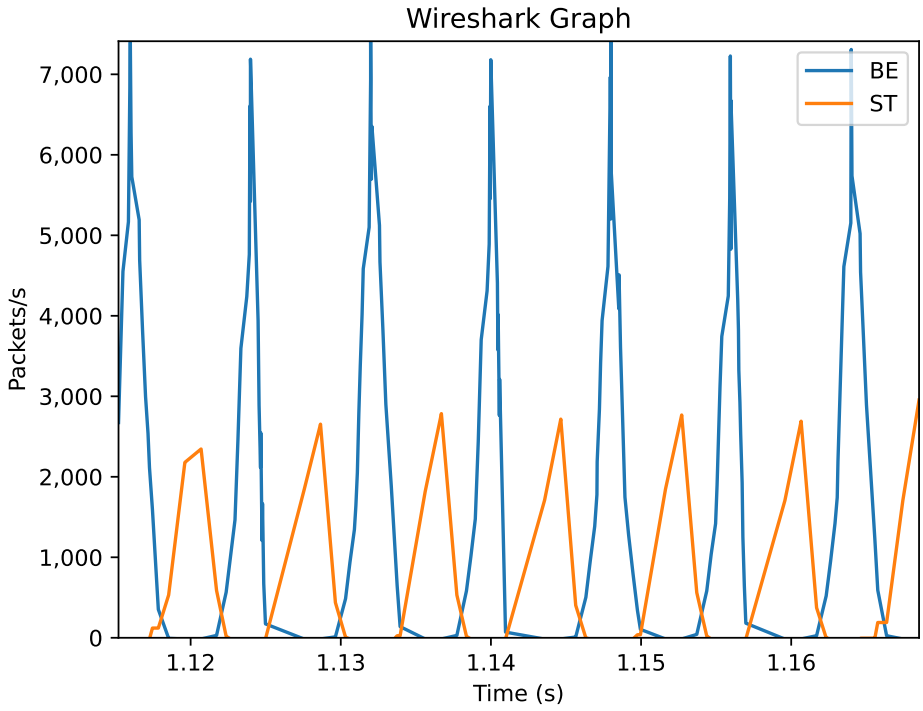

Figure 7. Real-time capture by the created application demonstrating slots. This real-time view allows the network designer to follow the operation of the network and tune the different configurations to the desired requirements.

The results are similar to those provided by Wireshark but, in this case, obtained in real-time. The application allows easy management of the link. At the same time, they serve to prove the correct operation of the TSN-enabled system.

Using this configuration framework, the designer can create a network with different configurations and see their results. A more sophisticated real-life example can be seen in Figure 8.

In this example, each traffic type in TSN is based on the Priority Code Point (PCP) bits of the Virtual Local Area Network (VLAN) tag. In this set-up, the traffic has been classified as follows:

- ST: Brake Info (Data Distribution Service (DDS) Stream 1, VLAN 11, PCP 2).
- RT: Camera Real Time Video (DDS Stream 2, VLAN 12, PCP 5).
- BE: Remaining TCP/IP traffic. (VLAN 3, PCP 6).

TSN configuration is distributed into four slots that complete a Cycle-time of 10 ms. The distribution of the traffic in each slot is as follows: (1) Free. (2) ST. (3) Free. (4) RT+BE. As can be seen, the different streams are constrained in the configured slots.

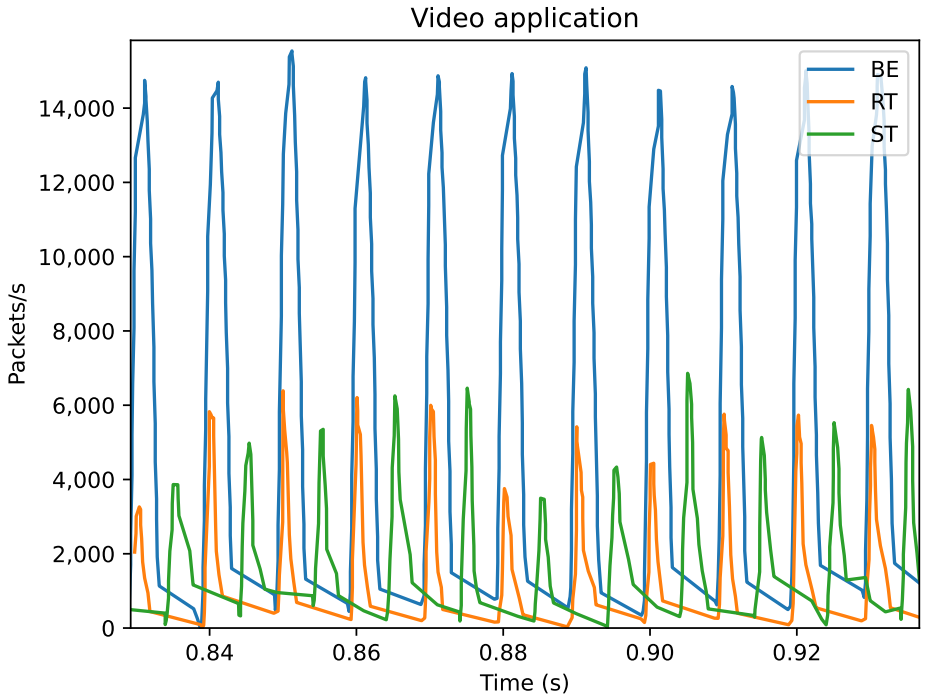

Figure 8. A real-life example of an in-car network. ST is composed of brake info. RT provides the real-time video while BE is composed of the rest infoentertainment data.

6. Conclusions

The main result of the work described in this paper is the construction of a TSN-capable system that can be used to provide a reliable and scalable network. Therefore, it has been possible to implement the two primary TSN standards in Linux end devices and validate the RELY_TSN_PCIe cards' correct operation. The end devices have been included in the TSN network. On the one hand, all the network clocks have been synchronized by implementing the IEEE 802.1ASrev standard, using an open-source daemon in the end devices. On the other hand, we have implemented the ordered sending of packets in time slots following the IEEE 802.1Qbv standard using public Linux Kernel patches. Furthermore, we have created a configuration and visualization tool that helps the network designer to setup and understand the operation of the system. Thanks to this work and the two open technologies used, progress is being made in implementing TSN in standard equipment, i.e., non-proprietary equipment.

Author Contributions: Conceptualization, J.L. and J.C.; methodology, J.C.; software, J.C.; validation, L.M.; formal analysis, A.Z.; investigation, J.L.; resources, J.L.; writing—original draft preparation, J.L. and J.C.; writing—review and editing, J.L.; visualization, J.L.; supervision, J.J.; project administration, J.J.; funding acquisition, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by the Ministerio de Economía y Competitividad of Spain within the project TEC2017-84011-R and FEDER funds as well as by the Department of Education of the Basque Government within the fund for research groups of the Basque university system IT978-16.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. IEEE. IEEE Specification for 10 Mbps Ethernet. 1983. Available online: https://standards.ieee.org/ieee/802.3/7071/ (accessed on 19 April 2022).

2. Kleines, H.; Detert, S.; Drochner, M.; Suxdorf, F. Performance Aspects of PROFINET IO. *IEEE Trans. Nucl. Sci.* **2008**, *55*, 290–294. [CrossRef]
3. Cena, G.; Bertolotti, I.C.; Scanzio, S.; Valenzano, A.; Zunino, C. Evaluation of EtherCAT Distributed Clock Performance. *IEEE Trans. Ind. Inf.* **2012**, *8*, 20–29. [CrossRef]
4. He, F.; Zhao, L.; Li, E. Impact Analysis of Flow Shaping in Ethernet-AVB/TSN and AFDX from Network Calculus and Simulation Perspective. *Sensors* **2017**, *17*, 1181. [CrossRef] [PubMed]
5. Yang, X.; Huang, Y.; Shi, J.; Cao, Z. A Performance Analysis Framework of Time-Triggered Ethernet Using Real-Time Calculus. *Electronics* **2020**, *9*, 1090. [CrossRef]
6. Jin, X.; Xia, C.; Guan, N.; Zeng, P. Joint Algorithm of Message Fragmentation and No-Wait Scheduling for Time-Sensitive Networks. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 478–490. [CrossRef]
7. Time-Sensitive Networking Task Group. IEEE 802.1 Standards. 2018. Available online: http://www.ieee802.org/1/pages/tsn.html (accessed on 19 April 2022).
8. Hallmans, D.; Ashjaei, M.; Nolte, T. Analysis of the TSN Standards for Utilization in Long-life Industrial Distributed Control Systems. In Proceedings of the 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 8–11 September 2020. [CrossRef]
9. Cisco. Time-Sensitive Networking: A Technical Introduction. 2017. Available online: https://www.cisco.com/c/dam/en/us/solutions/collateral/industry-solutions/white-paper-c11-738950.pdf (accessed on 19 April 2022).
10. Gavriluț, V.; Pop, P. Traffic-type Assignment for TSN-based Mixed-criticality Cyber-physical Systems. *ACM Trans. Cyber-Phys. Syst.* **2020**, *4*, 1–27. [CrossRef]
11. Time-Sensitive Networking Task Group. 802.1Qbv—Enhancements for Scheduled Traffic. Available online: https://www.ieee802.org/1/pages/802.1bv.html (accessed on 19 April 2022).
12. Kim, Y.J.; Kim, J.H.; Cheon, B.M.; Lee, Y.S.; Jeon, J.W. Performance of IEEE 802.1AS for automotive system using hardware timestamp. In Proceedings of the The 18th IEEE International Symposium on Consumer Electronics (ISCE 2014), Jeju, Korea, 22–25 June 2014. [CrossRef]
13. Time-Sensitive Networking Task Group. 802.1AS—Timing and Synchronization. Available online: https://www.ieee802.org/1/pages/802.1as.html (accessed on 19 April 2022).
14. Stanton, K.B. Distributing Deterministic, Accurate Time for Tightly Coordinated Network and Software Applications: IEEE 802.1AS, the TSN profile of PTP. *IEEE Commun. Stand. Mag.* **2018**, *2*, 34–40. [CrossRef]
15. Zhao, L.; Pop, P.; Craciunas, S.S. Worst-Case Latency Analysis for IEEE 802.1Qbv Time Sensitive Networks Using Network Calculus. *IEEE Access* **2018**, *6*, 41803–41815. [CrossRef]
16. Farzaneh, M.H.; Knoll, A. Time-sensitive networking (TSN): An experimental setup. In Proceedings of the 2017 IEEE Vehicular Networking Conference (VNC), Turin, Italy, 27–29 November 2017. [CrossRef]
17. Vlk, M.; Brejchová, K.; Hanzálek, Z.; Tang, S. Large-scale periodic scheduling in time-sensitive networks. *Comput. Oper. Res.* **2022**, *137*, 105512. [CrossRef]
18. Kumar, G.N.; Katsalis, K.; Papadimitriou, P.; Pop, P.; Carle, G. Failure Handling for Time-Sensitive Networks using SDN and Source Routing. In Proceedings of the 2021 IEEE 7th International Conference on Network Softwarization (NetSoft), Tokyo, Japan, 28 June–2 July 2021; pp. 226–234. [CrossRef]
19. Finn, N. Introduction to Time-Sensitive Networking. *IEEE Commun. Stand. Mag.* **2018**, *2*, 22–28. [CrossRef]
20. Nayak, N.G.; Dürr, F.; Rothermel, K. Routing algorithms for IEEE802.1Qbv networks. *ACM SIGBED Rev.* **2018**, *15*, 13–18. [CrossRef]
21. Gavriluț, V.; Pop, P. Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications. In Proceedings of the 2018 14th IEEE International Workshop on Factory Communication Systems (WFCS), Imperia, Italy, 13–15 June 2018. [CrossRef]
22. Pahlevan, M.; Balakrishna, B.; Obermaisser, R. Simulation Framework for Clock Synchronization in Time Sensitive Networking. In Proceedings of the 2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC), Valencia, Spain, 7–9 May 2019; pp. 213–220. [CrossRef]
23. Falk, J.; Hellmanns, D.; Carabelli, B.; Nayak, N.; Durr, F.; Kehrer, S.; Rothermel, K. NeSTiNg: Simulating IEEE Time-sensitive Networking (TSN) in OMNeT++. In Proceedings of the 2019 International Conference on Networked Systems (NetSys), Munchen, Germany, 18–19 March 2019. [CrossRef]
24. Cochran, R.; Marinescu, C. Design and implementation of a PTP clock infrastructure for the Linux kernel. In Proceedings of the 2010 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, Portsmouth, NH, USA, 27 September–1 October 2010; pp. 116–121. [CrossRef]
25. de Oliveira, D.B.; Casini, D.; de Oliveira, R.S.; Cucinotta, T. Demystifying the Real-Time Linux Scheduling Latency. In Proceedings of the 32nd Euromicro Conference on Real-Time Systems (ECRTS 2020), Virtual, 30 June 2020. [CrossRef]
26. Reghenzani, F.; Massari, G.; Fornaciari, W. The Real-Time Linux Kernel. *ACM Comput. Surv.* **2020**, *52*, 1–36. [CrossRef]
27. SoC-e. RELY-PCIe Time-Aware Redbox-DAN-Switch PCIe Platform. 2021. Available online: https://www.relyum.com/web/rely-pcie/ (accessed on 19 April 2022).
28. Cochran, R. The Linux PTP Project. Available online: http://linuxptp.sourceforge.net/ (accessed on 19 April 2022).
29. Vehent, J. QOS & contrôle du trafic. *Gnu/Linux Magazine*, May 2010; Volume 127.

30. Gomes, V.C. *Net/Sched: Introduce the Taprio Scheduler*; Technical Report; Intel Inc.: Santa Clara, CA, USA, 2018. Available online: https://lwn.net/ml/netdev/20180929005943.12928-1-vinicius.gomes@intel.com/ (accessed on 19 April 2022).
31. Intel. Adopting Time-Sensitive Networking (TSN) for Automation Systems. 2020. Available online: https://software.intel.com/content/www/us/en/develop/articles/adopting-time-sensitive-networking-tsn-for-automation-systems-0.html (accessed on 19 April 2022).