embeddedworld2023
Exhibition&Conference
... it's a smarter world

Quo Vadis Real Time Ethernet

Prosper Leibundgut, Hans Dermot Doran
Institute of Embedded Systems (InES)
Zurich University of Applied Sciences
Winterthur, Switzerland
email: {leiu, donn}@zhaw.ch

***Abstract*—Real time Ethernet (RTE) protocol suites are commonly operated within an exclusively allocated Ethernet based network that is used to exchange data for a distributed real time application. In practice, RTE protocol stack implementations interlace the maintenance of their data objects on the (standardised and loosely coupled) application layer with the task of traffic fitting. The latter includes the egress and ingress of application data over the underlying layers but also the coordination (scheduling) of the same. The set of time sensitive networking (TSN) IEEE standards is an addendum to common Ethernet (IEEE 802.3*). It has the aim to provide technologies to implement deterministic Ethernet networks. In factory automation RTE, an ongoing establishment of such technologies is observed. They lay the ground for various possibilities to shift the mechanisms for scheduling data transmissions towards networking juncture elements, e.g. Ethernet switch. This work intends to fabricate a stronger separation between the application layer and the tasks concerning traffic fitting. A demonstration setup is developed. It consists of an Ethernet switch (partly TSN capable), two programmable logic controllers (PLCs) and one input/output (I/O) device. Simultaneous operation of two unsimilar RTE protocol suites within the same network is shown. Possible optimisations applied to RTE application components, which target a higher level of determinism, are presented. Measurements underpin the chosen optimisations.**

***Index Terms*—real time Ethernet, time sensitive networking, PROFINET, CC-Link, multi protocol stack operation**

I. Introduction

For the realisation of industrial real time Ethernet (RTE) applications, a wide selection of protocol suites is available. In most of the cases, RTE protocol suite standards are maintained by large based associations but often with a minor number of main influencing companies. Market players whose core business is the fabrication of sensor/actuator hardware, e.g. to be used as part of industrial machinery, have an indisputable interest in providing broad RTE protocol suite compatibility so that their products can be integrated into various, often pre existent, process control systems. Such device manufacturers often rely on third-party vendors which provide the desired bundle, in the form of a composition (platform) of components such as RTE enabled network hardware and software in order to hold RTE connectivity available. In the case of an already present hardware platform, it is often decided to purchase a RTE protocol stack, [1]–[3], which can be adopted rather than developing a complete own implementation. Hardware based RTE interfacing products utilise field programmable gate array (FPGA) technology, where a subset of functionality can be hardware offloaded through a hardware description language (HDL) to circumvent processing overhead, caused through e.g. a (constrained) central processing unit (CPU) and, if present, operating system (OS) scheduling routines, [4]–[6]. Some interface providers even take a step further and manufacture their own application-specific integrated circuits (ASICs), [7]–[9]. Albeit the aforementioned products offer rich configuration possibilities for various RTE protocol suites, the configuration sets are pinned to one particular RTE protocol suite and have to be installed on the platform before or during the device commissioning phase. The time sensitive networking (TSN) addenda to standard Ethernet originate from the audio/video domain and have been gradually extending to additionally cover deterministic timing requirements, targeted for the industrial automation domain. These addenda are provided in the form of IEEE standards which foremost address mixed critical data traffic organisation (shaping) on the Ethernet media access control (MAC) layer. With the appearance of the IEEE consortium on the scene, a global and cross-domain operating standardisation body is raising expectations for less proprietary implementations.

Within the scope of this present work, several RTE protocol suites are examined, commonalities are identified and a separation of unified RTE application scheduling domains is presented. In a next phase, a demonstrator application is evolved with the aim to run multiple RTE protocol suites simultaneously. The selected RTE protocol stacks are initially commissioned and run without applying optimisations, neither to the platforms operating system nor to the RTE protocol stack implementations. In order to reach a higher degree of determinism, a set of optimisations is applied, which mainly affects the network traffic fitting mechanism and the operating system scheduler. The impact of the various optimisations is documented by several conducted measurement series.

II. RTE APPLICATION CORNER STONES

Typical scenarios of RTE applications involve controlling nodes, commonly in the form of programmable logic controller (PLC) devices, and input/output (I/O) devices. The latter serve as network interconnectivity port. Their role is to receive data, targeted for actuator devices and/or to transmit data which originate from sensor devices. Actuators and sensors are either an integral part of the I/O device or are connected to it through e.g. peripheral interfaces or buses. The rate at which data is packeted and transmitted over the network, respectively its reciprocal, the "cycle time", is an important key figure for the conception and configuration of a RTE application. Industrial *human control*[1] systems operate with data refresh cycle times ranging from the domain of several seconds down to $\approx 8\,\mathrm{ms}$. Mechanical machinery *process control*[1] commonly operates in the range of several milliseconds with shortest data refresh cycle times of $\approx 4\,\mathrm{ms}$. The sub millisecond threshold is undercut for high precision, possible synchronous, *motion control*[1] applications, e.g. assembly robot coordination and operation among a car assembly line, high grade printing machinery, et cetera [11]. Data refresh cycle times for such applications reach dimensions of several or even less than $100\,\mu\mathrm{s}$. Such high data refresh frequencies usually require tailor made hardware features as well as isochronicity. A shared time base across a RTE application network (clock domain) is established and maintained through high precision time synchronisation protocols that are described in standards such as IEEE 1588 [12], IEEE 802.1AS [13] or proprietary derivatives of the latter. Within the scope of this work, the main focus lies on the I/O device.

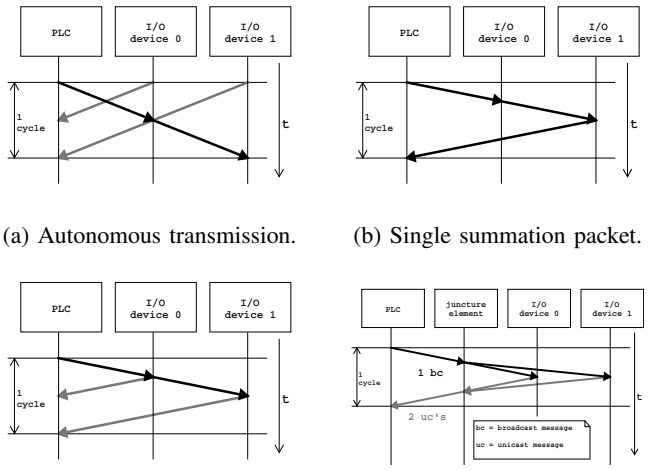
III. RTE PROTOCOL SUITES

The International Electrotechnical Commission (IEC) standards 61158 and 61784-2 include real time communication profiles which are based on Ethernet technologies. The communication profile families [10, table 21.3] (CPF) are named by the brand name of the according practical implementation of the RTE protocol suite. Some of these protocol suites, referenced by brand name, are briefly described in the following.

A. EtherCAT

IEC 61158-6-CPF12 (for the application layer protocol) and in IEC 61784-2, profiles 12/1 and 12/2 [10, tab. 21.4]. The specification is maintained by the EtherCAT Technology Group (ETG). The main protocol data unit (PDU), is a so called "EtherCAT frame" [14]. The PDU is preferably transported directly within the data link layer. Possibilities to transport the main PDU with a user datagram (UDP) or even a stream (TCP) exist [15]. Ethernet for Control Automation Technology (EtherCAT) uses master and slave application communication roles. The EtherCAT master node sends a telegram which traverses through all the EtherCAT slave nodes and then travels back to the EtherCAT master node [16], see fig. 1b. Typical physical implementations of EtherCAT network segments use a daisy chain or a ring topology based on switched Ethernet. From a communication point of view, the cyclic real time data exchange is driven solely by the master device [17, sec. II.A, para. 2]. The master sends one frame at a given cycle time. Whilst the frame progresses through the slave nodes, they read the master output data and write the master input data at their a priori defined octet offset in the frame. The application protocol layer uses methods such as reading or writing values from/to addressable registers which is EtherCAT specific. Another method is the use of "CANopen over EtherCAT" (CoE) which adopts the CANopen object dictionary [14].

[1]The terms (*human control*, *process control* and *motion control*) for classifying a real time Ethernet application are derived from [10, sec. 21.3.1].



(a) Autonomous transmission. (b) Single summation packet.

(c) Token passing. (d) Explicit request/response.

Fig. 1: Communication methods for cyclic data exchange.

B. PROFINET Input/Output

IEC 61158-6-CPF3 (for the application layer protocols) and in IEC 61784-2, profiles 3/4, 3/5 and 3/6 [10, tab. 21.4]. The PROFINET Input/Output (PNIO) (herein after referred to as PROFINET) real time protocol suite specification, [18], is maintained by PROFIBUS & PROFINET International (PI) [19]. Process Field Net (PROFINET) includes multiple protocols in its suite which operate on different layers of the Open Systems Interconnection (OSI) model. For setup and configuration (non real time communication), e.g.: On the transport layer, typically through datagrams (UDP), with the distributed computing environment / remote procedure call (DCE/RPC) application protocol. On the data link layer with the discovery and basic configuration protocol (DCP). Real time application data exchange is prevalently performed on the data link layer. PROFINET mainly employs two application communication roles: Controller and I/O device. A controller (PLC) sets up an exclusive "application relation" (AR) to each of the RTE application participating I/O devices. During operation, the two parties of the respective application relation autonomously exchange their real time application (I/O) data at an initially configured cycle time, see fig. 1a.

Typical network topologies that are used to operate real time applications through PROFINET are daisy chain, ring, or star, based on switched Ethernet. The application data protocol layer is PROFIBUS Nutzer Organsiation (PNO) proprietary and used for non Ethernet carriers as well, e.g. PROFIBUS.

C. Ethernet POWERLINK

IEC 61158-6-CPF13 (for the application layer protocol) and in IEC 61784-2, profile 13/1 [10, tab. 21.4]. The Ethernet POWERLINK (EPL) specification is maintained by the Ethernet POWERLINK Standardization Group (EPSG). EPL was originally designed for shared Ethernet and to be operated with Ethernet hubs as juncture elements. Two main real time application participating roles are distinguished: Managing node (MN), master, and controlled node (CN), slave. EPL employs a request-response message sequence to exchange real time application data, which implies that, the communication schedule is driven by the managing node only [20, sec. 4.2.4.1]. During operation, Ethernet POWERLINK adheres to a preliminary defined "isochronous cycle". The cycle is sliced into the following phases: Start period, isochronous period, asynchronous period and lastly an idle period [10, fig. 21.2], [20, fig. 19]. After the managing node has sent a start-of-cycle message (multicast frame), it sequentially polls every controlled node in order to exchange real time application data. Afterwards, the MN sends a start-of-asynchronous message, as indication for all the CNs that all real time application data has been exchanged within the isochronous period of the current cycle. EPL operates directly on the data link layer for the real time application data exchange. Asynchronous messages are preferably transported over IP/UDP [20, p. 28, fig. 4], [10, sec. 21.4.2.1]. The EPL communication profile is based on CANopen communication profiles [20, p. 26].

D. CC-Link Industrial Ethernet

IEC 61158-6-CPF8 (for the application layer protocol) and in IEC 61784-2, profiles 8/4 and 8/5 [10, tab. 21.4]. The CC-Link specification is maintained by the CC-Link Partner Association (CLPA). From an application layer point of view, Control & Communication Link (CC-Link) Industrial Ethernet (IE) technologies abstract the distributed real time application (participants) as *"Network Shared Memory"* [21]. This means that the cyclic real time application data exchange mainly consists of reading from and/or writing to a "memory location". Basically, CC-Link IE is operated directly on the data link layer and uses master and slave application communication roles. The concrete implementation forms fundamentally differ, depending on the technology. CC-Link IE Field, commonly used to operate a single assembly line, and CC-Link IE Control adopt a token passing method, where a token is passed from one station to the next, [22], [23]. Only the station which currently possesses the token is allowed to transmit data, see fig. 1c. The path of the token logically builds a ring structure. CC-Link IE Field Control is used to operate factory backbones, [21]. CC-Link IE Field Basic is a CC-Link variant which is intended for the operation on top of an internet protocol (IP) stack. Real time application data is transported within user datagrams (IP/UDP) where the PLC (master) device cyclically polls the slave devices through broadcast packets, see fig. 1d. CC-Link IE technologies are designed to be used on daisy chain, ring or star network topologies, based on switched Ethernet. CC-Link IE Field Basic is used with daisy chain and star topologies or a mixture of both, [24, p. 10].

E. EtherNet/IP

IEC 61158-6-CPF2 (for the application layer protocol) and in IEC 61784-2, profiles 2/2 and 2/2.1 [10, tab. 21.4]. Ethernet Industrial Protocol (EtherNet/IP) is defined and maintained by the Open DeviceNet Vendors Association, Inc. (ODVA). Typical operation for cyclic I/O data exchange takes place on top of an internet protocol (IP) stack with the user datagram protocol (UDP) on the transport layer. For the connection establishment process between PLC and I/O device and for infrequent low-priority messages, the transmission control protocol (TCP) is used on the transport layer. EtherNet/IP uses the object oriented common industrial protocol (CIP) on the application layer. [25], [26]

F. Modbus/TCP

IEC 61158-6-CPF15 (for the application layer protocol) and in IEC 61784-2, profile 15/1 [10, tab. 21.4]. Modbus/TCP is promoted by the Modbus Organisation [11]. As the name lets assume, during default operation, periodic data exchange is performed on top of an IP protocol stack in a request-response manner. Connections between application peers have the form of a data stream, which means, using TCP on the transport layer. The application layer is realised through the proprietary Modbus application protocol [27]. Modbus/TCP is one of the most widely used [10, sec. 21.4.1.1] RTE protocol suites in the range of *human control*, see section II, real time applications.

IV. Domains of scheduling

Despite the variation of the communication methods and the hook-in points at different layers of the OSI model, see section III, a unified separation of different scheduling domains is observed across real time Ethernet protocol suites. The common denominator of all the scheduling domains is, that they have to adhere to the specific desired data refresh cycle time of the distributed RTE application.

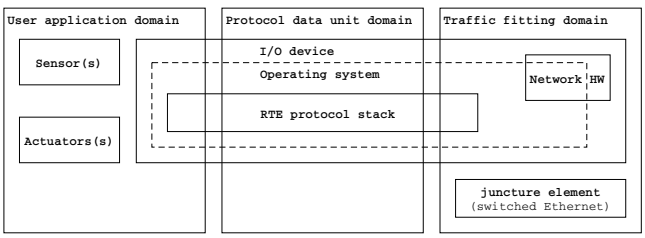


Fig. 2: Scheduling domains applied to a RTE application.

A. User application domain

The predominant task of a user application that runs on a I/O device is the interaction with either sensor(s) (reading) and/or actuator(s) (writing). Besides, data pre processing steps are sometimes necessary to be carried out before sending data over the RTE network to the controlling node or transfer them to an actuator unit. Usual limiting factors are interface or bus data rates of peripheral connections (e.g.: inter-integrated circuit (I^2C), serial peripheral interface (SPI), universal asynchronous receiver-transmitter (UART), ...) as well as processing resources of sensor/actuator periphery.

B. Protocol data unit domain

As yielded throughout section III, RTE protocol suites tend to utilise proprietary application protocol layers where individual application PDUs are defined by the respective standards. An essential task of this scheduling domain includes the marshalling of sensor data (reading) and actuator data (writing) into or out of PDU(s) of the actual RTE protocol suite. From an I/O device perspective: *Reading* includes real time data transition from the user application domain (e.g.: sensor input) to the PDU domain and preparation of the PDU(s) for the transition to the traffic fitting domain. *Writing* includes PDU(s) transition from the traffic fitting domain to the PDU domain and preparation of the real time data for the transition to the user application domain (e.g.: actuator output). Tasks of the PDU domain are typically performed by the specific "real time Ethernet protocol stack", which is a concrete implementation of a standardised RTE protocol suite. Limiting factors are the hardware characteristics of the I/O device, e.g. CPU capabilities as well as the limitations of the OS, if any, that runs on top of the hardware of the I/O device.

C. Traffic fitting domain

The term traffic fitting, as used in this present work, is understood as the process of transmitting and receiving real time data with an a priori known RTE application cycle time. Fitting means the introduction of techniques which have the aim to guarantee the adherence to the previously mentioned data refresh cycle time. Traffic fitting techniques are manifold and may be part of network hardware directly on the I/O device or indirectly, provided through network juncture elements. I.e.: Multi network port I/O devices integrally implement an Ethernet switching MAC layer. Single port I/O devices typically access a network through an outsourced Ethernet switching unit and implement an Ethernet endpoint MAC layer.

D. Inter domain transitions

If the chosen hardware platform of the I/O device allows it to run the domains of scheduling autonomously, e.g. with dedicated hardware units or real multithreading (threads are capable of running concurrently on separate CPU cores), mechanisms for synchronising data transitions between the scheduling domains have to be used in order to maintain data consistency. An exemplary mechanism is triple buffering, where a data producing domain writes alternately to two back buffers and a data consuming domain reads from one front buffer. An advantage of this technique is, that the mutual exclusion phase is (timely) minimised down to the rotation of the buffer addresses[2].

V. Time sensitive networking

TSN is an umbrella term for several specific standards (with IEEE 802.1Q as base standard) which are amendments to standard Ethernet, IEEE 802.3*. The goal of these addenda is to provide deterministic connectivity for IEEE 802 networks [28]. The operating range where the TSN techniques have their main impact is on the Ethernet MAC layer. In the context of scheduling domains, TSN mechanisms are applied at the traffic fitting domain, see section IV-C. Techniques described in the TSN IEEE standards enable the coincident orchestration of "mixed critical" data traffic within one network infrastructure. Data traffic characterisation classes: *Best effort*, *reserved* (or rate constrained) and *scheduled traffic* [29]. One base feature introduced by IEEE 802.1Q are quality of service (QoS) levels [30, p. 1920] which can be used to prioritise network packets within an IEEE 802.3* network. In order to guarantee deterministic latency along network paths, several traffic shaping mechanisms are engineered. Some of which are briefly touched on here: For *reserved* traffic, the credit based shaper (CBS) (IEEE 802.1Qav, [31]) algorithm allows to guarantee a certain data rate, mapped to QoS levels. The asynchronous traffic shaper (ATS) is suited for preferring asynchronously emerging data packet bursts, e.g. non real time system logging (monitoring) data [32], [33]. For *scheduled traffic*, often required to operate e.g. motion control RTE applications at cycle times below $1\,\mathrm{ms}$, the time aware shaper (TAS) (IEEE 802.1Qbv, [34]) is a suitable choice [35]. The TAS requires a common time base across the RTE application participants which has to be established through a time synchronisation protocol, e.g. IEEE 802.1AS [13].

A. IEEE standardised extended MAC layer

The TSN standards bundle a base layer of IEEE standardised technologies. These are applicable to real time Ethernet applications by providing an augmented, non-proprietary Ethernet MAC layer. Before the introduction of the TSN IEEE Ethernet amendments, specific individual amendments were conceived, described, developed and maintained by the RTE protocol suite committees and their affiliates. This comes with the implication, that RTE applications which require *scheduled traffic* (exclusive fixed time slice / phase allocation per communication cycle) might require tailor made hardware implementations that are bound to one specific RTE protocol suite. The TSN standards on the other hand provide a more generic approach which makes RTE hardware platforms better portable across RTE protocol suites, if they have implemented

[2]The triple buffer is a last-in-first-out (LIFO) data structure with a depth of two between two autonomously operating units of scheduling. Reference implementations at the InES exist in a hardware description language (FPGA entity) as well as in C for POSIX.1-2001 compliant operating systems.

the required TSN standards for the specific RTE application use case.

B. Industry acceptance

An ongoing acceptance and partial adoption of techniques stemming from the TSN standards bundle is observed across RTE protocol suite maintenance organisations. PROFINET selectively introduces a subset of TSN standards such as the generalized precision time protocol (gPTP), IEEE 802.1AS [13], in combination with the TAS, IEEE 802.1Qbv [34], to run *"clock-synchronous motion control applications"* on the basis of standard IEEE 802.3* (partially) TSN capable hardware. Further, PROFINET adopts the mechanism of "frame preemption", IEEE 802.1Qbu [36], which allows high prioritised cyclic real time Ethernet frames to preempt low prioritised best effort Ethernet frames while they are already in the process of transmission on a network port [37], [38]. Similarly, CC-Link (IE TSN) introduces gPTP, IEEE 802.1AS [13], combined with the time aware shaper, IEEE 802.1Qbv [34], in order to carry mixed critical data traffic on top of one network hardware infrastructure [39].

VI. Demonstrator

After an overview of different RTE protocol suites has been gained, see section III, common scheduling domains could be identified, see section IV. Further, the IEEE TSN standards provide a versatile tool box to engineer a generic Ethernet MAC layer, see section V.

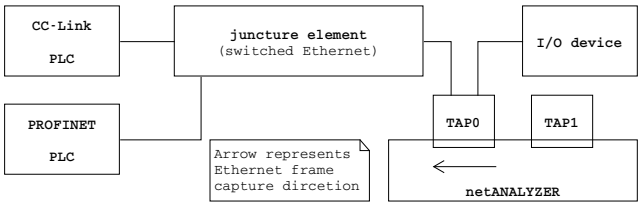


Fig. 3: Basic setup of the RTE demonstrator application.

In this section, an experimental RTE application scenario is presented which consists of a single juncture element (switched Ethernet), two PLC devices, PROFINET Input/Output and CC-Link Industrial Ethernet Field Basic and one single I/O device. The I/O device serves both PLC devices simultaneously by maintaining cyclic real time application data exchange for the two unsimilar RTE protocol suites.

A. Selection of the RTE protocol suites

The choice of the two RTE protocol suites has been made up on two major considerations: 1) PROFINET is a Europe originating, globally accepted RTE protocol suite with Siemens as prevailing influencer. Its interest group, PROFIBUS & PROFINET International, has about 1700 members [19]. CC-Link has its origins in Asia and was introduced by the Mitsubishi Electric Corporation. The CC-Link Partner Association (CLPA) also spans globally and has currently over 4000 corporate members [40]. 2) The cyclic data exchange between PLC and I/O device relies on different communication methods – the PROFINET participants transmit autonomously, see fig. 1a, whereas the CC-Link PLC broadcasts request messages and expects explicit unicast response messages from the I/O device, see fig. 1d. Further, the PNIO application protocol is transported directly within the data link layer whereas the used CC-Link variant, CC-Link Industrial Ethernet Field Basic, transports its application protocol on top of an IP protocol stack with UDP as transport layer.

B. Vanilla approach

The term "vanilla" means that a hardware platform or software is brought into operation, not (or only slightly [41]) altered from its original form, how it was distributed from its suppliers [42]. Applying this modus operandi to the components of the demonstration application, as part of this work, it is understood as using the hardware components "off-the-shelf" as well as using them with their shipped default OS distribution with a pre-compiled operating system kernel image installed. All the modifications introduced are entirely based on the available set of configuration possibilities of the existing hardware and operating system.

C. Application scope

The emphasis of the demonstrator application lies primarily on the protocol data unit scheduling domain, see section IV-B, secondly with a focus on the traffic fitting scheduling domain, see section IV-C. The user application scheduling domain, see section IV-A, exists in a subordinate role by including sensor(s) and actuator(s) as mock objects only. The I/O device is equipped with a single Ethernet network port. The link speed is limited to $100\,\mathrm{Mbit/s}$ as requested by PROFINET [43, p. 74] and CC-Link IE Field Basic [24, p. 10]. The chosen data refresh cycle time is $1\,\mathrm{ms}$, situated at the upper bound form a *motion control* application point of view and below the lower bound from a *process control* application point of view, see also in section II.

D. Hardware and operating systems

1) PLC devices: The PLC devices are treated as black boxes within the scope of this work and are assumed to be "golden", meaning their operation is flawless in terms of adhering to timing constraints (determinism).

2) I/O device: The I/O device platform was chosen upon the consideration of providing a loose framework that relies on a generic hardware platform which, "out of the box", serves broad configuration possibilities. This allows various measurement series to be conducted and offers parameter scalability, such as a configurable CPU clock frequency, availability of multiple operating system scheduler policies, without the requirement of recompiling the OS kernel, et cetera. Elaborated configuration sets can be mapped (fitted) to more specific and constrained platforms if necessary. A suitable

platform for this framework is the Raspberry Pi (version 3, model B+) single board computing platform[3].

The operating system which runs on the I/O device platform is a debian GNU/Linux derivative[4]. The kernel is configured to make use of high resolution clock sources which are available on the I/O device hardware platform (CONFIG_SCHED_HRTICK=y). These clock sources can be accessed through OS system calls and reach a resolution of 1 ns the finest. The queueing discipline (qdisc) of the Ethernet controller for transmitting data over Ethernet is pfifo_fast[5].

3) Juncture element: The NXP LS1028ARDB is a reference design board that provides Ethernet switching functionality[6]. The Ethernet MAC layer implements a subset of TSN IEEE standards. Some functionalities, described in the aforementioned standards, are offloaded to dedicated hardware units[7]. E.g.: The CBS algorithm (IEEE 802.1Qav) can be offloaded to the network interface controller of the according network switch port. The LS1028ARDB runs a Linux kernel[8]. The queueing discipline of the Ethernet switch ports is mqprio – Multiqueue Priority Qdisc (Offloaded Hardware QOS) [53] – to take advantage of the hardware offloaded priority level queues.

E. RTE protocol stack software

The PROFINET I/O device as well as the CC-Link IE Field Basic slave device RTE protocol stack implementations are contributed by the company RT-Labs in the form of readable program source code which can be edited. Holding on to the vanilla approach, see section VI-B, both stacks were initially compiled without any modification. Both RTE stacks have a multithreaded software architecture. The source code, as delivered, offers the compilation option to use the SCHED_FIFO[9] scheduling policy which was selected to compile the sources. If the option is not selected, the threads of the RTE stacks would just use the SCHED_OTHER[10] scheduling policy.

F. Optimisations

An initial measurement series was performed whose results are shown in section VII-A. It is noticed that, when using the SCHED_FIFO configuration option, see section VI-E, the on-wire network packet cycle times, originating from the I/O device, yield a broad deviation of the expected value, 1 ms. While operating both RTE protocol stacks simultaneously, deviations as large as 1.081 ms for the PROFINET stack and 30.249 ms for the CC-Link stack are observed, see fig. 4b.

Within this chapter, optimisations are presented that have the aim of reaching a better level of determinism for operating RTE protocol stack implementations on the given hardware platforms, see section VI-D2 and VI-D3. Table I references the specific optimisations and shows in which scheduling domain, i.e. protocol data unit domain (PDU) or traffic fitting domain (TF), it takes place and which RTE hardware components, i.e. I/O device (IOD) or juncture element (JE), are affected by the respective optimisation. The measurement column of table I references the measurement results sections that substantiate the optimisation measure.

TABLE I: Relations to scheduling domain and measurements.

Optimisation	Sched. domain	Components	Measurements
Section VI-F1	PDU	IOD	VII-B, VII-C
Section VI-F2	PDU	IOD	VII-B, VII-C
Section VI-F3	TF	IOD, JE	VII-E
Section VI-F4	TF	IOD, JE	VII-E

1) Change of scheduling policy: Since version 3.14 of the Linux kernel, its scheduler provides deadline task scheduling in the form of a scheduling policy, SCHED_DEADLINE. The deadline scheduling policy guarantees[11] that during a given time period (relative deadline), a certain task receives a specific amount of CPU computation time (*runtime*). To configure time values which go below 1 ms, the scheduler must be configured to use high resolution timer ticks (HRTICK) [57]. The source code of the RTE protocol stacks was analysed. Despite the strong differences between the two RTE protocol suite variants, e.g. by using different communication methods, see

[3]The platform is a system on chip (SoC), Broadcom BCM2837B0, with an Advanced RISC Machines (ARM) Cortex-A53 reduced instruction set computer (RISC) central processing unit, a quad core CPU with variable clock frequency, ranging from [600, ..., 1400] MHz. It has 1 GB of main memory. The wired network connectivity is powered by a Microchip LAN7515 Gigabit Ethernet controller (compatible with the Ethernet core of the LAN7800 controller [44]). The controller is limited to 300 Mbit/s because it is connected to the SoC via universal serial bus (USB) version 2.0 [45]. The Ethernet controller has no hardware offloaded multi priority level transmission queues [46].

[4]The OS kernel version is 5.15.61 and it is compiled for the ARM 32-bit architecture, namely armv7l, with symmetric multiprocessing (SMP) and "voluntary preemptive" kernel code, an interstage between non and fully preemptive kernel code. Having compiled the kernel with SMP enabled means, that the CPU cores are treated equally and the entire available memory is shared between the processor cores [47].

[5]This queueing discipline uses three priority "bands" (0, ..., 2). Each band has assigned a first-in-first-out (FIFO) data structure. As long as data packets are present in band 0 to be transmitted, they are always serviced first, before band 1. The same applies to the relationship between band 1 and band 2 [48]. The pfifo_fast algorithm takes into account the "Linux priority" value that is assigned to the socket buffer (sk_buff) out of which a packet originates [49]. The mapping of sk_buff priorities to the aforementioned bands is found in [50].

[6]The reference design board is equipped with a dual core ARM Cortex-A72 RISC CPU. It offers 4 RJ-45 Ethernet ports with link speeds of 10/100/1000 Mbit/s, attached through a quad serial gigabit media-independent interface (QSGMII).

[7]Each of the Ethernet ports offers configurability for features that are defined within the set of TSN standards: IEEE 802.1Qbv [34], IEEE 802.1Qci [51], IEEE 802.1Qbu [36], IEEE 802.1Qav [31], IEEE 802.1CB [52]. Each switch port is equipped with eight hardware offloaded priority level transmission queues.

[8]The Linux kernel version is 5.15.5. It was compiled for the ARM 64-bit architecture, namely aarch64, with "real time preemptive" kernel code and symmetric multiprocessing (SMP) enabled.

[9]SCHED_FIFO (a FIFO scheduling algorithm) is a greedy real time policy and prohibits the preemption through tasks with lower or equal priority. The task might be preempted by tasks with higher priority. Therefore this policy cannot guarantee to meet a certain time deadline. [54]

[10]This is the default policy that a newly spawned process/thread gets on a portable operating system interface (POSIX) compliant operating system [55].

[11]When setting the SCHED_DEADLINE scheduler policy through the sched_setattr() system call [56], a so called admittance test is performed by the operating system kernel. The admittance test calculates if the scheduling policy change, with the requested runtime and deadline parameters, is feasible. If not, the system call will fail [54].

fig. 1a and fig. 1d, or by hooking in at fundamentally different locations of the OSI reference model, see section III-B and III-D, both implementations have a similar software architecture. The RTE protocol stack implementations use an internal application scheduler, driven by a POSIX timer whose callback function periodically[12] sets an event. A main application thread checks for pending events and processes them. The thread which handles the POSIX timer and the main application thread[13] are altered to use the `SCHED_DEADLINE` scheduling policy. Table II shows the configuration parameters for the `SCHED_DEADLINE` policy of the affected RTE protocol stack threads.

TABLE II: `SCHED_DEADLINE` configuration parameters.

Thread	Runtime	Deadline	Period
PNIO POSIX timer	100 µs	1000 µs	1000 µs
PNIO main application thread	200 µs	1000 µs	1000 µs
CC-Link POSIX timer	100 µs	1000 µs	1000 µs
CC-Link main application thread	200 µs	1000 µs	1000 µs
Total runtime within period	**600 µs**	**–**	**–**

The guaranteed *runtime* parameter for the main application thread is selected according to the results of the execution time evaluation, see section VII-C.

2) Exclusive computing resource allocation: In order to exclusively allocate one or more CPU cores for the deadline scheduling processes, see section VI-F1, "Linux control groups" (`cgroup`[14]s) are used. For the current application setup where two RTE protocol stacks are intended to run simultaneously, one `cgroup` is created. One CPU core is exclusively allocated to the `cgroup`. All the processes whose scheduling policy was changed to `SCHED_DEADLINE` are assigned to the `cgroup`.

3) Prioritisation of data packets: The cyclic real time network data packets which are exchanged between the PLC devices and the multi RTE protocol stack I/O device are extended with a IEEE 802.1Q virtual local area network (VLAN) tag, consisting of 4 octets. The VLAN tag contains a sub field with a width of 3 bits that represents the "priority code point" (PCP). The priority value (QoS level) that was chosen for the real time data packets is 6, according to [18, p. 100525]. Both RTE protocol stacks are equal in significance, therefore both implementations use the same PCP value. Further, the network sockets which handle the real time network packets are configured with a Linux Priority (`SO_PRIORITY`) value of 6 which stands for "Interactive" traffic and maps into the highest prioritised band of the `pfifo_fast` queueing discipline, see also section VI-D2.

4) Traffic shaping on juncture element: The selected Ethernet switch, see section VI-D3, provides support for the credit based shaper (CBS) algorithm which is described in IEEE 802.1Qav [31]. The switch ports are configured to guarantee the required bandwidth for the prioritised real time data packets, see above in section VI-F3 through the CBS algorithm. During packet ingress at the switch ports, the PCP value, carried by the IEEE 802.1Q VLAN tag, is mapped to a Linux Priority, 6. Two traffic classes are defined (bands), 0 and 1. Band 0 is always serviced first [50]. The Linux Priority 6 is then mapped to band 0 and four hardware offloaded transmission queues are assigned to band 0. All the other priorities are mapped to band 1.

G. Scalability

The combined optimisations on the I/O device, namely using deadline scheduling and exclusive CPU allocation, see VI-F1 and VI-F2, can be described in a more generic context:

Let S be the set (`cgroup`) of processes (p) that take on the scheduling policy `SCHED_DEADLINE`. Let $c \in \mathbb{N}^*$ be the number of CPU cores that are exclusively allocated for S. Let T be the unified static period duration.

$$\sum_{p \in S} p.sched\_runtime \quad \leq \quad T \times c \tag{1}$$

$$\forall p \ \in \ S \ : \ (\ p.sched\_runtime > 0 \ ) \ \wedge \ (\ p.sched\_runtime \leq T \ ) \tag{2}$$

The statement in eq. (1) must hold. Otherwise the cumulated guaranteed CPU time would exceed the effective available computation resources of the given `cgroup` (S). The statement in eq. (2) must hold. A single processes guaranteed runtime must not exceed its defined period duration [54]. The scalability considerations are made mainly from a user space perspective. If $c > 1$ and multiple real time data processing cores simultaneously request data transmission over the network, it is a matter of the queueing discipline (`qdisc`) that is configured for the affected network interface. The implementation of the `qdisc` itself depends on the capabilities of the used Ethernet controller on the hardware platform, e.g. offloaded packet transmission queues et cetera.

VII. Measurement results

Single stack means, a specific RTE protocol was in sole operation on the I/O device during the measurement data were captured. Dual stack means that both RTE protocol stacks were in simultaneous operation on the I/O device during the measurement data capturing.

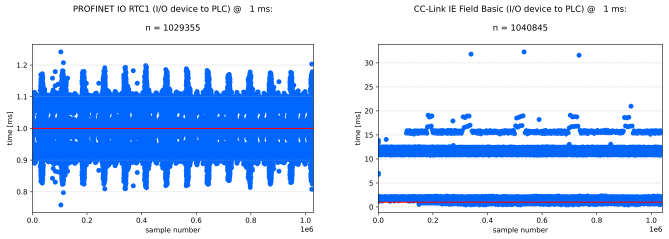
A. Initial scatter plots

The initial scatter plots, see fig. 4, depict the network packet intermediate times during cyclic real time data exchange. The red line is at the theoretically perfect (expected) cycle time, that is 1 ms. The on-wire packet capture setup is according to fig. 3, see `TAP0`.

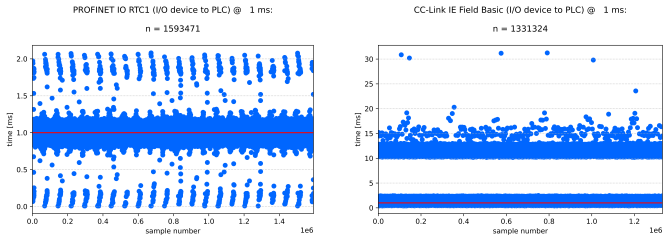
[12]The scheduler tick period of the current RTE protocol stack implementations is a configurable user application parameter. It was selected to be $\frac{cycle\_time}{2}$ (with $sched\_tick\_interval \mid cycle\_time$ in mind), according to the WCET evaluation of the cyclic real time data processing times, see section VII-C, i.e. $\frac{1\,\mathrm{ms}}{2} = 500\,\mu\mathrm{s}$. Having a lower scheduler tick interval than the actual RTE application cycle time prevents ingress packet overruns in the case of CC-Link IE Field Basic which only transmits cyclic real time data if a pending request packet is present, see also fig. 1d.

[13]Cyclic real time network data transmission (from a user space point of view) is handled by this thread, i.e. writing to the socket buffer.

[14]*"A cgroup is a collection of processes that are bound to a set of limits or parameters defined via the cgroup filesystem."* [58]



(a) Single RTE stack operation.



(b) Dual RTE stack operation.

Fig. 4: Unmodified RTE stacks, on-wire cycle times.

B. Deadline scheduling and computing resource allocation

With regard to the deterministic behavior of the I/O device platform (including its operating system), a generic system performance evaluation (derived from [59]) series was conducted. Two processes are spawned. Each process runs a measurement loop with 1×10^6 (N) iterations. To minimise other influences and capturing mainly the overhead of the OS scheduling, the body of one iteration only contains: Suspending the process operation until an expected time ($t\_exp_n = t\_exp_{n-1} + 500\,\mu s$) and, after the process is rescheduled by the operating system kernel, retrieving the current time (t_cur). The relative error is the Δ between the current time and the expected time ($e\_rel = |t\_cur - t\_exp_n|$). During the measurement, no computationally intensive tasks are running, the operating system only manages its basic background tasks. The result plot, see fig. 5, shows a comparison of different Linux scheduling policies, once performed without exclusive CPU allocation (pinning) and once with pinning the two concurrently running processes to one exclusively allocated CPU core. Each graph consists of the cumulated samples over the two processes, i.e. 2×10^6 samples per graph.

The probability plot (fig. 5) has to be interpreted as follows: E.g.: Approximately 99.5% of all the captured samples of the relative error in the case of the `SCHED_DEADLINE` policy in fig. 5 are around $20\,\mu s$ or lower. The earlier the graph maxes out on the abscissa, the better degree of determinism is reached, i.e. "`SCHED_DEADLINE` pinned".

C. RTE protocol stack execution time analysis

This measurement series was performed in order to evaluate a suitable *runtime* parameter for the deadline scheduling policy. The execution times are evaluated by time stamping the thread which is responsible for the cyclic real time data

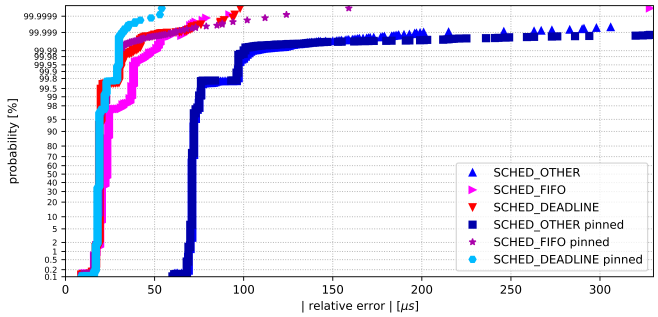


Fig. 5: I/O device platform – scheduling evaluation.

processing. One sample is the lapsed time from the start of the cyclic processing routine until the end of it.

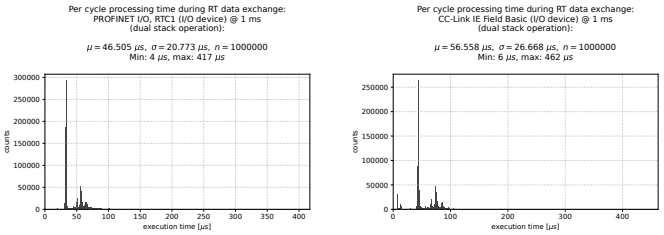


Fig. 6: RTE protocol stack cyclic processing execution times.

The two execution time evaluation histograms (fig. 6) were captured on unchanged versions of the RTE protocol stack implementations, compiled with the option to use the `SCHED_FIFO` policy with real time priorities for its threads. Since the demonstrator targets simultaneous RTE stack operation, the samples were captured during dual stack operation. The execution times are not normally distributed. With the results from the scheduling performance evaluation in mind, see section VII-B, a significant improvement in terms of determinism is observed, when using the `SCHED_DEADLINE` policy. It was chosen to prefer an estimate for the *runtime* parameter, rather than just selecting the worst case execution time (WCET) that was yielded by the measurement series because the WCET is a statistical outlier in each of the captured measurements. The estimate is defined as follows: $runtime = \lceil \mu + 3\sigma \rceil$ where $runtime$ is expressed in hundreds of µs (1×10^{-4} s). The ceiling function was applied to the expression to have a safety cushion, affordable for the current requirements. For the PROFINET RTE protocol stack, this yields a runtime value of $200\,\mu s$, for the CC-Link RTE protocol stack a runtime value of $200\,\mu s$.

D. Optimised dual RTE stack operation

Figure 7 depicts the scatter plots as well as the histograms that visualise the on-wire captured samples, according to the capture setup shown in fig. 3. All the measures that were taken, see section VI-F, to optimise the deterministic operation of the given demonstrator application were enabled during the capturing of the samples.

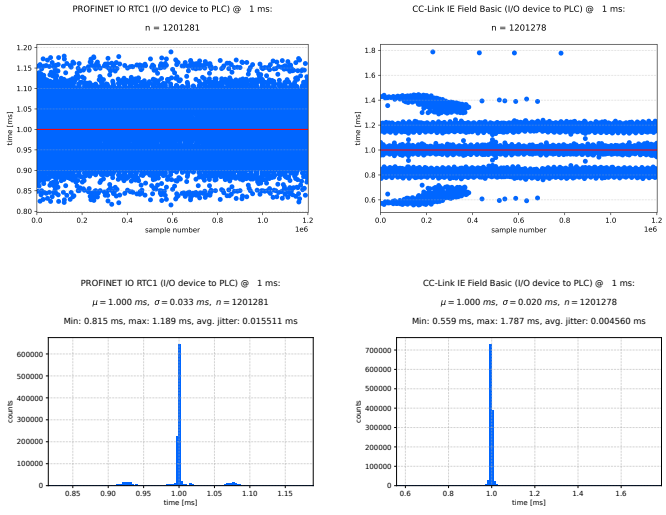


Fig. 7: Optimised dual RTE stack, on-wire cycle times.

Table III compares the measurement figures from the initial situation, unmodified dual RTE protocol stack operation, with the ones from the on-wire cycle time captures of the optimised demonstrator. The capturing setup is depicted in fig. 3.

TABLE III: Initial vs. optimised dual RTE stack operation.

Variation	**Avg. [ms]**	**σ [ms]**	**Max. [ms]**	**Avg. jitter [ms]**
PNIO unmodified	1.000	0.045	2.081	0.021913
PNIO optimised	1.000	0.033	1.189	0.015511
CC-Link unmod.	1.197	1.103	31.249	0.356171
CC-Link opt.	1.000	0.020	1.787	0.004560

E. Robustness audits

To get an order of magnitude about the robustness of the optimised demonstrator application, two methods are introduced which have the intention to distort the cyclic real time data exchange. The demonstrator application withstands all the robustness tests, described in VII-E1 and VII-E2.

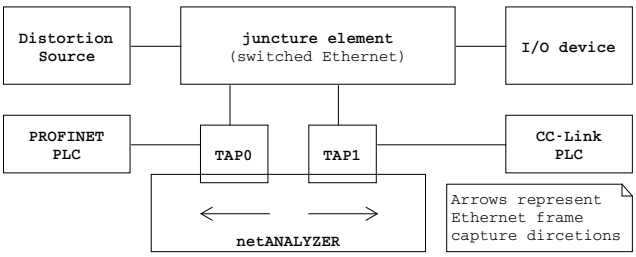


Fig. 8: Demonstrator application distorted operation setup.

1) PROFINET netload: Distortion through a test agent device which interferes the RTE traffic of the demonstrator application. The interfering data traffic is produced by the load generator that is used for the "PROFINET Netload Robustness for Security" test, formerly known as "PROFINET Security Level 1" test. The generated load adheres to the "Netload Class 3 @ 100 Mbit/s", which is briefly described as *"Advanced robustness against netload"* [60]. The PROFINET netload test is an industry accepted robustness test for I/O devices and serves as useful stress test during the development (continuous integration) of an I/O device. It is also part of the PROFINET certification process that certified PROFINET I/O devices have to undergo.

2) `iperf3`: Distortion through a test agent device which interferes the RTE traffic of the demonstrator application. The interfering data traffic is produced by the network performance measurement program named `iperf3` [61]. In default operation of `iperf3`, a server program is started on one host (e.g. I/O device) and a load generator program is started on another host (e.g. test agent). The load generator uses the maximum possible bandwidth of the link speed towards the server program (IP/TCP). The intention behind the usage of `iperf3` is, that a maximum level of distortion within the juncture element as well as on the I/O device (device under test) is reached. In contrast to section VII-E1 above, `iperf3` additionally requires the launch of a server program that has to be installed on the I/O device.

3) Effect of the CBS algorithm: The impact of the credit based shaper algorithm on the traffic characteristics of the cyclic real time data packets during distorted, see VII-E2 above, operation is shown by the traffic pattern plots in fig. 9[15]. The packet capturing setup is according to fig. 8. The packet schedule is in proper order despite the distortion when using enhanced Ethernet with a TSN traffic management algorithm, i.e. IEEE 802.1Qav.

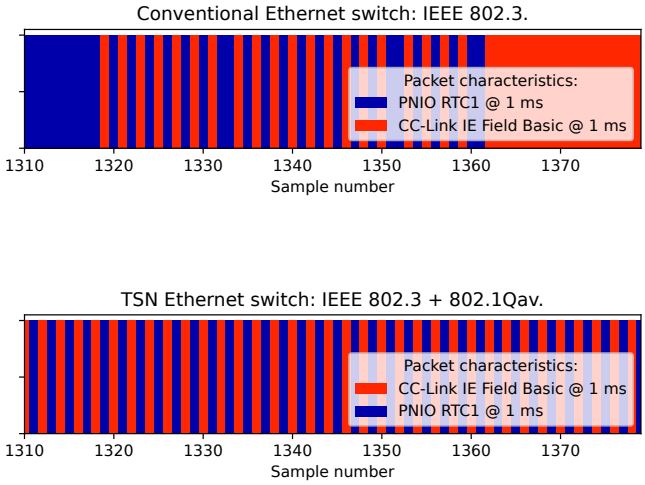


Fig. 9: Effect of the CBS algorithm.

[15]The "conventional Ethernet switch" which was used for this measurement series is a TP-Link TL-SG108, a 8 port store and forward unmanaged IEEE 802.3* standard Ethernet switch that supports link speeds of 10/100/1000 Mbit/s with a switching capacity of 16 Gbit/s. The switch device is not configurable at all. [62]

VIII. Conclusions

After retaining important characteristics of a RTE application, an inquiry was made across widely used RTE protocol suites which yielded individualities and commonalities. Time sensitive networking IEEE standards are evolving into a description of a RTE protocol suite independent Ethernet MAC layer, also for high demanding timing constraints. A demonstrator application was developed with the goal of having a generic environment which is capable of (even simultaneously) running different RTE protocol stacks. Measurements were carried out to undergird technical decisions during the demonstrator development. Five main conclusions are drawn and presented in the following enumeration:

1) A configurable and scalable RTE application setup can be built to simultaneously run multiple proprietary RTE protocol suites on one and the same I/O device. Its operation is enabled by arranging an appropriate configuration set for the computing resources of the targeted I/O device hardware platform.
2) Coexistence of multiple RTE protocol suites on one device comes with a gain of flexibility and eases commissioning for I/O device platform manufacturers with regard to offer compatibility for various RTE protocol suites.
3) TSN technology provides a generic (non-proprietary), IEEE standardised extended Ethernet MAC layer. This entails a unification for I/O device platform manufacturers. By implementing a suitable subset of TSN IEEE standards, a common base layer can be established for taking on the tasks of network traffic fitting. Individual RTE protocol suite implementations profit by using the features supplied by the underlying extended media access control layer, instead of implementing proprietary specifications.
4) Optimising the RTE protocol stack implementations with a focus on the scheduling policy (`SCHED_DEADLINE`) and explicit processor core allocation (`cgroup`) raises the level of determinism for cyclic real time network packets. Maximum deviation values from the configured (expected) cycle time (i.e.: $1\,\mathrm{ms}$) are reduced by a factor of $\approx\ 5.7$ for a PROFINET protocol stack and by a factor of $\approx\ 38.4$ for a CC-Link IE Field Basic protocol stack during simultaneous RTE protocol stack operation.
5) Despite the individuality of different RTE protocol suites, e.g. the usage of different communication methods and OSI layers, three unified domains of scheduling can be identified: The user application domain, the protocol data unit domain and the traffic fitting domain. These domains can be, if the target hardware platform allows it, operated autonomously and must adhere individually to the configured RTE application data refresh cycle time.

IX. Acknowledgement

We acknowledge the contribution of source code of real time Ethernet protocol stack implementations by RT-Labs AB, Sweden with gratitude.

References

[1] Molex, LLC in the United States of America, *Software Development Kits*, 2022 (accessed October 30, 2022). [Online]. Available: https://www.molex.com/molex/products/family/profinet_protocol_stacks_sdks
[2] RT-Labs, "Pricing and licenses," 2022, (accessed October 30, 2022). [Online]. Available: https://rt-labs.com/products/pricing-and-licenses/
[3] KUNBUS GmbH, "Protokollstacks für am64x und am243x," 2022, (accessed October 31, 2022). [Online]. Available: https://www.kunbus.de/protokollstacks
[4] Weidmüller Interface GmbH & Co. KG, *I/O System IP20 - u-remote*, 2023 (accessed February 10, 2023).
[5] Institute of Embedded Systems, *easyIRT*, 2015 (accessed July 16, 2021). [Online]. Available: https://www.zhaw.ch/storage/engineering/institute-zentren/ines/forschung-und-entwicklung/realtime-ethernet/profinet.pdf
[6] Softing Industrial Automation GmbH, "Softing protocol ip - profinet," 2016, (accessed November 1, 2022). [Online]. Available: https://industrial.softing.com/uploads/softing_downloads/ProtocolIP_PN_D_EN_160501_300_01.pdf
[7] Siemens AG, "Ertec 200p – data sheet," 2014, (accessed November 2, 2022). [Online]. Available: https://cache.industry.siemens.com/dl/files/421/71879421/att_105246/v1/DataSheet_ERTEC200P_V11.pdf
[8] Hilscher Gesellschaft für Systemautomation mbH, "netx 51," 2014, (accessed November 1, 2022). [Online]. Available: https://www.hilscher.com/fileadmin/cms_upload/en-US/Resources/pdf/netX-51_Datasheet_11-2014_GB.pdf
[9] Texas Instruments Incorporated, "Quad-core arm cortex-r5f-based mcu with industrial communications and security up to 800 mhz," 2022, (accessed October 31, 2022). [Online]. Available: https://www.ti.com/product/AM2434
[10] M. Felser, "Real-time ethernet for automation applications," *Industrial Communication Technology Handbook, Second Edition*, 06 2009.
[11] H. D. Doran, "Research paper – security in industrial ethernet," 2021, unpublished.
[12] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2019 (Revision of IEEE Std 1588-2008)*, pp. 1–499, 2020.
[13] "IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications," *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, pp. 1–421, 2020.
[14] EtherCAT Technology Group (ETG), *EtherCAT Device Protocol (EDP) Overview*, 2020 (accessed October 12, 2022).
[15] www.ethercat.org, "Ethercat for factory networking," *PC Control – The New Automation Technology Magazine*, pp. 1 – 4, 2009.
[16] EtherCAT Technology Group (ETG). Ethercat – the ethernet fieldbus. [Online]. Available: https://www.ethercat.org/en/technology.html#1.1
[17] D. Ganz, S. Leschke, and H. D. Doran, "Improving ethercat master-slave synchronization precision using ptcp embedded in ethercat frames: A proof-of-concept," in *2015 IEEE World Conference on Factory Communication Systems (WFCS)*, 2015, pp. 1–7.
[18] PROFIBUS Nutzerorganisation e. V. (PNO), *Application Layer protocol for decentralized periphery – Technical Specification for PROFINET*, version 2.4mu1, order no.: 2.722 ed., 2020.
[19] ——, *About PI*, 2023 (accessed February 10, 2023). [Online]. Available: https://www.profibus.com/pi-organization/about-pi
[20] Ethernet Powerlink Standardization Group, *EPSG Draft Standard 301 – Ethernet POWERLINK – Communication Profile Specification*, 2013 (accessed November 7, 2022). [Online]. Available: https://www.ethernet-powerlink.org/fileadmin/user_upload/Dokumente/Downloads/TECHNICAL_DOCUMENTS/EPSG_DS_301_V-1-2-0_02__4_.pdf
[21] CC-Link Partner Association, "CC-Link IE Control Network," undated, (accessed November 9, 2022). [Online]. Available: https://www.cc-link.org/en/cclink/cclinkie/cclinkie_c.html
[22] ——, "CC-Link IE Field – Specifications," undated, (accessed November 9, 2022). [Online]. Available: https://am.cc-link.org/en/cclink/cclinkie/cclinkie_f_spec
[23] ——, "CC-Link IE Control Specification," undated, (accessed November 9, 2022). [Online]. Available: https://am.cc-link.org/en/cclink/cclinkie/cclinkie_c_spec
[24] Mitsubishi Electric Corporation, "CC-Link IE Field Network Basic Reference Manual," 2022, (accessed January 1, 2023). [Online]. Available: https://dl.mitsubishielectric.com/dl/fa/document/manual/plc/sh081684eng/sh081684engg.pdf

[25] ODVA, Inc., "CIP on Ethernet Technology," 2021, (accessed January 31, 2023). [Online]. Available: https://www.odva.org/wp-content/uploads/2021/05/PUB00138R7_Tech-Series-EtherNetIP.pdf

[26] Schneider Electric Industries SAS, "Principles of ethernet/ip communication," 2011, (accessed January 31, 2023). [Online]. Available: https://scadahacker.com/library/Documents/ICS_Protocols/Schneider%20-%20Principles%20of%20EtherNetIP%20Communication.pdf

[27] http://www.Modbus-IDA.org, "Modbus application protocol specification v1.1b," 2007, (accessed February 1, 2023). [Online]. Available: https://modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

[28] Time-Sensitive Networking (TSN) Task Group. IEEE 802.1. [Online]. Available: https://1.ieee802.org/tsn/

[29] M. D. Johas Teener, A. N. Fredette, C. Boiger, P. Klein, C. Gunther, D. Olsen, and K. Stanton, "Heterogeneous networks for audio and video: Using ieee 802.1 audio video bridging," *Proceedings of the IEEE*, vol. 101, no. 11, pp. 2339–2354, 2013.

[30] "IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks," *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)*, pp. 1–1993, 2018.

[31] "IEEE Standard for Local and metropolitan area networks– Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams," *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*, pp. 1–72, 2010.

[32] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. Elbakoury, "Performance comparison of ieee 802.1 tsn time aware shaper (tas) and asynchronous traffic shaper (ats)," *IEEE Access*, vol. 7, pp. 44 165–44 181, 2019.

[33] R. Hummen, S. Kehrer, and L. Wüsteney, "White paper – tsn – time sensitive networking," Belden Inc., 2019, (accessed January 26, 2023). [Online]. Available: https://www.belden.com/hubfs/resources/knowledge/white-papers/tsn-time-sensitive-networking.pdf

[34] "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-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015)*, pp. 1–57, 2016.

[35] L. Zhao, P. Pop, and S. Steinhorst, "Quantitative performance comparison of various traffic shapers in time-sensitive networking," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2022. [Online]. Available: https://doi.org/10.1109%2Ftnsm.2022.3180160

[36] "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)*, pp. 1–52, 2016.

[37] PROFIBUS Nutzerorganisation e. V. (PNO), *PROFINET over TSN – Guideline*, 2021 (accessed February 7, 2023). [Online]. Available: https://www.profibus.com/download/profinet-over-tsn

[38] ——, *FAQs – PROFINET over TSN*, 2023 (accessed February 7, 2023). [Online]. Available: https://www.profibus.com/technology/industrie-40/profinet-over-tsn#tab4-221437

[39] CC-Link IE TSN. CC-Link Partner Association. [Online]. Available: https://www.cc-link.org/en/cclink/cclinkie/cclinkie_tsn.html

[40] Growth of CLPA. CC-Link Partner Association. [Online]. Available: https://www.cc-link.org/en/clpa/global/index.html

[41] C. Soh and S. Sia, "The challenges of implementing "vanilla" versions of enterprise systems." *MIS Quarterly Executive*, vol. 4, 01 2005.

[42] About: Vanilla software. DBpedia. [Online]. Available: https://dbpedia.org/page/Vanilla_software

[43] PROFIBUS Nutzerorganisation e. V. (PNO), *PROFINET – Design Guideline*, 2022 (accessed February 10, 2023). [Online]. Available: https://www.profibus.com/download/profinet-installation-guidelines

[44] Raspberry Pi Foundation, *3B+ ethernet: is it LAN75xx or LAN78xx ?*, 2018 (accessed February 10, 2023). [Online]. Available: https://forums.raspberrypi.com/viewtopic.php?t=208399

[45] ——, "Raspberry pi 3 model b+," 2018, (accessed January 1, 2023). [Online]. Available: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf

[46] Microchip Technology Inc., *SuperSpeed USB 3.1 Gen 1 to 10/100/1000 Ethernet Controller*, 2021 (accessed February 10, 2023). [Online]. Available: https://ww1.microchip.com/downloads/aemDocuments/documents/UNG/ProductDocuments/DataSheets/LAN7800-Data-Sheet-DS00001992H.pdf

[47] A. S. Gillis, "Smp (symmetric multiprocessing)," 2022, (accessed January 2, 2023). [Online]. Available: https://www.techtarget.com/searchdatacenter/definition/SMP

[48] M. A. Brown. 6.2. pfifo_fast, the default linux qdisc. [Online]. Available: https://tldp.org/HOWTO/Traffic-Control-HOWTO/classless-qdiscs.html#qs-pfifo_fast

[49] Linux man pages online, *pfifo_fast - three-band first in, first out queue*, 2002 (accessed October 27, 2022). [Online]. Available: https://man7.org/linux/man-pages/man8/pfifo_fast.8.html

[50] ——, *PRIO - Priority qdisc*, 2001 (accessed October 27, 2022). [Online]. Available: https://man7.org/linux/man-pages/man8/tc-prio.8.html

[51] "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)*, pp. 1–65, 2017.

[52] "IEEE Standard for Local and metropolitan area networks–Frame Replication and Elimination for Reliability," *IEEE Std 802.1CB-2017*, pp. 1–102, 2017.

[53] Linux man pages online, *MQPRIO - Multiqueue Priority Qdisc (Offloaded Hardware QOS)*, 2013 (accessed October 27, 2022). [Online]. Available: https://man7.org/linux/man-pages/man8/tc-mqprio.8.html

[54] Linux manual page, "sched – overview of cpu scheduling," 2021, (accessed May 18, 2022). [Online]. Available: https://man7.org/linux/man-pages/man7/sched.7.html

[55] Oracle Corporation and/or its affiliates, *Scheduling*, 2010 (accessed March 3, 2023). [Online]. Available: https://docs.oracle.com/cd/E19455-01/806-5257/mtintro-69291/index.html

[56] Linux manual page, "sched_setattr, sched_getattr - set and get scheduling policy and attributes," 2021, (accessed May 18, 2022). [Online]. Available: https://man7.org/linux/man-pages/man2/sched_setattr.2.html

[57] S. Rostedt, "Using SCHED_DEADLINE – Controlling CPU Bandwidth," 2016, (accessed May 18, 2022). [Online]. Available: https://elinux.org/images/f/fe/Using_SCHED_DEADLINE.pdf

[58] Linux manual page, "cgroups - linux control groups," 2022, (accessed January 12, 2023). [Online]. Available: https://man7.org/linux/man-pages/man7/cgroups.7.html

[59] J. Gschwend and K. Brunner, "TSN Demo – Demonstration of various TSN Technologies," 2021, internal ZHAW documentation, unpublished.

[60] PROFIBUS Nutzerorganisation e.V. (PNO), "Profinet netload robustness for security," 2022, (accessed January 18, 2023). [Online]. Available: https://www.profibus.com/download/profinet-netload-robustness-for-security-guideline-former-\security-level-1-netload

[61] J. Dugan, S. Elliott, B. A. Mah, J. Poskanzer, and K. Prabhu. iPerf - the ultimate speed test tool for TCP, UDP and SCTP. Iperf.fr. [Online]. Available: https://iperf.fr/

[62] www.tp-link.com, "5/8-port 10/100/1000mbpsdesktop switch," 2015, (accessed January 20, 2023). [Online]. Available: https://static.tp-link.com/res/down/doc/TL-SG105-108.pdf