Scheduling for Time-Critical Applications Utilizing TCP in Software-Based 802.1Qbv Wireless TSN

Richard Candell*, Karl Montgomery*, Mohamed Kashef (Hany)*,
Susruth Sudhakaran†, and Dave Cavalcanti †
**Smart Connected Systems Division, National Institute of Standards and Technology (NIST)*, Gaithersburg, MD, USA
†*Intel Labs, Intel Corporation* Hillsboro, OR, USA
Emails: {richard.candell, karl.montgomery, mohamed.kashef}@nist.gov,
{susruth.sudhakaran, dave.cavalcanti}@intel.com

***Abstract*—Time-sensitive networking (TSN) is emerging as a viable means to achieving deterministic quality of service (QoS) communications within mission critical industrial applications such as feedback control systems, robotics, and precision sensing and actuation. Originally developed for Ethernet-based audio-video applications using the User Datagram Protocol (UDP). TSN assumes unidirectional flows from source (talker) to destination(listener) and it is most easily implemented using UDP in which packets are transmitted without an acknowledgment from the recipient. However, most existing industrial protocols are implemented using the reliable Transport Control Protocol (TCP) in which each transmission is explicitly acknowledged. In this work, a bandwidth efficient TSN schedule is developed to accommodate the TCP traffic flow between two synchronized robots collaboratively moving an object. We then demonstrate an IEEE 802.1Qbv TSN schedule over an IEEE 802.11 wireless medium that guarantees robot performance requirements are maintained while accommodating concurrent best-effort traffic flows. The process for schedule selection and experimental data collection is discussed, and TSN configuration parameter tuning and experimental results are provided.**

***Index Terms*—Wireless, TSN, factory communications, IEEE 802.11, IEEE 802.1Qbv, WLAN**

I. Introduction

A. *TSN for Industrial Wireless*

Wireless connectivity is increasingly becoming the primary mode of communications on the factory floor. Its ease of installation and low maintenance overhead make wireless an attractive technology for factory communications. Factory communications include classes of devices and communications applications requirements ranging from basic ambient environmental sensing and the industrial internet of things (IIoT) to time-sensitive applications, mission critical sensing, precisely timed actuation, feedback motion control, and safety integrated systems (SIS).

Time sensitive applications often require communications transactions to occur within a single scan interval such as with time-precise actuation and robot synchronization. In these types of applications, information must arrive at their intended destinations on-time (i.e., with low latency and low jitter). Time-sensitive networking (TSN) has been developed as a set of standards by the IEEE 802.1 working group [1] to support applications needing time-based determinism. These standards were originally developed to support audio-video (AV) distribution systems and are now envisioned to support other time-critical applications such as those found on the factory floor. Having been originally developed for AV applications, TSN operates fairly easily with User Datagram Protocol (UDP) transport layer. However, most industrial protocols [2] are designed for Transport Control Protocol (TCP), such as Modbus/TCP, Ethernet/Internet protocol (IP), Profinet, Open Platform Communiciations (OPC), OPC/United Architecture (UA), Extensible Markup Language remote procedure call (XML-RPC), and Serial Real-time Communications System (SERCOS), requiring an explicit acknowledgment (ACK) for each TCP transmission, thus making the configuration of TSN functions such as time-aware scheduling (defined by the IEEE 802.1Qbv specification) more challenging. Additionally, each of these protocols is functional over a wireless physical transport, such as wireless local area network (WLAN), and is a desirable communications option for industry. TSN has been demonstrated to support TCP in a non-optimal manner in [3] with severe loss in channel bandwidth to accommodate the TCP ACK in the protected window. It is the central focus of this paper to demonstrate a TSN time-aware scheduling approach which supports the TCP transport more efficiently.

B. *Real-Time TCP-based applications overview*

Time critical applications require deterministic data delivery from the underlying network stack. The data becomes irrelevant if it is not delivered on time. Most of these applications rely on popular middleware or protocols like Modbus/TCP, OPC, and Ethernet/IP. Most of these protocols rely on TCP as transport layer for delivering data reliably. TCP is a connection-oriented protocol that provides reliable data delivery through acknowledgments and retransmissions, but it also has other very important features, namely, flow and congestion control. TCP's flow and congestion control mechanisms are very well studied with many versions available [4]. The goal of such mechanisms is to adapt the amount of data transmitted by the source to avoid or minimize congestion on the network while providing reliability. The TCP congestion control with its multiple phases (slow start, congestion avoidance, and congestion detection) was originally designed to minimize congestion on the Internet. On the other hand, the TCP behavior has an intrinsic randomness (e.g., as the number of bytes sent changes during the slow start and congestion avoidance phases) which goes against the principles of isochronous real-time control systems. This random TCP behavior is even more pronounced when using wireless communication as link conditions change dynamically due to various factors causing TCP transmission rate adaptation and inefficiencies [5].

As TSN technologies bring the possibility of converged networks for all types of traffic across wired and wireless links, the TCP behavior needs to be carefully considered in conjunction with TSN features, especially 802.1Qbv time-aware scheduling. For each time-critical application data flow from talker to listener, TCP creates a correlated data flow in the reverse direction (from listener to talker) for delivering acknowledgments that do not exist in connection-less transport. Timely delivery of the TCP ACK becomes important and needs to be considered in the TSN resource reservation mechanism.

C. Wireless TSN overview

TSN refers to a group of networking-related protocols and standards developed by the IEEE 802.1 working group in order to provide reliable, and time bounded (deterministic) delivery of data over 802-LAN (Local Area Network) technologies. Wireless Time-Sensitive Networking (WTSN) aims to extend the features as defined by this standard over wireless networks. WTSN allows industrial and robotics applications to share real-time information across a wired-wireless hybrid network. Two fundamental features of WTSN, namely Time Synchronization and Time Aware Scheduling (TAS) make this possible by guaranteeing deterministic and bounded latency to time critical streams. Time Synchronization is defined by IEEE 802.1AS which is a profile of the IEEE 1588 standard. It defines a protocol to distribute a common reference time to all nodes in a WTSN network to enable precise synchronization of time across all TSN nodes. TAS is defined by IEEE 802.1Qbv standard and it enables guaranteed delivery of time-sensitive data in the presence of interfering background traffic on the same network. By using the common time reference guarantee provided by the IEEE 802.1AS standard, the IEEE 802.1Qbv defines a set of time-controlled gates for each of the queues associated with the various traffic classes in each of the WTSN enabled nodes in the WTSN network. In this way, a time-aware schedule is defined and synchronized across all the nodes in a WTSN network for every traffic flow. This ensures timely delivery of data for time critical traffic flows as well as avoids interference from other flows during the transmission of critical packets.

D. Contributions and Paper Organization

In this paper, we will explore the challenges of implementing WTSN to provide deterministic delivery of time critical control data exchanged using the Robot Operating System(ROS) [6] middleware over TCP protocol with high efficiency. The collaborative robotic leader-follower use case serving as the testbed scenario for this research is described in detail in [3] and is shown in Fig.1. The contributions of this paper are as follows:

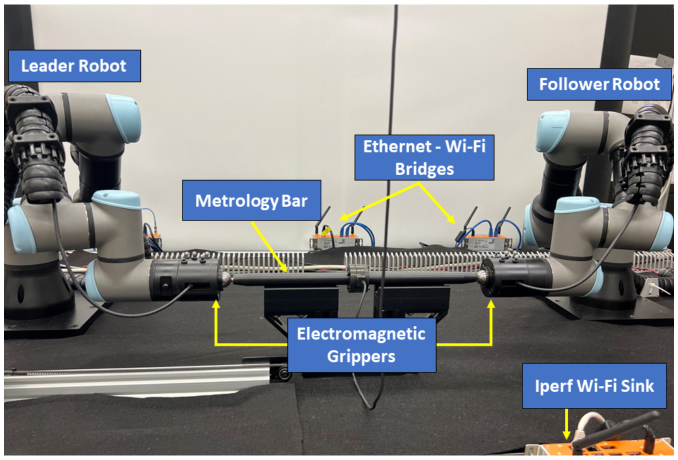

Fig. 1. Leader-Follower Collaborative Robotics WTSN Use Case. Two robots are synchronized by a velocity controller in which the Leader robot instructs the Follower robot trajectory information isochronously every 8 ms. Tight delay and jitter is critical for this application to function according to specifications.

1) We demonstrate a viable methodology for the design of 802.1Qbv schedule that accommodates a TCP traffic flow to support the operational requirements of a collaborative robotic scenario without wasting precious channel bandwidth by designing windows to independently accommodate the TCP data transmission and ACK.
2) We provide an operational scenario that is representative of actual industrial use cases and reusable by industry and other researchers to improve upon WTSN implementations.
3) We provide a discussion of the challenges to WTSN implementation and recommendations for improvements to the related standards.

We begin in Section II by describing the problem presented by the physical application and the TCP stack while deriving a WTSN schedule. In Section III, we present our methodology for experimentation and measurement. Results and discussions are then presented in Section IV followed by conclusions in Section V.

II. Efficient Time-Aware Scheduling for TCP

A. 802.1Qbv Time-Aware Scheduling over Wi-Fi

An implementation of the 802.1Qbv Time-Aware Scheduling over Wi-Fi has been described in [7]. It enables mapping of traffic streams to queues at the network stack (using the Linux Qdisc features [8]) and controls the queues (opening and closing) based on the requirements of the traffic streams with the goal of protecting the time-critical streams from interfering traffic.

The 802.1Qbv standard supports up to 8 different traffic classes between talker and listeners on the network, but without lack of generality, consider two classes of traffic -

Time Critical (TC) traffic and best-effort (BE) traffic between a talker and a listener end devices. The TC stream is assumed to be periodic with constant packet size and inter-arrival periods with strict delivery deadlines, whereas the BE traffic may include any other low priority traffic, typically without timing constraints. A time-aware schedule is characterized by a periodic cycle Time (T_{cycle}), which is the period at which the schedule repeats. Each cycle is divided into time slots (or protected windows) that can be mapped to different traffic classes (TC traffic and BE traffic). During a particular slot, only traffic belonging to that class is allowed to pass through the gate (gate is open) and all other traffic streams are blocked (gates closed). This gating rule is accomplished by a common schedule applied to all wireless nodes associated with the network and a single time reference for all nodes, which is enabled by precision time protocol (PTP) synchronization.

To derive the scheduling configuration including T_{cycle} and the slot times for the various traffic classes, a network configuration tool or scheduler need specific knowledge about each TC stream and the required footprint on the wireless medium. In other words, the scheduler needs to know the exact traffic profile for each TC flow and the amount of airtime required for transmitting the TC packets within a known deadline with a very high success probability. The protected windows also must strike a balance between meeting the minimum timing requirement of the TC traffic and maximizing the amount of traffic allowed on the network so that BE traffic requirements are not completely ignored. In addition, accommodation must be made for a slot where no traffic is let through, which is usually called the guard band and it is used to ensure BE traffic that starts at the end of its window can finish transmission before the following protected window.

The configuration of the protected periods must consider the packet size, the latency requirement of the traffic (deadline) and the effective data rate of the link. In the case of Wi-Fi links, in addition to the data rate, which is a function of the MCS (Modulation and Coding Scheme) used, the channel access delay also needs to be considered to compute an effective data rate. The 802.1Qbv gate control mechanism can be applied directly to the multiple queues within the 802.11 medium access control (MAC) layer and traffic can be released/blocked with high precision considering the appropriate configuration parameters for the link. Such implementation would require hardware changes to the Wi-Fi queuing management in the chipset, which would not be feasible with existing Wi-Fi cards. The current WTSN implementation using off the shelf chipsets [3] supports time-aware scheduling in software by leveraging existing 802.1Qbv implementation in the Linux OS using the Qdisc capability (see Fig. 2). In this implementation, the 802.1Qbv control of the OS queues is not tightly synchronized with the underlying Wi-Fi Queues and MAC behavior. Once the packets are released from the queues in the Kernel, they enter the Wi-Fi layer (mapped to one of the MAC queues), and the MAC layer follows the channel access procedure required for transmitting the frames over the air. Therefore, the amount of data passed down to the Wi-Fi layer during the multiple

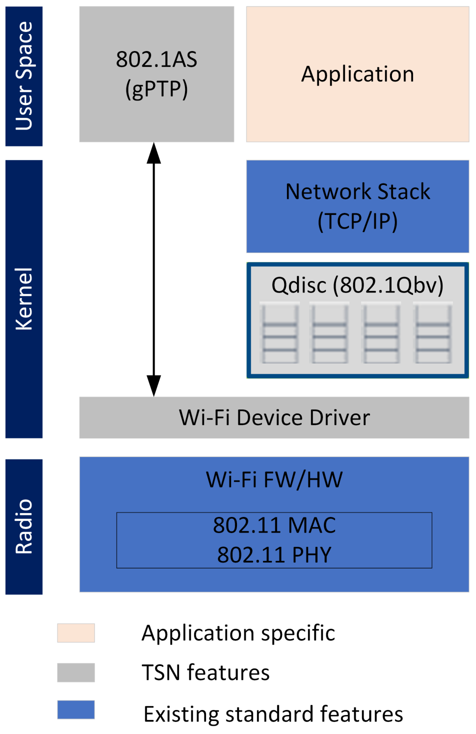

Fig. 2. Time-sensitive Networking 802.1Qbv Protocol Stack. All Application traffic traverses the 802.1Qbv Qdisc configured in the kernel of the operating system. The Qdisc configuration will determine the behavior of the TSN schedule and flow characteristics of the protected and best-effort windows.

time slots in the schedule needs to be carefully controlled to avoid overflowing into the next slot due to packets already buffered in the Wi-Fi queues that cannot be stopped by the 802.1Qbv gates at the operating system (OS) level.

Despite the limitations of software-based 802.1Qbv implementation, results have shown the benefits of avoiding congestion in the Wi-Fi domain and confirm that the 802.1Qbv layer is still able to control the latency as reported in several experiments [7]. Future enhancements in the hardware can optimize the queuing process and further reduce the achievable bounded latency.

B. *Scheduling Challenges for TCP Service*

The following issues arise when talkers generate application streams over TCP transport in TSN-capable networks:

a) Unidirectionality Assumption: The TSN configuration as defined in 802.1Qcc and the 802.1Qbv scheduling features deal with unidirectional streams from talker to listeners including all the overhead from higher layers including the combined application layer and TCP transport layers constituting the payload handled by the 802.1Qbv schedule. Once a TCP socket is opened for serving a given traffic stream, the listener automatically creates a correlated reverse flow for TCP ACK that may impact the behavior of the traffic stream generated by the talker. This is an open issue in most TSN configuration guidelines (802.1Qcc), as they only consider application data traffic streams from talkers to listeners independent of the transport service required.

b) Over-Allocation of Bandwidth: Accommodation of the TCP ACK requires additional time within the TSN protected window. This may lead to an over-provisioning of the resource, time, within the TSN schedule as the channel will tend to go idle while waiting for the ACK, thereby wasting channel bandwidth. In addition, packet aggregation may also lead to a waste of network resources when the listener discards some of the aggregated packets. One option to protect the TCP ACK is to set the protected window for the TC data stream to accommodate the TCP ACK in the reverse direction as shown in Fig. 3. This however creates a dependency on timely execution of other components in the network stack as the listener must generate the TCP ACK and make it available at the Wi-Fi layer for transmission during the same protected window as the corresponding TCP data packet. Moreover, the protected window duration must be large enough to accommodate the packet transmission, Idle wait time, and the ACK transmission. In our previous work [3], the Idle time was more than an order of magnitude of the packet transmission time. This problem is addressed in this paper in which a separate protected window is created to accommodate the ACK with best-effort traffic arriving in between.

c) Packet Aggregation and Variable Payloads Sizes: If the reverse stream carrying the TCP ACK is not prioritized to ensure timely and reliable delivery, it may trigger the aggregation of application packets at the talker. For example, when a TCP ACK is delayed, it may result in the TCP layer queuing up future packets till the TCP ACK is received. This creates a risk that the protected period is too small for the larger aggregated packet. In some applications, such as ROS-based applications, the listener is not aware of the data aggregation by TCP, and it could consume the first message while ignoring the remaining messages in the aggregated packet. This could cause the application to misbehave by using stale data or missing critical control data. Any excessive delays or loss of the TCP ACK might also trigger congestion control changing the amount of data that the talker sends.

C. Optimized Time-Aware Schedule

Understanding the time-sensitive traffic requirements is a fundamental step in designing an optimized time-aware schedule. The most critical stream in this collaborative robotics use case is the periodic communication between the leader and follower robots, whereby the leader conveys the position that the follower should move to in the next step. It is important to ensure timely delivery of position data to the follower in a deterministic fashion without being affected by other traffic streams on the same network. In our implementation, based on the ROS middleware, new position data is generated by a ROS node (leader) every 8 ms (125 Hz). The ROS application data is written to the TCP stream interface, which generates TCP packets at the transport layer and delivers them to the TSN layer in the kernel. The scheduler identifies the data segments as part of the TC stream and places them into the protected

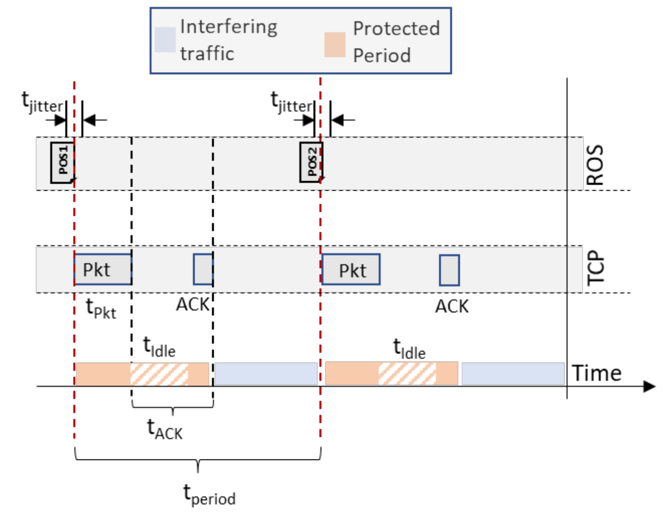

Fig. 3. Resource Over-Allocation. Allocation of bandwidth to accommodate the TCP ACK within the same protected window creates a variable channel Idle time thus wasting resources. A solution is needed to minimize the amount of Idle time needed for the ACK thus optimizing channel bandwidth availability.

window queue. Once the TC queue opens, the TSN scheduler uses a time-aware traffic shaping algorithm to determine how many bytes may be transmitted according to the parameters of the configured TSN Qdisc. Once data passes through the TSN gating mechanism, it arrives at the physical layer device - in this case, Wi-Fi for over-the-air transmission.

Inspecting the TC traffic and characterizing its time footprint over the air is important to properly set the protected window size. We measured the transmission time of the position packets (t_{Pkt}) over the Wi-Fi link using a real-time spectrum analyzer (RTSA) and observed $t_{\mathrm{Pkt}} = 88\ \mu s$ at MCS 15 and $t_{\mathrm{Pkt}} = 110\ \mu s$ at MCS 13. Given the testbed configuration, the links between the access point (AP) and the wireless stations (STAs) have high signal-to-noise ratio (SNR) values, and MCS 13 was the worst-case MCS used for leader-follower transactions. In addition to the transmission time over the air, we also need to consider other overheads in the 802.11 protocol, including the channel access delay as well as potential contention from other 802.11 transmissions in overlapping networks. Given the unlicensed operation in the 2.4, 5, and 6 GHz bands, it is possible that other devices that are not part of the managed TSN infrastructure get access to the channel that increases channel access delays for the STAs trying to transmit the TC traffic within the protected window. In a managed network, which is expected to be the case for most industrial deployments, admission control policies and frequency planning can be used to minimize the potential impact of "unmanaged" devices, i.e., devices that do not implement TSN features and may not adhere to the timing and scheduling rules set for the network. There are several techniques that can minimize the impact of unmanaged networks, including multi-link operation and 802.1CB redundancy, which are outside the scope of this

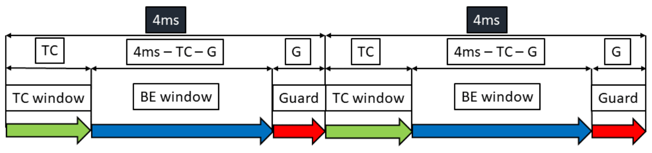

Fig. 4. TSN schedule for the accommodation of the TCP acknowledgement. Here, the duration of the time-critical traffic window is TC, and G is the guard interval. The total schedule period is 8 ms which matches the scan interval of the leader-follower velocity controller. The TC and G durations are configurable where TC is an integer multiple of the worst-case packet transmission time and G is a function of the Qdisc limit buffer size.

work, but could be incorporated into future work.

To accommodate the TCP behavior with minimal impact on the application performance and network efficiency, we have defined a schedule template, illustrated in Fig. 4, where TCP data segments and ACKs belong to the same TC traffic class but can use different protected windows. We chose to use a symmetric TC window configuration for the TCP transmission. In the case where a packet is missed in a TC window, the next cycle can transmit the packet, regardless of weather it is a data packet or ACK. We defined the T_{cycle} to be half of the application period, denoted by t_{period}, so that the position data can be transmitted in the first 4 ms cycle (within the protected window), while the ACK can be transmitted in the next protected window thus closing the 8 ms scan cycle. Note that depending on the protected window size, it is also possible for the ACK to be ready for transmission within the same window as the TCP data segment. If the ACK is not yet available for transmission to catch the first window, it has another opportunity in the next scheduling cycle. This approach provides a template for trading off the performance of the robot application with channel availability for the best-effort traffic. Ultimately, the robot application will have strict requirements on delay and jitter, and this places limits on how small the protected window sizes can be, given the channel conditions.

III. Measurement and Experimental Methodology

A. Testbed Setup

The leader robot follows a pre-planned circular path, for which the follower uses a real-time velocity controller, fed with positional updates from the leader at 125 Hz, or an 8 ms update interval, to keep up with the leader. Under ideal communications there is typically 13-15mm of error between where the follower is and where it should be, which is a property of the speed of the leader and gains for the velocity controller. This leader-follower ROS topic stream uses a predefined port, such that the WTSN schedule protects this traffic in the TC window. For the WTSN experiments, an 802.11Qbv schedule was run using software emulation on the AP, leader STA, and follower STA. This way, the AP gates the best-effort and protected traffic, while the leader and follower nodes are in sync, allowing for the protected traffic to be sent in time with the TC window. In this case, the iPerf traffic flows from the AP to another wireless STA using UDP, to emulate a camera's data stream.

The data streams are collected across the testbed using Ethernet-based tap devices called SharkTaps, which are routed to the globally synchronized data collection machine for packet captures. These packet captures allow for the calculation of packet flight and inter-arrival times, to analyze the latency performance of the network. Since we use Ethernet to Wi-Fi bridges, we can capture all relevant packets that are sent through the wireless network. We use IEEE 1588, PTP [9] to synchronize the time for the wireless stations, AP, and collection machine, for data collection and WTSN schedule timing purposes. We have observed the timing errors to be within <1 μs of error from the Grand Leader (GL) clock with 99% confidence.

B. Data Collection

Compared to our previous work, we have improved the position tracking of the robots by installing a infrared tracking system utilizing four OptiTrack 13W cameras, to reduce overall measured uncertainty of positional errors from 1.4 mm to 0.22 mm at the 99% confidence level. Moving this measurement to a singular camera tracking system greatly improved the uncertainty for the position measurements of each robot. For the latency measurements, we have calculated the uncertainty to be <22 μs with the 99% confidence level using SharkTaps.

For the experiments, a wired baseline test was performed to see the performance of Ethernet-based connections as a benchmark. Then, we enabled the wireless stations and the AP, without WTSN, with increasing levels of traffic using the iPerf [10] source and sink. The sink is wireless, such that the AP transmits a UDP stream of 16, 32, 48, 64, or 80 Mbps, wirelessly. The packet length is 1000 Bytes for these transmissions, emulating a wireless camera stream to stress the network. Then, we enabled the WTSN schedule with various window sizes with a 2, 4, 6, and 8x multiplier of a maximum transmission time of 110 μs for the leader-follower traffic, which we measured using a time-domain radio frequency (RF) capture. After picking a multiplier for the TC window size, based on the physical performance requirement to be discussed in the next section, we then performed the same iPerf traffic stream levels, to show that the performance is not greatly altered with high levels of BE traffic.

C. Objectives

While optimizing the schedule to adapt to the TCP traffic, we deal with a number of objectives at different levels. The network level objective is measured through the round-trip time (RTT) of the TC stream, which can be minimized ideally by allowing the TC traffic the explicit use of the channel with no interfering streams. The application-level objective is measured through the Cartesian error between the leader and follower locations. Similar to the RTT of TC packets, the Cartesian error is minimized by having no interfering streams. On the other hand, we consider channel utilization objectives that target the efficient use of the channel by allowing the TC stream to achieve a predefined level of performance, while maximizing the corresponding channel efficiency metrics.

We define the protected window utilization metric by the ratio of the time utilized by the TC packet and its acknowledgment to the total time of the protected window as follows,

$$U_{\text{TC}} = \frac{t_{\text{pkt}} + t_{\text{ack}}}{2\text{TC}}. \tag{1}$$

The denominator has the protected window size multiplied by 2 because the schedule allows for two protected windows to occur during a single update period to accommodate the position update packet from the leader and its corresponding ACK from the follower. The other metric is the residual channel bandwidth after protecting the TCP flow which is defined by:

$$R_{\text{CH}} = 1 - \frac{2\text{TC}}{t_{\text{period}}}. \tag{2}$$

D. Schedule Tuning

In this work, we have a single TC stream that needs to be scheduled, and hence, we have two main tuning parameters for the schedule. These parameters are the TC protected window size (t_{TC}) and the guard band (G). The guard band is mainly tuned to allow for any remaining traffic from the non-TC streams to finish and guarantee that the wireless channel is free and ready for the TC traffic. Note that the guard band is very important in the TSN setup, where the scheduling is performed at the Kernel, and hence, the guard band allows any traffic at the Radio space to be finished before the start of the protected window. The other parameter is the size of the protected window, which should be large enough to allow for the TC packets to be transmitted successfully accommodating any overhead by the lower layers and should not be too big to balance the utilization and residual channel metrics as described above.

We set the protected window size as a base value of 110 μs scaled by a multiplier factor (5x, 10x, …). We conducted multiple experiments increasing the protected window size by various scaling factors, as described in the next section. We also configured a guard band based on the worst-case packet transmission time for the BE traffic. In the next section, we evaluate several performance metrics and efficiency tradeoffs for various configurations of the protected window sizes.

IV. Results

This section highlights the experiments performed with the dual-lift use case, under various network configurations. Data used occurs during the circular path of the leader and follower, which under ideal communication should have a fairly constant error value. We include a wired baseline with all Ethernet connections to show this benchmark. At the application level, we can tolerate up to 15.7mm of root mean squared (RMS) Cartesian error between the leader's and follower's positions. Further, we compare the system performance under TSN scheduling to the basic carrier sense multiple access (CSMA) scheme where all traffic streams contend for the wireless medium. However, in the case of no interfering traffic, the CSMA is considered the ideal wireless case in which the wireless data will not wait for its protected window to access the channel but will have access to the channel faster as there is no competing traffic.

A. Qdisc Tuning

In order to perform the 802.1Qbv traffic shaping at the Kernel, we tune the Qdisc traffic control at the Kernel to both apply the schedule and limit the traffic going to the Wi-Fi layer. The TSN schedule is performed at the Kernel so the protected windows and guard band gates are opened to allow data to the queue of the Qdisc traffic shaper. As a result, the buffer size needs to be tuned to prevent the BE stream to fill it to the extent that it occupies the channel in the TC protected window. Also, it has to allow the BE traffic to buffer at least two packets to leave a space for a second ready packet while one is being transmitted. We set the LIMIT to two multiplied by the maximum transmission unit (MTU) of the BE traffic which equals to 2084 including the UDP overhead.

The second parameter is the BUFFER limiting the maximum burst size that can go to the Wi-Fi layer. It is tuned to allow a single packet of the BE traffic from iPerf to go through with other smaller size packets from other BE streams from the operational testbed. We set the BUFFER, or the burst size, to 1500 bytes. Finally, the RATE is the token arrival rate for the token bucket algorithm performed by the Qdisc which controls the average rate of the data going to the Wi-Fi layer and we tune it to be the maximum allowed data rate at the wireless channel to avoid traffic delay or loss. We set the RATE to 104 Mbps corresponding to the observed bit rate achieved with MCS 15 due to the IEEE 802.11 MAC overhead.

B. Multiplier Selection

In this subsection, we show the detailed results for the multiplier selection to accommodate any Wi-Fi layer processing to allow the TC traffic to be transmitted during the protected window. We present the leader-follower performance at the network and application layers for different multipliers of the TSN schedule with no interfering traffic to the wired baseline, and the CSMA wireless ideal benchmark. In this results, we set the guard band to 770 μs which equals to six times the time needed for a single packet of the BE traffic to be transmitted.

Figs. 5 and 6 illustrate the different multipliers (2-8x) tested to determine what protected window length is optimal, based on the network and physical performance of the use case. We have determined that the 6x multiplier, or 660 us, to be optimal, as it is less than the 15.7mm threshold, while still providing ample time for the BE traffic. Moreover, at the network level, almost 90% of the packets have a RTT less than 8 ms and 98% of the packets have a RTT less than 12 ms, which is considered adequate for this sort of application.

C. Channel Efficiency Tradeoff

In this section, we present the tradeoff due to the multiplier selection between the improved network and application per-

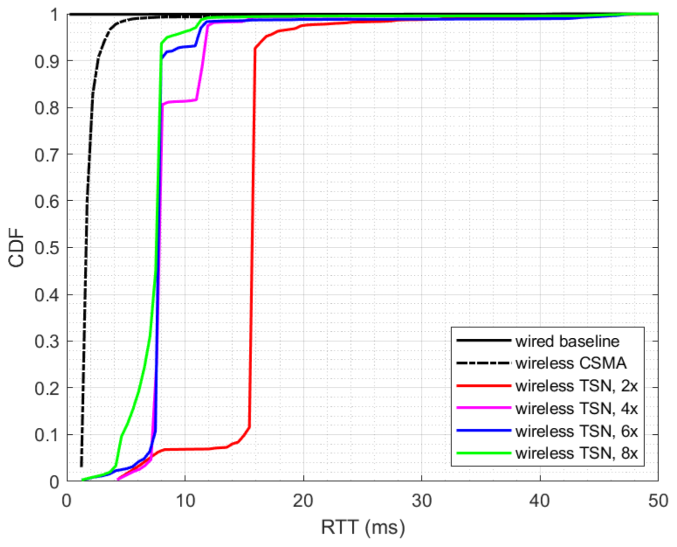

Fig. 5. RTT of leader-follower position data using WTSN schedule with 2-8x multipliers.

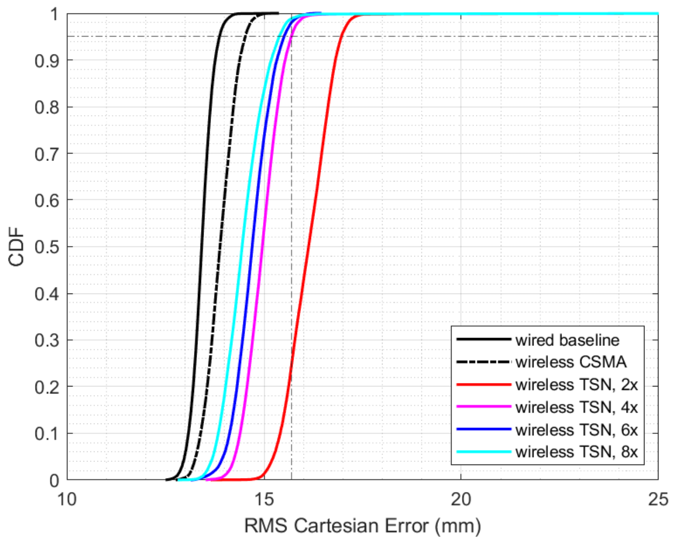

Fig. 6. WTSN schedule with 2-8x multipliers with no interfering traffic. The vertical dashed and dotted line represents the 15.7mm error operational requirement threshold and the horizontal line represents the 95th percentile.

formance metrics, and the channel usage efficiency metrics. In Table I, we present the results when the multiplier of the protected window equals 2, 4, 6, and 8. We present the root mean square (RMS) Cartesian error at the 95th percentile, the percentage of the data packets with RTT $\leq$ 8 ms and 12 ms, the protected window utilization as defined in (1), and the channel bandwidth residual after protecting the TCP flow as defined in (2). Additionally, one can see that the proposed schedule for accommodating the TCP traffic has a better protected window utilization compared to the corresponding scheduling scheme in [3], where the protected window had been set

TABLE I
COMPARISON OF MULTIPLIER IMPACTS TO OPERATIONAL AND NETWORK PERFORMANCE

Multiplier	2	4	6	8
RMS Cartesian Error at 95th Percentile	16.9	15.7	15.4	15.3
Percentage Packets with RTT $\leq$ 8 ms	7	80	90	94
Percentage Packets with RTT $\leq$ 12ms	7	97	98	99
Protected Window Utilization (U_{TC})	0.42	0.21	0.14	0.11
Channel Bandwidth Residual (R_{CH})	0.89	0.78	0.67	0.56

to accommodate both the packet and acknowledgement. In that work, a protected window utilization of 5.6% occurred to achieve the same use case performance compared to the 14% value in the proposed TSN scheduling approach. The tradeoff is demonstrated such that a procedure for selecting an application-specific value for the multiplier can be performed by building a similar table with the corresponding application requirements.

As shown in Table I, one can observe that increasing the protected window size can improve the network and physical performance in the leader-follower use case. However, this improvement rate decreases at higher multiplier values. In this case, for example, the improvement in performance by changing the multiplier from 6 to 8 is much less than the improvement by changing the multiplier from 4 to 6.. On the other hand, a similar increase has a clear negative impact on the protected window utilization and the channel bandwidth residual. Hence, this demonstrates the need for optimizing the multiplier value with respect to various use case objectives.

D. best-effort Traffic Impact

In this subsection, we show the impact of allowing an iPerf traffic stream to be transmitted at the BE window of the schedule. We vary the rate of the iPerf interfering traffic from 16, 32, 48, 64, and 80 Mbps. We show these values at the legends of the following figures. We compare it to the cases of the wired baseline and the case of no interfering traffic, which is referred to as having a rate of 0cMbps.

It can be observed from Fig. 7 that higher interference levels with the CSMA scheme with no TSN, especially 48mbps and above, are impacting the performance of the use case greatly with higher average error, and most notably, jerkiness in the movement of the follower. To remedy this, we implement the wireless TSN schedule discussed in Section III.

In Fig. 8, we present the physical performance of the testbed with the existence of the iPerf interfering BE traffic. The physical performance is clearly improved compared to the CSMA performance in Fig. 7. In the higher values of BE traffic rates, the degradation rate of the performance while deploying the TSN schedule is much lower in the CSMA case.

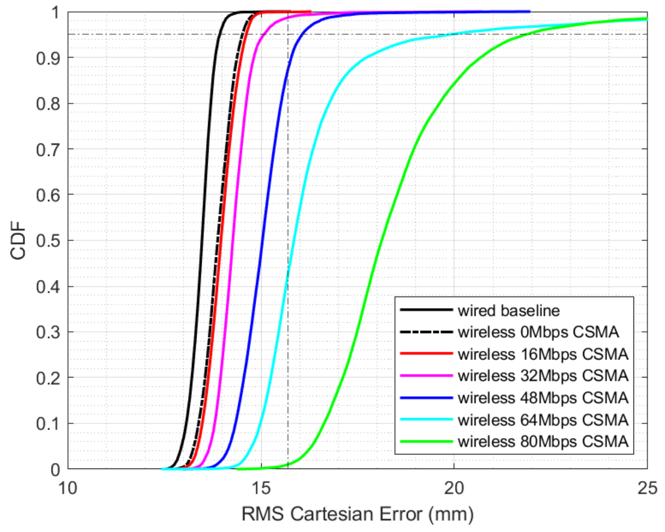

Fig. 7. Leader-follower error using Wi-Fi CSMA with 0-80Mbps interfering traffic. CDF plot shown to show the increasing error with higher interference levels.

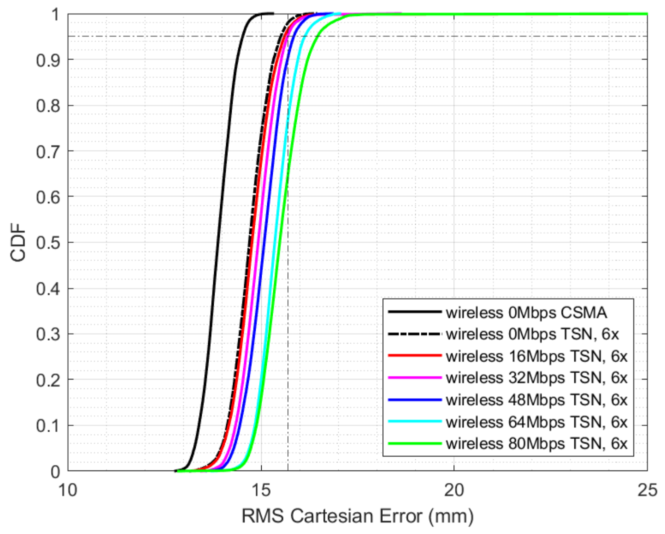

Fig. 8. WTSN schedule with 6x muliplier with 0-80 Mbps of BE traffic CDF. The vertical dashed and dotted line represents the 15.7 mm error operational requirement threshold and the horizontal line represents the 95th percentile.

V. Conclusion

Reliable protocols are ubiquitous in industrial networks and must be considered when discussing time-aware scheduling of network traffic for use in industrial time-critical applications. Time-critical industrial applications include precision actuation, feedback control, and safety-integrated systems. In this work, a methodology for 802.1Qbv scheduling of wireless time-critical industrial traffic in a collaborative robotic scenario that accommodates the TCP acknowledgement and efficiently utilizes the channel bandwidth is presented. Through careful analysis of the traffic patterns for a particular use case, this efficiency scheme may be utilized for other industrial wireless applications that must coexist in an heterogeneous mix of traffic. While the merit of this approach has been demonstrated in this paper, more research is required to reduce the impact of high-throughput best-effort traffic conditions, as exemplified in the results.

Next steps to this research may include further optimization of Qdisc 802.1Qbv configuration to enable tighter instantaneous bandwidth control of the best-effort traffic window, eliminating overflow of packets in the guard interval into the next protected time-critical window. Finally, synchronization is needed between the application, network stack and the TSN control layer which would include adding time-awareness to the application layer, TCP/IP implementation, and Wi-Fi medium access control with channel management implementations. Finally, the method presented here requires strict control of the channel conditions to maintain the MCS above 13. Under extreme channel conditions in a production environment, the time-aware schedule would need to quickly adapt the best-effort allocated bandwidth and extend the protected window duration to accommodate a lower transmission bit rate.

Disclaimer

Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

References

[1] IEEE 802.1 Time Sensitive Networking (TSN) Task Group: https://1.ieee802.org/tsn/.
[2] Eric D. Knapp, Joel Thomas Langill, ”Industrial Network Security (Second Edition)”, Syngress, 2015, ISBN 9780124201149, https://doi.org/10.1016/B978-0-12-420114-9.00018-6.
[3] R. Candell, K. Montgomery, M. Kashef Hany, S. Sudhakaran, J. Albrecht and D. Cavalcanti, ”Operational Impacts of IEEE 802.1Qbv Scheduling on a Collaborative Robotic Scenario,” IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society, Brussels, Belgium, 2022, pp. 1-7, doi: 10.1109/IECON49645.2022.9968494.
[4] A. Afanasyev, N. Tilley, P. Reiher and L. Kleinrock, ”Host-to-Host Congestion Control for TCP,” in IEEE Communications Surveys and Tutorials, vol. 12, no. 3, pp. 304-342, Third Quarter 2010, doi: 10.1109/SURV.2010.042710.00114.
[5] A. Mammadov and B. Abbasov, ”A review of protocols related to enhancement of TCP performance in wireless and WLAN networks,” 2014 IEEE 8th International Conference on Application of Information and Communication Technologies (AICT), Astana, Kazakhstan, 2014, pp. 1-4, doi: 10.1109/ICAICT.2014.7035964.
[6] Jason M. O’Kane, ”A Gentle Introduction to ROS”, Jason M. O’Kane, 2013, ISBN 1492143235.
[7] S. Sudhakaran, K. Montgomery, M. Kashef, D. Cavalcanti and R. Candell, ”Wireless Time Sensitive Networking Impact on an Industrial Collaborative Robotic Workcell,” in IEEE Transactions on Industrial Informatics, doi: 10.1109/TII.2022.3151786.
[8] V. Gomes, “TAPRIO - Time Aware Priority Shaper.” https://manpages.ubuntu.com/manpages/focal/en/man8/tc-taprio.8.html.
[9] ”IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems,” IEEE Std 1588-2019 (Revision of IEEE Std 1588-2008) , vol., no., pp.1-499, 16 June 2020, doi: 10.1109/IEEESTD.2020.9120376.
[10] “iPerf - The ultimate speed test tool for TCP, UDP and SCTP,” https://iPerf.fr/iPerf-doc.php