Time-Sensitive Networking's Scheduled Traffic Implementation on IEEE 802.11 COTS Devices

Pablo Gutiérrez Peón*†, Paraskevas Karachatzis*, Wilfried Steiner*, and Elisabeth Uhlemann†
*TTTech Computertechnik AG, Vienna, Austria
†School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden
Email: {pablo.gutierrez-peon, paraskevas.karachatzis, wilfried.steiner}@tttech.com, elisabeth.uhlemann@mdu.se

***Abstract*—Wired real-time network deployments based on time-sensitive networking (TSN) are becoming increasingly popular in several application fields including industrial automation and the automotive area. This is to a great extent due to the scheduled traffic (ST; IEEE 802.1Qbv), a TSN mechanism that precisely triggers transmissions at pre-defined instants from a set of independent hardware queues. Unfortunately, the wireless counterparts based on IEEE 802.11 (Wi-Fi) currently lack standardized or functional solutions that would be able to fulfill strict real-time requirements. This work proposes the use of ST over IEEE 802.11, taking advantage of an existing but overlooked mechanism from commercial off-the-shelf (COTS) equipment. An extensive set of tests, spanning different implementation configurations, traffic patterns, and levels of interference evaluates the proposal in terms of reliability and delivery delays and serves as a proof of concept. The tests also evaluate the much-needed clock synchronization quality of an out-of-the-box software-only implementation of the precision time protocol (PTP) due to its relevance in enabling ST.**

I. Introduction

The local area network (LAN) standards IEEE 802.3 (Ethernet) and IEEE 802.11 (Wi-Fi) are the prevailing technologies to enable data exchange between computer systems in home and office environments. Their unbeatable success, founded on easy-to-configure, fast and low-cost technologies, has not passed unnoticed in other fields where computer networks are present, including industrial automation, the automotive area, or aerospace. However, in contrast to home and office environments, these application fields have a set of requirements in common, namely the need for providing real-time guarantees. More specifically, bounded data delivery delays and high reliability are required and imposed by an environment that feeds the system with sensor inputs and expects a timely response via actuators [1]. Unfortunately, the real-time performance from Ethernet and IEEE 802.11 suffers when data frames coming from various sources interfere while accessing shared resources, particularly when stored in the network device queues before the transmission [2]. Further, collisions at the medium access control (MAC) level are often preventing IEEE 802.11 from being reliable [3].

In the case of IEEE 802.11, many alternatives have been formulated to provide a bounded medium access time and a more reliable MAC. The first efforts from the IEEE standards in such a direction were part of IEEE 802.11e. Despite the improvements, they only mitigated but did not solve the problem as collisions can still occur, and only a reduced number of devices are supported in a network [3]. In the case of Ethernet, several network technologies have tried to enable a real-time service. From PROFINET in the industrial field to AFDX in aerospace, a collection of proprietary field-specific Ethernet extensions were deployed that require special hardware and consequently suffer from limited interoperability.

This scenario has changed with the introduction of the standardized mechanisms proposed by IEEE 802.1 "Time-Sensitive Networking" (TSN), notably due to IEEE 802.1Qbv "Enhancements for Scheduled Traffic" (ST) and IEEE 802.1AS-Rev "Timing and Synchronization for Time-Sensitive Applications" with its clock synchronization mechanism based on the precision time protocol (PTP). Both mechanisms combined allow for bounded-delay medium access on Ethernet, enabled by network-wide, coordinated scheduled transmissions [4]. As a result, TSN has experienced a warm reception, becoming the cutting-edge networking technology to apply in crucial use cases like in-car communication backbones [5] or communications between heterogeneous industrial machines [6].

Although TSN's ST conception has Ethernet switches in mind, the mechanism is not restricted to Ethernet's physical layer (PHY). Applying an ST-like mechanism also in IEEE 802.11 enables the fulfillment of the real-time requirements of the data exchanges by scheduling the medium access. Further, having the ST mechanism supported both over Ethernet and IEEE 802.11 allows for network-wide scheduling of data exchanges facilitating end-to-end real-time guarantees.

This paper presents a proof-of-concept implementation of the use of TSN's ST over IEEE 802.11. The implementation modifies the driver in Linux systems (`ath9k`) that manages the commercial off-the-shelf (COTS) Atheros 802.11n PCI/PCIe chips to enable several built-in hardware mechanisms to work in the same fashion as the ST from IEEE 802.1Qbv. The proposal has the advantage of using already commercially available hardware without requiring any hardware redesign and is in theory applicable to other chip models providing comparable hardware support. The proof-of-concept evaluation is carried out by comparing different MAC configurations with and without ST support. An evaluation of the PTP clock synchronization protocol is also presented based on a software-only implementation.

The remainder of the paper is structured as follows. Section II presents the background on IEEE standards for Ethernet

and IEEE 802.11 and their support for real-time guarantees. Section III reviews related work. Section IV describes how to enable ST over IEEE 802.11. Section V covers the performance evaluation of the implementation. Finally, the conclusion is presented in Section VI.

II. IEEE LOCAL AREA NETWORK STANDARDS AND THEIR REAL-TIME SUPPORT

A. IEEE 802.3 - IEEE standard for Ethernet

The carrier sense multiple access with collision detection (CSMA/CD) protocol regulates the medium access on Ethernet. CSMA/CD checks if the transmission medium is free before attempting a transmission and while it is taking place. If some other signal is detected on the wire, a random backoff waiting time is introduced, making the access time depend on other users sharing the medium. However, most of the current Ethernet deployments use switch-based topologies with the end systems connected via switches and switches that can as well be linked together. Dedicated full-duplex wired connections are placed between each pair of network devices, effectively avoiding interference from other devices at the medium level. Unfortunately, frames still have a chance to interfere in the queues at the switches while waiting to be transmitted. If the traffic pattern generation at the end systems is random, frames might experience varying delays when accumulating at the queues and even get lost if they overflow them [2].

B. Time-Sensitive Networking (TSN)

A successful attempt to overcome the randomness of Ethernet in the framework of IEEE standards has been made through TSN and its offer that includes mechanisms to support guaranteed latencies with low jitter. In TSN, each frame carries a priority in the range of 0-7 that goes into the Ethernet frame header as part of the so-called VLAN tag field (IEEE 802.1Q). After the switching engine redirects the frame to the corresponding port based on its destination, the priority filter uses the frame priority to place frames on 8 separate queues that exist on each port (Figure 1), allowing a priority-based partition of the traffic through the entire TSN network. Each queue is followed by a shaper that controls the flow of frames. Among the several shapers described in the standard, the credit-based shaper (CBS; IEEE 802.1Qav) can evenly distribute the amount of traffic in time, which is beneficial for streaming applications.

Right after the shapers phase, traffic goes through the transmission gates (TGs; IEEE 802.1Qbv). A TG adopts two states: open allows traffic to go through while closed holds it. The state of the TGs over time is defined in the gate control list (GCL). The GCL commands what frames can go through at every instant, enabling ST by adopting a time-division multiple access (TDMA) scheme with time-triggered transmissions. In case more than one of the gates is simultaneously open, frames are selected for transmission according to their priority. It should be highlighted that the standard does not describe how

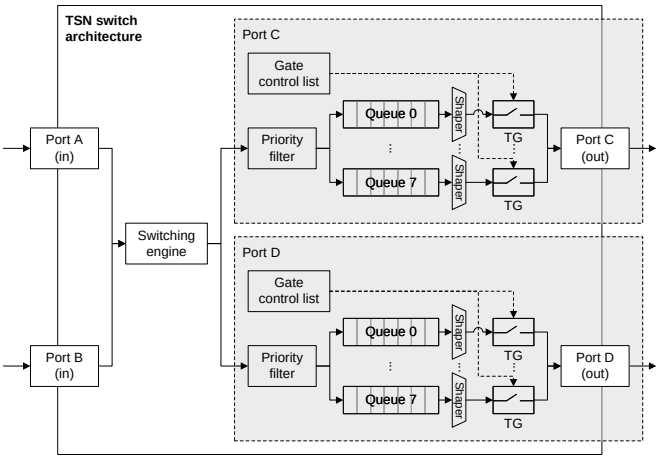

Fig. 1. TSN switch architecture.

to create a schedule, and the implementation is left open, but scheduling solutions are already available [4].

Both the CBS and the TGs mechanisms are meant to be implemented in hardware in pursuance of a better timing performance. Only the TGs are needed for traffic requiring a precise latency and low jitter, while the prior shaping phase can be disregarded [4].

Sharing a common notion of time is essential for the TGs to let traffic go through in such way exchanges are coordinated across a set of independent devices. In IEEE 802.1AS-Rev, a single node, the grand master, is selected either manually or as an outcome of the so-called best-master protocol. The grand master serves as the reference to which other nodes, the slaves, synchronize their clocks. The value of the grand master clock is sent periodically to the slaves as part of a `Sync` message. Depending on the implementation, an accurate value of the sending time of the `Sync` message might be included as part of the message itself (one-step) or sent in a `Follow_Up` message (two-step). Having received the sending time, slaves can calculate their offset from the grand master and correct their clocks accordingly. The impact of the propagation delay is also considered in the correction and based on a per-link measurement procedure realized with a sequence of message exchanges between the two nodes connected to each link.

Network devices implementing IEEE 802.1AS-Rev should comply with the requirement to have a maximum offset between clocks of 1 µs for up to 7 links. To reach such precision, hardware support is highly recommended. These numbers are easily achievable in wired settings. In contrast, the results in Section V will show that a less stable outcome can be observed over wireless links with a software-only implementation, limited due to the lack of hardware timestamping. However, it is possible to deal with the impediments of software timestamping and achieve an acceptable performance [7].

C. IEEE 802.11 - The baseline MAC

The baseline MAC in IEEE 802.11, termed distributed coordination function (DCF), is founded on the carrier sense multiple access with collision avoidance (CSMA/CA) protocol. CSMA/CA is a variant of CSMA/CD constrained by the current limitation of off-the-shelves wireless radios to perform CD due to their inability to transmit and receive simultaneously on the same frequency. In CSMA/CA, the medium is sensed for some duration, the DCF interframe space (DIFS), seeking other transmissions and deferring access if detected. If several devices sense the same transmission and defer access, they could collide once the ongoing transmission is over. In order to avoid this, an extra random time in the range of a contention window (CW) is added. In DCF, acknowledgment frames (`ACK`) give feedback to the data sender whether a transmission succeeded. If the `ACK` is not received, a retransmission could be triggered.

An additional MAC mechanism in the IEEE 802.11 standard is the point coordination function (PCF), which is applicable when end systems, referred to as stations by the standard, are connected through an access point that takes the coordinator role. In PCF, the medium access is divided into periodic phases having two parts: one contention-free using the PCF mechanism and one contention-based using DCF. The PCF mechanism gives a station the right to transmit after receiving a polling frame (`CF-Poll`) coming from a coordinator node. The PCF mechanism forces a sensing medium wait of PCF interframe space (PIFS), which is shorter than DIFS, thus prioritizing frames using PCF over DCF. With PCF, delay-bounded channel access is achievable, but it is not mandatory in the standard and the mechanism has some scalability limitations [3].

D. IEEE 802.11e - MAC quality of service enhancements

The standard amendment IEEE 802.11e aims to provide quality of service enhancements targeting delay-critical applications by defining two new coordination functions: enhanced distributed channel access (EDCA) and HCF controlled channel access (HCCA).

In EDCA, frames are assigned a priority carried as part of the quality of service (QoS) field in the frame header. Such priorities are tied the so-called access categories (AC), which translate to different values for the CW and the IFS, and different queues where to store the frames before transmission. The IFS is renamed arbitration inter-frame space (AIFS) in the standard amendment. Still, the default values for the CW make the range of possible sensing times between different priorities overlap, which might cause frames from lower priority ACs to get access before higher priority ones even if they start to arbitrate simultaneously [8].

The HCCA mechanism is similar to PCF but allows for the initiation of contention-free phases inside the contention period. For both EDCA and HCCA, once access to the medium is provided, it is possible to perform transmissions for a duration of a transmission opportunity (TXOP). However, its polling mechanism results in a significant communication overhead when transmitting frames with a small payload [9]. Further, currently COTS devices do not implement HCCA, and it is unclear if and when the industry will adopt HCCA in the future.

III. Related work

Several academic works have proposed and implemented real-time wireless solutions based on IEEE 802.11. Most of them suggest TDMA schemes to organize the medium access. The access is often carried out in cycles, with some time slots precisely assigned and some accessed through CSMA/CA. The actual schedule of the slots differs from using EDF [10], round-robin [11] or time-triggered [12][13][14][15]. The work in [16] provides a framework where different scheduling policies could be applied and relies, like this paper, on one-shot mechanisms from the hardware. The synchronization of the nodes to the network coordinator is done with a transmission triggered from the coordinator before each data transmission [10] with the consequent overhead, using the IEEE 802.11 beacon frame [13][14] or through specific synchronization messages [11][12][15]. The schedule for the next cycle might be included as part of the beacon or sent in a specific frame. Several of the analyzed works present implementations using modified versions of the `ath9k` driver, with numbers being quite promising in terms of reliability and jitter, especially when compared to standard DCF, yet highly dependent on the interference scenario. The work in [10] shows that results over 99% of frames delivered on time can be achieved if including planned retransmissions. In [11], improvements concerning throughput and timeliness are provided over comparable factory automation use cases. The proposal from [13], named RT-WiFi, brings major improvements in terms of frame delivery latency compared to DCF after removing retransmissions but at the price of losing between 7% to 10% of the frames in office interference scenarios. An implementation of RT-WiFi adopting a software-defined radio (SDR) approach instead of COTS hardware is presented in [17]. The SDR option offers a hardware alternative in such cases where an open source driver controlling the hardware does not exist or parts of the required hardware functionality are not openly available. The authors in [16] use the built-in one-shot mechanism from Atheros chipsets to achieve 99% of frames delivered with low jitter, similar to the numbers in [14], with 95% of the frames delivered on time and µs range jitter in scenarios with moderate interference. In contrast, the work in [15] controls the transmissions without specific hardware support, incurring larger overheads.

Based on the adoption of TSN in wired deployments, recent years have shown a progression towards wireless solutions based on TSN, as opted for in this paper. The contribution from [18] leaves out COTS equipment and decides on a new FPGA-based hardware design with the proposal of a new PHY. The customized hardware exhibits promising results in terms of delivery latency, but for now, it is not compliant with TSN or IEEE 802.11. In [19], the behavior of ST is emulated and some preliminary results are provided, focused

on showing the performance degradation in terms of delays when compared to the use of ST on the wired network and the penalties introduced by the lack of a more precise control of the hardware. Finally, the authors in [20] present an implementation of ST over IEEE 802.11. Their results are mainly tackled to describe the impact of enabling ST on the behavior of a particular use case dealing with robots, lacking a comprehensive analysis of the performance in terms of reliability and delays.

IV. Enabling scheduled traffic in IEEE 802.11

The support for ST is enabled on the Atheros AR928X chipset via modifications on the Linux device driver that is in charge of its management.

A. Linux IEEE 802.11 support and clock synchronization

The operation and control of the hardware in Linux are done by device drivers in the form of kernel modules. The modules are dynamically loaded into the kernel at runtime when needed, typically when the hardware associated with the driver is detected. Despite the great variety of hardware options, operating systems like Linux aim to offer unified interfaces to the applications that hide the details of the devices behind standardized calls. This unified interfacing translates in Linux into three main types of device drivers classified according to the way they transfer data: character drivers, accessed as a stream of bytes in a file; block drivers, with data being transferred in memory blocks; and network drivers, with the transfer happening via sockets and formatted as network packets, i.e., an operating system's typical name for frames.

Once a network device driver is loaded into the kernel, it provides a network interface to exchange frames and additional administrative features like assigning network addresses, setting transmission parameters, or keeping statistics. Due to the particularities of wireless transmissions, the IEEE 802.11's MAC layer in Linux is handled by the Media Access Control Sublayer Management Entity (MLME). MLME tries to tackle the varying reliability, security issues, and power constraints in wireless networks by defining a set of operations for managing the networks' scanning, joining, association and re-association, power management, and time synchronization. Depending on the hardware architecture, MLME can be completely done in hardware (full MAC) or software (soft MAC). The latter has the advantage of simplifying hardware and providing more flexibility. Two kernel modules, `mac80211` and `cfg80211`, take the MLME responsibility for all IEEE 802.11-based soft MAC network devices in Linux (Figure 2). `Mac80211` also receives frames from and to the socket applications and the hardware-specific modules. Other important responsibilities are building the IEEE 802.11 frame header, including the QoS field, updating statistics, segmenting the frames if they are larger than what IEEE 802.11 allows, and enqueueing them if the hardware queues are not available. Under the Linux MLME, there are hardware-specific modules, which are addressed in Section IV-B.

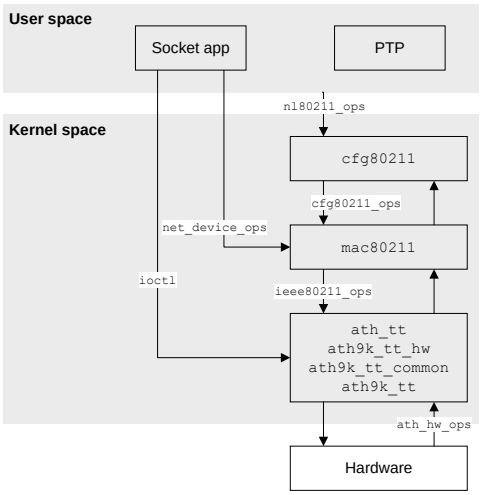

Fig. 2. Architecture and interfaces of the Linux IEEE 802.11 MAC and the modified `ath9k` driver modules.

Along with the MAC driver, the option chosen in this paper to synchronize the Linux end systems is the PTP software `ptpd` that runs as a user-space application. PTP is part of IEEE 1588, a superset of the IEEE 802.1AS-Rev.

B. Enabling scheduled traffic on an IEEE 802.11 chipset

The implementation relies on the Atheros AR928X chipset, an IEEE 802.11n compliant solution in the 2.4 GHz band, reaching up to 150 Mbps throughput. The chipset covers the PHY, and MAC layers, both accessible for configuration and status reports from the host device by a set of registers. For such purposes, the `ath9k` driver was used, given that its code is available as open source. The modifications made in the implementation described here are related to the MAC layer and thus independent of the role of stations and access points. An implementation of the proposal over other hardware models is in theory possible, subject to the existence of a mechanism to control the exact time when to trigger the transmission of data frames.

The chipset MAC offers ten transmission queues (Figure 3). Mapping to the queues is configurable, but by default, data frames go into queues Q0 to Q3 based on their EDCA priority specified as part of the frame header. Queues Q4 to Q7 are left unused but can as well be dedicated to data frames if needed so that the whole range of EDCA priorities is covered. Lastly, queues Q8 and Q9 are reserved for beacon frames. For frames transmitted over both the IEEE 802.11 and Ethernet networks, the EDCA priority value is often selected to match the priority from the VLAN tag. The partition of traffic into different queues based on their priority effectively prevents traffic of higher criticality to interfere with traffic of lower criticality in the device memory. Each transmission queue comprises two modules that manage the frame sending together: the queue control unit (QCU), which selects the frames to be transmitted based on the so-called QCU frame scheduling policy, and the distributed coordination function unit (DCU), which carries out the EDCA channel access.

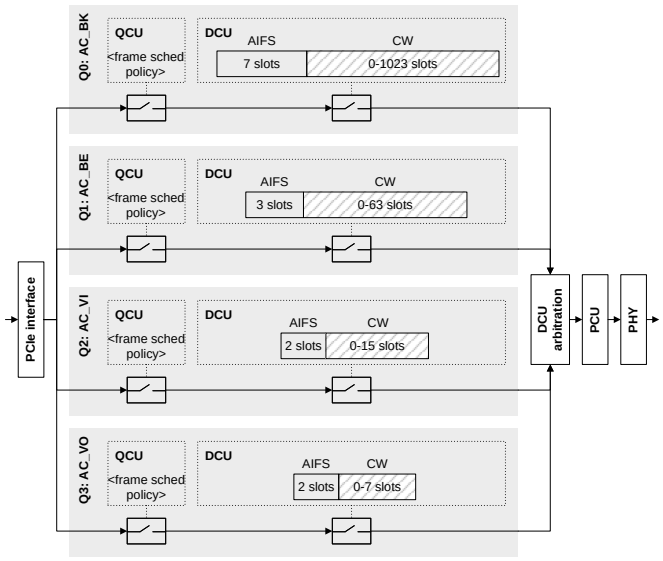

Fig. 3. Chipset MAC queues. Only the queues Q0-Q3 are shown, corresponding to the default chipset mapping between ACs and EDCA priorities.

The interface between the chipset and the host is based on PCIe. The interface serves to access the registers and perform the frame exchange. The first version of the driver was developed over a USB-accessed chipset, but the delay introduced into tasks like writing registers was in the order of ms, while the PCIe interface offers delays in the order of µs. After the frame is sent over the PCIe interface, it is stored in the chipset in a buffer in the form of a FIFO linked list. The responsibility of the corresponding QCU is to select the frames from the linked list and make them available to the DCU to proceed with the transfer.

The decision on when a frame is selected for transmission depends on the QCU frame scheduling policy. Except for beacon-dedicated queues that trigger the beacon frame periodically, all other queues apply an as-soon-as-possible policy by default, with frames sent to the DCU as soon as no other frame is in the DCU. Among the rest of the available policies, the one that best resembles a time-triggered scheduling as enabled by the TGs is CBR. CBR allows the selection of a frame for transmission periodically, but unfortunately, it does not fully fit the purpose since the TGs mechanism is not restricted to periodic transmissions only. One option could be to change the value of the CBR period each time a frame has to be sent out, but testing showed that the hardware did not cope well with the dynamic change of the period, and frames were not dispatched on time. Luckily, the chipset offers a solution that enables choosing the specific moment when a queue is allowed to transmit data. The solution relies on the one-shot mechanism (Figure 4), used in combination with the CBR QCU frame scheduling policy. First, the user shall write which QCUs should select frames for transmission. Then, the one-shot mechanism is used to trigger a frame transfer from each active QCU to the corresponding DCU.

Further, the chipset QCU frame scheduling policies allow

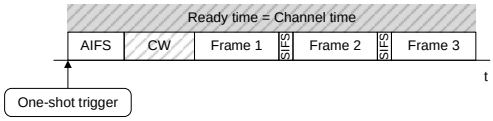

Fig. 4. Example of the transmission of three frames and the use of one-shot, ready-time, and channel-time mechanisms to access the channel.

the transmission of one frame at a time instead of a set of them, like in the TGs or TXOP mechanisms. This behavior can be amended using the ready-time mechanism from the chipset. Ready time specifies the duration during which the QCU marks the frames as ready to be sent to the DCU, emulating the effect of the TG being open for a user-defined duration.

Once the QCU makes the frame available to the DCU, the EDCA channel access protocol is followed. The CW values can be adjusted accordingly to avoid the problem of overlapping priorities with the default AC parameters. In this regard, the channel-time mechanism from the chipset requires only the first frame in some user-defined duration to arbitrate for channel access. The rest of the frames would avoid waiting for AIFS and CW, leaving just the SIFS in between (Figure 4). The lack of continuous arbitration efficiently reduces the overhead of the DCU channel access for each frame. Furthermore, the duration can be set to be equal to the ready time, allowing the scheduler to put as many frames as possible inside the ready-time duration. After the DCU grants the medium access to a frame, it is handed over to the protocol control unit (PCU), which is responsible for sending the frame to the baseband logic and performing other DCF-compliant tasks.

The kernel modules `ath_tt`, `ath9k_tt_hw`, `ath9k_tt_common` and `ath9k_tt` from Figure 2 are the modified version in this paper of the `ath9k` driver modules provided in the Linux kernel. These modules are in charge of setting the proper configuration values for the described mechanisms and performing the transfer of frames from the `mac80211` module to the chipset and vice versa over the PCIe interface. To enable the TGs mechanism, a schedule shall be configured in the aforementioned kernel modules. The schedule is made available to the kernel modules by exposing them as character drivers, allowing a direct path to user applications via `ioctl` calls. Currently, the distribution of the schedule configuration is done prior to the network operation. However, the driver can load a new schedule dynamically, so the configuration mechanisms from the TSN's IEEE 802.1Qcc "Stream Reservation Protocol (SRP) Enhancements and Performance Improvement" could be used in the future. Having a schedule available, the times when to open and close the gates are programmed using Linux kernel timers. When the timers expire, their handling routines are programmed to reconfigure the one-shot, ready-time, and channel-time mechanisms.

V. Proof-of-concept evaluation

A. Evaluation description

The purpose of the evaluation is to assess the concept of applying ST over IEEE 802.11. A comparison of the performance between the MAC using ST and DCF is made. Additionally, the impact on the performance of two other MAC configuration options is evaluated: default vs. prioritized channel access and active vs. inactive channel time. The considered performance measures are reliability, delay between packet arrivals, and clock synchronization precision. Reliability is calculated as the portion of packets arriving at the destination from those sent. Since there are two reasons why packets might not be delivered, a distinction is made. On the one hand, packets that are not transmitted due to the hardware being overflown by too many transmission requests. On the other hand, packets that are transmitted but not received due to channel conditions. Furthermore, the time between data arrivals and, more interestingly, the jitter for such measure indicates the uncertainty regarding the timing of packets, being low jitter the desired outcome. Finally, it is relevant to assess the quality of the clock synchronization, which is required by ST to operate and useful for applications relying on a shared time-base between nodes.

The Atheros AR928X IEEE 802.11 chipset used in the implementation is installed via PCIe interface on five industrial PCs model MFN-100 running a non-real-time version of Linux to serve as a proof of concept for the use of industrial equipment. The five nodes, attached in an infrastructure topology (Figure 5) are placed evenly over an area of 150 m^2 in an office building. Three additional Raspberry Pi nodes are used to generate interference, of which two are senders and one is receiver.

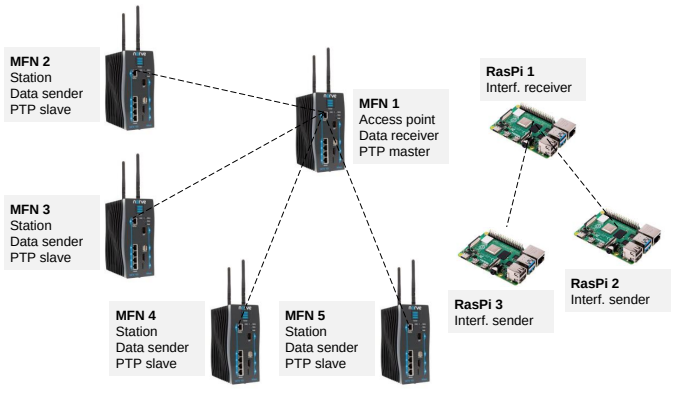

Fig. 5. Evaluation scenario.

The performance of the implementation is evaluated by comparing 72 scenarios resulting from the combination of different protocol-related, traffic-related, and fault-injection-related parameters.

The protocol-related parameters, summarized in Table I include the MAC protocol used, either DCF or ST, also whether default or prioritized channel access parameters are used and finally if channel time is active. Table I also describes the mapping of different kinds of transmissions into priorities. The data generated to be exchanged are assigned a medium priority, whereas the network services and clock synchronization data are assigned a low priority. The low-priority assignment is not generally considered a drawback for protocols like PTP, since they are often transmitted without any kind of prioritization. Also, the wireless network broadcasts beacon frames periodically. These messages are transmitted on the highest priority and will have some impact on the data frames. This interfering effect has not been thoughtfully studied and tackled and is left as future work. Retransmissions via automatic repeat request (ARQ) have also been disabled, giving every packet just one chance to be delivered. The nodes are synchronized using `ptpd`, an implementation of PTP for Linux, with one node statically selected as the PTP master. `Ptpd` runs as a user-space application on the nodes and lacks hardware timestamping support in this evaluation. Further, timestamping is performed in a two-step process. The evaluation of the clock synchronization provides an idea of how good the synchronization can be under such non-ideal conditions and whether it can be used to enable ST.

TABLE I
Protocol-related evaluation parameters.

Parameter	Value
MAC protocol	DCF
	ST
Channel access:	Default: 2, [0-1023]
AIFS [slots], CW [slots]	Prioritized: 0, 0
Channel time (ST only)	Inactive
	Active and equal to the duration of the current ST slot
Priority mapping	Beacon frame (high priority), generated data (medium priority), network services and clock synchronization (low priority).
ST schedule	10 ms long, with two segments triggering medium- and low-priority data respectively, and a 1 ms buffer between the two.
Acknowledgement frames (`ACK`)	Enabled
Automatic repeat request (ARQ)	Disabled
Fragmentation	Disabled
Hidden node handling	RTS/CTS disabled
Rate adaptation	Enabled
Maximum rate [Mbps]	54 (IEEE 802.11g)
Clock synchronization protocol	PTP (`ptpd` software)
Clock synchronization timestamping	Software-only two-step process
Beacon frame period [ms]	100
Wi-Fi security options	WPA

In the case of traffic-related evaluation parameters, summarized in Table II, these cover the evaluation of different packet sizes classified into small, medium and large, and different levels of utilization of the available bandwidth, divided into

low and mid-high utilization and resulting on different burst sizes dependant on the packet size. At least 10000 data packets are generated on each test. Other parameters are applicable to all scenarios and include the generation of data randomly over the whole period and using non-blocking socket calls in Linux, which makes the application drop the data in case transmission requests flood the underlying hardware.

TABLE II
TRAFFIC-RELATED EVALUATION PARAMETERS.

Parameter	Value
Packet size [B] Estimated utilization [%] (burst size)	Small: 32 Low: 1.1% (1x)
	Small: 32 Mid-high: 28.6% (26x)
	Medium: 512 Low: 1.8% (1x)
	Medium: 512 Mid-high: 29.1% (16x)
	Large: 2048 Low: 4.1% (1x)
	Large: 2048 Mid-high: 28.4% (7x)
Number of generated packets	$\geq$ 10000
Packet generation period [ms]	50
Generation pattern	Random over the period
Addressing	Unicast, with one hop to the access point
Linux socket calls	Non-blocking

Finally, the fault-injection evaluation parameters, summarized in Table III, indicate the type of interference that has been present during the evaluation, either the default office background noise that exists in the experimentation environment or the office noise complemented with the additional disturbance generated from a couple of Raspberry Pis.

TABLE III
FAULT-INJECTION-RELATED EVALUATION PARAMETERS.

Parameter	Value
Type of interference	Office noise
	Office noise + additional disturbance
Additional disturbance characteristics	Data size [B]: 256 Burst size: 20x Period [ms]: 100 Number of generating nodes: 2x Estimated utilization: 5.7%

B. Results

Table IV depicts the reliability results for each protocol, traffic and fault-injection parameter, obtained after aggregating the results from all the scenarios where the given parameter is present. The results are also provided after considering only the parameter options that lead to the best and worst scenarios, which will be detailed in the analysis below. The proportion of delivered packets (del.) is provided for each. Then a distinction is made between the packets dropped due to overload before attempting the transmission (drp.) and those sent but interfered (int.).

When considering all the scenarios, the ST-based MAC protocol outperforms DCF, having default parameters for channel access is generally better than using prioritized ones, and activating the channel time feature is more beneficial than having it deactivated. Further, scenarios with a high-medium utilization tend to exhibit worse performance due to a more congested medium, which is also applicable to scenarios where an additional disturbance is present. The analysis of the results shows that the best scenarios are those where the concurrence of DCF as MAC protocol, prioritized channel access, and a mid-high medium utilization does not occur. Under those beneficial conditions, values over 0.95 reliability are easily achievable, and DCF and ST provide similar results. In turn, the set of scenarios providing the worst results, the complement of those scenarios in the best set, counts with reliability values lower than 0.4 and as low as 0.1. A few conclusions can be inferred from the reliability results. First, the MAC protocols DCF and ST do not provide diverging results unless other parameters like prioritized channel access are considered, which harms DCF and should not be used in combination. The reason is that several prioritized nodes could try to access the medium at the same time in DCF, which results in collisions. Second, results also confirm the obvious, that a more congested medium, either due to a mid-high utilization from the traffic generated in the network or due to the additional disturbance, decreases reliability. Thus, it is essential to highlight that scheduling transmissions, as in ST, does not translate per se to better reliability. A large part of the reliability improvement comes from admission control mechanisms. By default, DCF is not used in combination with admission control and the nodes are not aware of the medium usage quota they shall not exceed to avoid overflowing the channel capacity. Such DCF nodes try to send data as soon as they have some data to exchange available, which could compromise the reliability of the transmissions. Finally, the proportion of packets dropped and interfered increases with mid-high utilization and additional interference. Such a scenario causes the wireless channel to be congested, translating the issue to the network interface, which is not able to accept more sending requests from the application, dropping the packets before they get a chance to be sent.

Table V summarizes the results for the delay between packet arrivals. The results are again provided for each configuration parameter after aggregating all the scenarios where the given parameter is present. The results for the parameters that result in the best and worst scenarios, which will be clarified in the analysis that follows, are also included. In all cases, the packet size and utilization parameters are excluded due to the lack of interest, e.g., bursty scenarios will naturally have a shorter delay between arrivals than scenarios where packets are individually sent. Hence, the interest is in evaluating if the

TABLE IV
RELIABILITY RESULTS.

Parameter		Results all scenarios			Results best scenarios			Results worst scenarios		
		Del.	**Drp.**	**Int.**	**Del.**	**Drp.**	**Int.**	**Del.**	**Drp.**	**Int.**
MAC protocol	DCF	0.7868	0.0673	0.1459	0.9763	0.0137	0.0100	0.2185	0.2281	0.5534
	ST	0.9582	0.0025	0.0393	0.9582	0.0025	0.0393	-	-	-
Channel access	Default	0.9347	0.0102	0.0551	0.9347	0.0102	0.0551	-	-	-
	Prioritized	0.8675	0.0380	0.0945	0.9973	0.0000	0.0027	0.2185	0.2281	0.5534
Channel time	Inactive	0.8630	0.0357	0.1013	0.9550	0.0082	0.0367	0.2185	0.2281	0.5534
	Active (ST only)	0.9774	0.0009	0.0218	0.9774	0.0009	0.0218	-	-	-
Packet size, utilization	Small, low	0.9983	0.0000	0.0017	0.9983	0.0000	0.0017	-	-	-
	Small, mid-high	0.8431	0.0318	0.1251	0.9395	0.0056	0.0549	0.3612	0.1632	0.4757
	Medium, low	0.9984	0.0000	0.0016	0.9984	0.0000	0.0016	-	-	-
	Medium, mid-high	0.8247	0.0341	0.1412	0.9511	0.0025	0.0464	0.1923	0.1923	0.6154
	Large, low	0.9980	0.0000	0.0020	0.9980	0.0000	0.0020	-	-	-
	Large, mid-high	0.7442	0.0785	0.1773	0.8726	0.0285	0.0989	0.1020	0.3287	0.5692
Type of interference	Office noise	0.9223	0.0265	0.0513	0.9874	0.0074	0.0052	0.2059	0.2359	0.5582
	Office noise + additional disturbance	0.8799	0.0217	0.0984	0.9389	0.0036	0.0574	0.2311	0.2202	0.5487

rest of the parameters make the delay between arrivals differ much between packet instances or not.

One of the desired and confirmed outcomes of having the ST MAC protocol is having fewer deviations in the delay between packet arrivals, i.e., jitter, than with DCF. Similarly, an active channel time mechanism reduces the time variance between arrivals due to removing the need to sense the channel for the subsequent packets in a burst. Surprisingly, prioritized channel access does not deliver a minor variance between packet arrivals. This behavior is likely due to the higher amount of packets lost when combining DCF and prioritized channel access. No significant differences are seen due to the different types of interference. The results show that the scenarios where the combination of DCF and prioritized channel access does not occur provide the best outcome. The difference between the best and the worst results is of one order of magnitude. Further, the use of ST is expected to benefit the delivery latency in the scenarios where the applications generating and consuming the data are synchronized with the ST schedule. In such cases, it might be possible to align the data generation, transmission and consumption steps so that the end-to-end latencies are minimized.

Finally, Table VI presents the clock synchronization quality results for the packet size and utilization, and type of interference parameters, given these two parameter groups are the most influential for the clock synchronization. The outcome is provided after aggregating the results from all the scenarios where the given parameter is present and also filtering to have the parameters, which will be clarified below, that lead to the best and worst scenarios. The aim is to have a synchronization offset of at most ± 1 ms to be able to follow the ST schedules.

The mean of the offset values $\overline{X}$ is provided to show that the offset variations are happening around zero, as expected. The standard deviation around that mean value is kept lower than 1 ms for low utilizations and the office-only interference,

TABLE V
DELAY BETWEEN PACKET ARRIVALS RESULTS.

Parameter		σ[s] results scenarios		
		All	**Best**	**Worst**
MAC protocol	DCF	0.0790	0.0196	0.1383
	ST	0.0189	0.0189	-
Channel access	Default	0.0200	0.0200	-
	Prioritized	0.0579	0.0176	0.1383
Channel time	Inactive	0.0491	0.0194	0.1383
	Active (ST)	0.0186	0.0186	-
Type of interference	Office noise	0.0382	0.0187	0.1362
	Office noise + add.	0.0396	0.0195	0.1405

showing that a higher utilization and more interference greatly impact the clock synchronization. For the best configurations, those leaving out the higher congestion medium, the nodes still go out of sync at least 10% of the time, which shows how vulnerable a software-based wireless synchronization based on PTP could be. However, even though the nodes might get out of sync for some time, this does not seem to hinder the ability of ST schedules to be followed and obtain better results concerning reliability and delay jitter than DCF as shown in Table IV. Thus, a software-based-only clock synchronization protocol could be good enough in systems without tight timing requirements.

VI. CONCLUSION

The increasing adoption of TSN mechanisms like ST in wired real-time deployments based on Ethernet and the search for a reliable and real-time version of the wireless IEEE 802.11 technology makes ST over IEEE 802.11 a sensible move. The implementation presented in this paper relies on existing hardware mechanisms from IEEE 802.11 COTS, configured via a custom-based Linux driver version, and is in theory

TABLE VI
CLOCK SYNCHRONIZATION QUALITY RESULTS.

Parameter		Results all scenarios			Results best scenarios			Results worst scenarios		
		$\overline{X}$ [s]	σ [s]	In sync	$\overline{X}$ [s]	σ [s]	In sync	$\overline{X}$ [s]	σ [s]	In sync
Packet size, utilization	Small, low	0.0000	0.0006	0.9094	0.0000	0.0006	0.9094	-	-	-
	Small, mid-high	0.0000	0.0014	0.7483	0.0003	0.0013	0.7841	-0.0003	0.0016	0.7126
	Medium, low	0.0001	0.0007	0.9005	0.0001	0.0007	0.9005	-	-	-
	Medium, mid-high	-0.0001	0.0014	0.8007	-0.0002	0.0009	0.9065	-0.0001	0.0019	0.6948
	Large, low	0.0000	0.0008	0.8617	0.0000	0.0008	0.8617	-	-	-
	Large, mid-high	-0.0001	0.0018	0.7988	0.0000	0.0013	0.8874	-0.0003	0.0023	0.7103
Type of interference	Office noise	0.0001	0.0008	0.9103	0.0001	0.0008	0.9103	-	-	-
	Office noise + additional disturbance	-0.0001	0.0014	0.7629	0.0000	0.0009	0.8199	-0.0002	0.0019	0.7059

applicable to other models having a similar hardware support. Different MAC protocol configurations have been tested against different traffic patterns and levels of interference. Results show that reliability can dramatically be affected by the protocol configuration, but overall the introduction of ST does not directly translate to an improvement in reliability when compared to DCF. In fact, it is the admission control mechanism, often associated with the use of scheduling mechanisms like ST, that makes the difference in avoiding a congested medium that could lead to poor reliability. Results also show an improvement with ST compared to DCF in terms of delivery jitter, which has a positive impact on real-time applications having delivery jitter requirements. ST can as well help in minimizing the end-to-end latency between senders and receivers if these are synchronized and coordinated with the ST schedule. Finally, a software-only version of PTP has been used to synchronize the nodes and enable a consistent view of the ST schedules on them. The results on clock synchronization quality show the limitations of not having hardware timestamping and the high sensibility and subsequent performance decrease under a congested medium. Yet, the ST mechanism proves to be functioning even in the presence of phases when synchronization is lost.

ACKNOWLEDGEMENTS

This work has been supported and funded by the Austrian Research Promotion Agency (FFG) via the Austrian Competence Center for Digital Production (CDP) under the contract number 881843.

REFERENCES

[1] E. A. Lee, "Cyber Physical Systems: Design Challenges," in *Proc. IEEE ISORC*, Orlando, USA, 2008, pp. 363–369.

[2] R. Marau, L. Almeida, and P. Pedreiras, "Enhancing Real-Time Communication over COTS Ethernet switches," in *Proc. IEEE WFCS*, vol. 6, Torino, Italy, 2006, pp. 295–302.

[3] R. Costa, P. Portugal, F. Vasques, C. Montez, and R. Moraes, "Limitations of the IEEE 802.11 DCF, PCF, EDCA and HCCA to handle real-time traffic," in *Proc. IEEE INDIN*, Cambridge, United Kingdom, 2015.

[4] S. S. Craciunas, R. Serna Oliver, M. Chmelík, and W. Steiner, "Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks," in *Proc. RTNS*, Brest, France, 2016.

[5] M. Ashjaei, L. Lo Bello, M. Daneshtalab, G. Patti, S. Saponara, and S. Mubeen, "Time-Sensitive Networking in automotive embedded systems: State of the art and research opportunities," *Journal of Systems Architecture*, vol. 117, p. 102137, 2021.

[6] D. Bruckner, M.-P. Stănică, R. Blair, S. Schriegel, S. Kehrer, M. Seewald, and T. Sauter, "An Introduction to OPC UA TSN for Industrial Communication Systems," *Proc. IEEE*, vol. 107, no. 6, pp. 1121–1131, 2019.

[7] A. Mahmood, R. Exel, and T. Sauter, "Delay and Jitter Characterization for Software-Based Clock Synchronization Over WLAN Using PTP," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1198–1206, 2014.

[8] J. R. Betiol Junior, J. Lau, L. de Oliveira Rech, A. Schiaffino Morales, and R. Moraes, "Experimental Evaluation of the Coexistence of IEEE 802.11 EDCA and DCF Mechanisms," in *Proc. IEEE ISCC*, Natal, Brazil, 2018, pp. 847–852.

[9] C. Casetti, C. F. Chiasserini, M. Fiore, and M. Garetto, "Notes on the Inefficiency of 802.11e HCCA," in *Proc. IEEE VTC-Fall*, vol. 4, Dallas, USA, 2005, pp. 2513–2517.

[10] L. Seno, G. Cena, S. Scanzio, A. Valenzano, and C. Zunino, "Enhancing Communication Determinism in Wi-Fi Networks for Soft Real-Time Industrial Applications," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 866–876, 2016.

[11] D. K. Lam, K. Yamaguchi, Y. Shinozaki, S. Morita, Y. Nagao, M. Kurosaki, and H. Ochi, "A Fast Industrial WLAN Protocol and its MAC Implementation for Factory Communication Systems," in *Proc. IEEE ETFA*, Luxemburg, 2015.

[12] P. Bartolomeu, J. Fonseca, and F. Vasques, "Implementing the Wireless FTT Protocol: A Feasibility Analysis," in *Proc. IEEE ETFA*, Bilbao, Spain, 2010.

[13] Y.-H. Wei, Q. Leng, S. Han, A. K. Mok, W. Zhang, and M. Tomizuka, "RT-WiFi: Real-Time High-Speed Communication Protocol for Wireless Cyber-Physical Control Applications," in *Proc. IEEE RTSS*, Vancouver, Canada, 2013.

[14] H. Trsek, S. Schwalowsky, B. Czybik, and J. Jasperneite, "Implementation of an advanced IEEE 802.11 WLAN AP for real-time wireless communications," in *Proc. IEEE ETFA*, Toulouse, France, 2011.

[15] C. Lusty, V. Estivill-Castro, and R. Hexel, "TTWiFi: Time-Triggered Communication over WiFi," in *Proc. ACM DIVANet*, Alicante, Spain, 2021.

[16] G. Cena, S. Scanzio, and A. Valenzano, "SDMAC: A Software-Defined MAC for Wi-Fi to Ease Implementation of Soft Real-time Applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3143–3154, 2018.

[17] Z. Yun, P. Wu, S. Zhou, A. K. Mok, M. Nixon, and S. Han, "RT-WiFi on Software-Defined Radio: Design and Implementation," in *Proc. IEEE RTAS*, Milan, Italy, 2022, pp. 254–266.

[18] Ó. Seijo, J. A. López-Fernández, and I. Val, "w-SHARP: Implementation of a High-Performance Wireless Time-Sensitive Network for Low Latency and Ultra-Low Cycle Time Industrial Applications," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3651–3662, 2020.

[19] B. Schneider, R. C. Sofia, and M. Kovatsch, "A Proposal for Time-Aware Scheduling in Wireless Industrial IoT Environments," in *Proc. IEEE/IFIP NOMS*, Budapest, Hungary, 2022.

[20] S. Sudhakaran, V. Mageshkumar, A. Baxi, and D. Cavalcanti, "Enabling QoS for Collaborative Robotics Applications with Wireless TSN," in *Proc. IEEE ICC*, Montreal, Canada, 2021.