An Analytical Evaluation for Software-based TSN in Industrial Wi-Fi Networks

Mohamed Kashef (Hany), Karl Montgomery, and Richard Candell
Smart Connected Systems Division, National Institute of Standards and Technology (NIST),
Gaithersburg, MD, USA
Emails: {mohamed.kashef, karl.montgomery, richard.candell}@nist.gov

***Abstract*—Time-sensitive networking (TSN) is an emerging topic for advancing wireless networking for industrial applications. TSN, as defined under the umbrella of IEEE 802.1 working group standards, addresses issues related to providing deterministic communications over IEEE 802-based Local Area Networks (LANs). In this work, we analyze the data balance equations at various levels for software-based wireless TSN (WTSN) applied over IEEE 802.11 lower network layers. We consider the case of a single time-critical traffic stream with a requirement of being transmitted over protected time slots while another best-effort traffic stream also uses the same wireless channel. We describe the expected channel usage and evaluate the ability of the software-based WTSN algorithm to satisfy the time-critical traffic stream requirements.**

***Index Terms*—Industrial wireless, Wi-Fi, wireless time-sensitive networking**

I. Introduction

Industrial wireless is essential for enabling smart manufacturing enhanced digitization, and increased intelligence in industrial automation. The operational requirements of industrial systems put stricter requirements on industrial wireless networks than for that of office and home wireless networks. One of these requirements is the data timeliness of time-critical applications, especially for robotic and mobile industrial equipment. The data timeliness includes both the time synchronization and low latency of communications within an industrial wireless network [1], [2].

Time-sensitive networking (TSN) defines a set of standards for achieving data timeliness over the IEEE 802 family networks. It generally describes how time-sensitive data is transmitted over networks with tight synchronization and latency guarantees [3]. Wireless TSN (WTSN) is the wireless counterpart of the wired TSN, which faces the challenges associated with wireless networking, such as reliability, stability, and interference-related effects. The TSN protocols can be extended to the IEEE 802.11 Wi-Fi, a part of the IEEE 802 family, to achieve WTSN capabilities without significant architecture changes [4], [5].

One aspect of WTSN is the traffic scheduling of the time-critical (TC) traffic streams to coexist with the best-effort (BE) traffic over the same Wi-Fi network. Traffic scheduling can be achieved by implementing the IEEE 802.1Qbv protocol that defines priority queues for traffic forwarding, based on their quality of service (QoS) requirements [6], [7]. A TSN-enabled access point (AP) is needed to moderate the flow of queued traffic from various nodes. This AP has to be tightly synchronized with all TSN-enabled nodes in the network.

One direct software-based implementation of IEEE 802.1Qbv was achieved by using the Qdisc token bucket algorithm with a gated input at Linux-based devices equipped with Wi-Fi interfaces. This implementation of the 802.1Qbv time-aware scheduling over Wi-Fi has been described in [8]. It enables the mapping of traffic streams to queues at the network stack (using the Linux Qdisc features [9]) and controls the queues based on the requirements of the traffic streams.

In this paper, we start by reviewing the basics of the token bucket traffic shaping algorithm being deployed in the software-based WTSN implementation over Wi-Fi. We further present the architecture and the data flow of this implementation and the governing equations that can be used to analyze the implementation performance. We used the resulting equations to study the analytical performance of this architecture and its limitations due to being implemented on the Kernel of the Linux device and without any changes to the underlying Wi-Fi implementation.

II. Time-aware Scheduling Protocol Structure

The scheduling protocol allows the data packets to go through few stages before traversing the wireless channel. Application data passes through the various network stack layers for packets to be available for the Wi-Fi chipset to perform the lower MAC and PHY layers operations. In the Linux kernel, before the packets are delivered to the Wi-Fi chipset, on-off packet gates are used to schedule the various traffic streams, followed by Qdisc traffic shaping that implements a token bucket algorithm.

A. The Token Bucket Algorithm

A token bucket protocol has been generally used for various traffic control functionalities in data networks. Basically, when the algorithm is applied, a data packet cannot be moved out of the network layer, except if the number of tokens in the token bucket is larger than or equal to the amount of bytes in the data packet. The token bucket is defined by its size, which determines the maximum burst size of data that can move out,

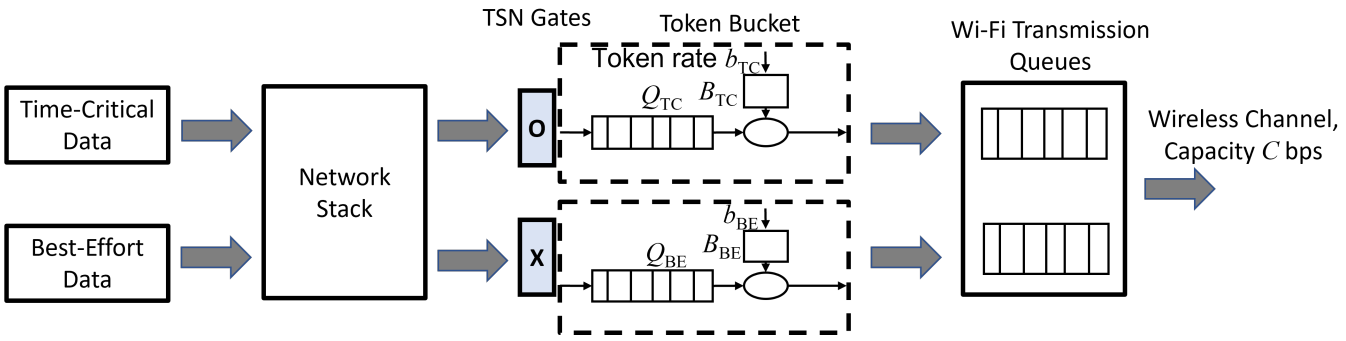

Fig. 1. Block diagram of various stages of the data flow through a TSN enabled device when the TSN gate for the time-critical stream is open.

and the rate of arriving tokens, which determines the average data rate that can move out of the network layer. Moreover, a data queue is introduced in this algorithm to determine the maximum amount of data that can be buffered for processing by the token bucket algorithm [10].

When applying TSN, traffic can incur data loss and delay due to the application of token bucket traffic shaping. Data can be discarded when the data buffer is full, which may occur due to a low token rate or data burstiness that causes the buffer to be full while no tokens are available. Data burstiness can also cause data delays when the data is not discarded, but rather stored in the token bucket data buffer, waiting for tokens to become available. On the other hand, tokens can be discarded, and hence, a lower average data rate is achieved when the token bucket is full and the data buffer is empty. This only causes a problem when it leads to data rate reduction because of the lost tokens. Hence, tuning is needed for the token bucket size, data buffer, and token rate to achieve the required traffic shaping objectives.

B. Time-aware Scheduling Protocol Data Flow

To achieve a full WTSN implementation, a complete network stack has to be implemented to apply a full gating approach for TC and BE traffic streams in order to achieve the required deterministic latency. A complete WTSN network stack implementation requires the design and implementation of the corresponding wireless chipsets which may be complicated. However, the use of traffic shaping in the upper layer above Wi-Fi chipsets can achieve close-to-optimal performance with respect to the WTSN requirements. In Fig. 1, the data flow of packets through the network stack of a software-based WTSN-enabled device is presented.

Traffic is identified by the user to be either TC or BE based on the respective latency requirements and then go through all of the network stack to prepare packets and associated headers before forwarding to the Wi-Fi chipset for wireless channel transmission. The gates of the TSN schedule and the Token Bucket protocol are implemented in software by using the Qdisc on the Linux kernel to follow the TSN schedule by allowing specific streams to forward packets to the Wi-Fi chipset at the specified schedule timings [11].

The packets travel through the Token Bucket algorithm, and the corresponding data queues accept the data and perform the traffic shaping before forwarding the packet to the Wi-Fi hardware, which implements the lower MAC and PHY layers. The following are the three main parameters to be tuned while configuring the Token Bucket queuing implementation. First, the limit parameter is the data buffer size, which is the number of bytes that can be queued waiting for tokens to become available. Second, the burst parameter is the bucket size, in bytes, which is the maximum number of bytes that tokens can be made available instantaneously. Third, the rate parameter of the tokens defines the average rate at which the data is dequeued from the data buffer.

The final layer is the wireless interface which contains the Wi-Fi transmission queues. TC data is forwarded to a higher priority queue, and the BE data is forwarded to a lower priority queue to make sure that the TC traffic gets higher priority for transmission even when the BE traffic transmission queue has data. This method is used to provide priority queues at the MAC layer where data packets are segregated on the basis of priorities. Whenever a particular station has access to the channel, it transmits data from the transmission queue that has the highest priority among the queued packets as described by the enhanced distributed channel access (EDCA) mechanism [12]. There are four access categories such that four priority queues are established to prioritize data frames within each station. When frames from two or more queues are ready to transmit at the same time, the higher priority queue is granted access to transmit and the other queues act as if a physical collision occurred during transmission.

C. Governing Equations

In this section, we derive the governing equations of the above data flow procedure in order to evaluate various performance metrics related to this WTSN implementation. We consider the case for a single TC traffic stream and a single BE traffic stream, as shown in Fig. 2. Based on the final results, we can extend the conclusions to the case of multiple TC traffic streams.

The TSN scheduling is performed by opening and shutting the synchronized TSN gates, The variable $G_{\mathrm{TC}}(k)$ represents

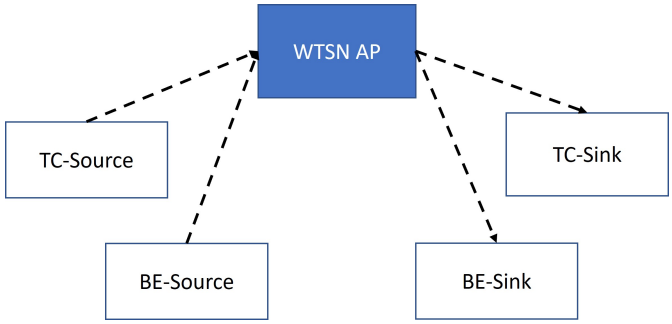

Fig. 2. The network setup with a single TC stream and a single BE stream, where the governing equations are derived.

the status of the TC traffic gate at the time slot k. It is defined as follows:

$$G_{\text{TC}}(k) = \begin{cases} 1 & 0 \leq (kh \bmod T) \leq t_{\text{TC}} \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where h is the time slot length, t_{TC} is the TC protected window size, and T is the TSN scheduling period. Similarly, $G_{\text{BE}}(k)$ represents the status of the BE traffic gate at the time slot k.

$$G_{\text{BE}}(k) = \begin{cases} 1 & t_{\text{TC}} \leq (kh \bmod T) \leq t_{\text{TC}} + t_{\text{BE}} \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where t_{BE} is the BE window size and the rest of the schedule period $T - t_{\text{TC}} - t_{\text{BE}}$ is called the guard interval, which should not allow any BE traffic to go to the Token Bucket buffer, such that any queued BE traffic can be transmitted within the guard interval.

Second, we derive the equations that represent the states of the various queues at the token bucket algorithm part. The token bucket data queue state at time k is denoted by $q_{(X)}(k)$, where $X \in \{\text{BE, TC}\}$ and is an indicator to a specific path of the data stream. The update equation of the state of the data buffer is as follows

$$\begin{aligned} &q_{(X)}(k+1) = q_{(X)}(k) \\ &+ \min(V_{(X)}(k), Q_{(X)} - q_{(X)}(k)) - min(q_{(X)}(k), b_{(X)}(k)) \end{aligned} \tag{3}$$

where $Q_{(X)}$ is the size of the data queue, $V_{(X)}(k)$ is the amount of arriving data, and $b_{(X)}(k)$ is the token bucket state. The arriving traffic can originate from the TC or BE streams. Also, there could be no incoming traffic if the stream corresponding to the open gate does not have any available data.

The value of $V_{(X)}(k)$ is different depending on the corresponding data stream. As a result, the value of $V_{(\text{TC})}(k)$ and $V_{(\text{BE})}(k)$ can be formulated as follows

$$V_{(\text{TC})}(k) = \begin{cases} s_{\text{TC}} I_{\text{TC}}(k) & G_{\text{TC}}(k) = 1 \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

$$V_{(\text{BE})}(k) = \begin{cases} s_{\text{BE}} N_{\text{BE}}(k) & G_{\text{BE}}(k) = 1 \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where s_{TC} and s_{BE} are the packet sizes for the TC and BE traffic streams, respectively, $N_{\text{BE}}(k)$ is the number of BE packets at a time slot, and $I_{\text{TC}}(k)$ is the indicator if a TC packet exists at a time slot. The TC stream usually includes control-related messages in industrial applications such that a single packet is transmitted through the schedule period. However, if multiple packets are being transmitted, a similar approach to the BE traffic will be used such that a number of packets will be used in the arriving traffic equation. We assume that the TC data packet is generated in the first time slot of the protected window without loss of generality, because the TC packet is defined as a short packet, and hence, would not affect the BE traffic window. The TC traffic indicator is evaluated as follows

$$I_{\text{TC}}(k) = \begin{cases} 1 & (kh \bmod T) = 0 \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

To evaluate the probability distribution of $N_{\text{BE}}(k)$, assuming a Poisson distribution where λ is the BE data arrival rate, we use the following equation,

$$\Pr(N_{\text{BE}}(k) = n) = \frac{(\lambda h)^n \exp^{-\lambda h}}{h!} \tag{7}$$

Then, the state of the token bucket queue at time slot k, denoted by $b_{(X)}(k)$, is represented as follows:

$$\begin{aligned} &b_{(X)}(k+1) = b_{(X)}(k) \\ &+ \min(r_{(X)}h, B_{(X)} - b_{(X)}(k)) - min(q_{(X)}(k), b_{(X)}(k)) \end{aligned} \tag{8}$$

where $B_{(X)}$ is the size of the token bucket and $r_{(X)}$ is the arrival rate of the tokens to the bucket.

The last phase in the data flow is going through the Wi-Fi priority queues to allow the TC traffic to be transmitted, even if there is BE traffic buffered at its Wi-Fi BE queue. We denote the amount of data going out from the token bucket algorithm at the time slot k by $q_{\text{o}_{(X)}}(k)$. It is evaluated as follows:

$$q_{\text{o}_{(X)}}(k) = min(q_{(X)}(k), b_{(X)}(k)) \tag{9}$$

The data is then forwarded to its corresponding priority queue, where the BE priority queue state is defined as follows:

$$\begin{aligned} q_{\text{p}_{(\text{BE})}}(k+1) = q_{\text{p}_{(\text{BE})}}(k) + q_{\text{o}_{(\text{BE})}}(k) - \\ \min(\max(Ch - q_{\text{p}_{(\text{TC})}}(k), 0), q_{\text{p}_{(\text{BE})}}(k)) \end{aligned} \tag{10}$$

where $q_{\text{p}_{(\text{BE})}}(k)$ and $q_{\text{p}_{(\text{TC})}}(k)$ are the states of the BE and TC priority queues, respectively, and C is the wireless channel capacity. As a result, the rate at which the data will be forwarded to the wireless channel is limited by Ch. Finally, the term $q_{\text{p}_{(\text{TC})}}(k)$ is evaluated as follows

$$q_{\text{p}_{(\text{TC})}}(k+1) = q_{\text{p}_{(\text{TC})}}(k) + q_{\text{o}_{(\text{TC})}}(k) - \min(Ch, q_{\text{p}_{(\text{TC})}}(k)) \tag{11}$$

D. Performance Metrics

In the following, we present the various considered performance metrics that represent the impact of the software-based TSN implementation on the wireless network performance, and the challenges that face this implementation to achieve the required latency determinism. Two main challenges are studied. First, the utilization of the wireless channel, due to applying a schedule, is affected by resource reservation. Second, the protected window of the schedule for the TC stream may have overflow from the BE traffic window because the TSN schedule is implemented before the Wi-Fi chipsets, so BE traffic overflow into the protected window could happen while data is being transmitted to the wireless channel.

We start by defining two metrics related to the wireless channel utilization, which are the protected window utilization and the channel utilization. The protected window utilization is defined as the ratio between the amount of time used for the TC traffic data transmission in the protected window and the duration of the protected window. The protected window can be designed to be small to increase the protected window utilization, but that could increase the probability of the TC traffic to miss the protected window due to randomness in data processing. The channel utilization is the ratio between the time used for data to be transmitted over the channel to the schedule period.

The probability of overflow to the protected window is the average probability that the BE data can be transmitted over the wireless channel during the protected window time. This could happen because the completion of a BE packet is still using the channel after being transmitted from the Wi-Fi BE transmission queue. Using Wi-Fi priority queues prevents any further packets to be sent over the channel from the BE queue if the the TC Wi-Fi queue has data. Once a EDCA transmission begins, it cannot be preempted by a higher priority queue.

III. Results and Discussion

In this section, we present the numerical results that are obtained through the governing equations in order to assess the performance of the discussed WTSN implementation. We set the BE packet size at 1000 B, the time slot at 0.1 ms, the size of TC packet at 50 B, the schedule period at 8 ms, the protected window size at 0.7 ms, and the wireless channel capacity at 104 Mbps. In order to limit the BE traffic to access the wireless channel ahead of the TC traffic, the BE token bucket data buffer has to be set to allow for a few BE packets with their overhead. Moreover, we set the BE token bucket size at 1500 B and the token rate at 104 Mbps to match the channel capacity. The TC token bucket parameters are set at relaxed values that allow the TC packets to always pass directly to the Wi-Fi chipset.

A. The Impact of BE Token Bucket Buffer Size

We study the impact of the BE token bucket data buffer size on the various performance metrics when the BE traffic rate is 104 Mbps. In Fig. 3, the overflow probability of the BE traffic to the protected window is shown against the token bucket data buffer size and the guard interval length. We observe that allowing more BE packets to be stored in the token bucket data buffer could increase the overflow probability, especially for smaller guard interval lengths.

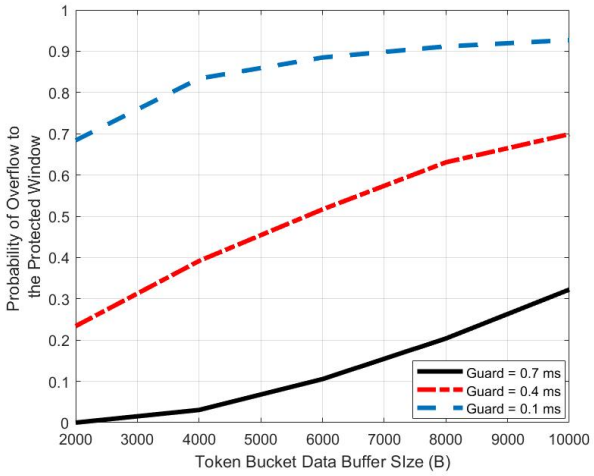

Fig. 3. The overflow probability of the BE traffic to the protected window against the token bucket data buffer size and the guard interval length.

In Fig. 4, the channel utilization is shown against the token bucket data buffer size and the guard window length. We observe that allowing more BE packets to be stored in the token bucket data buffer increases the channel utilization in the case of a high BE traffic rate, because more data are allowed to be buffered and not lost during the TC reserved slots. Similarly, decreasing the guard interval increases the channel utilization because more BE traffic is allowed. However, this enhancement in the channel utilization occurs at the expense of having a higher overflow probability.

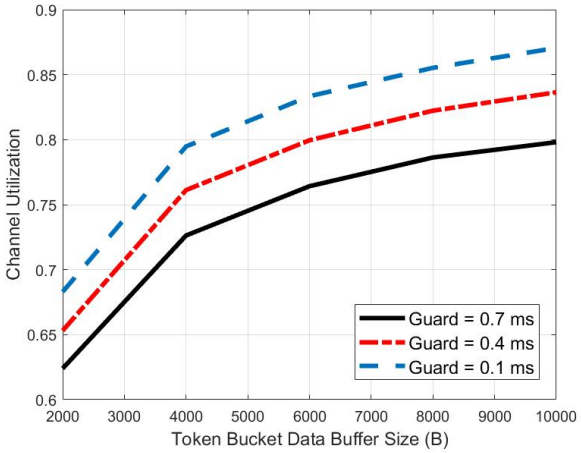

Fig. 4. The channel utilization against the token bucket data buffer size and the guard interval length.

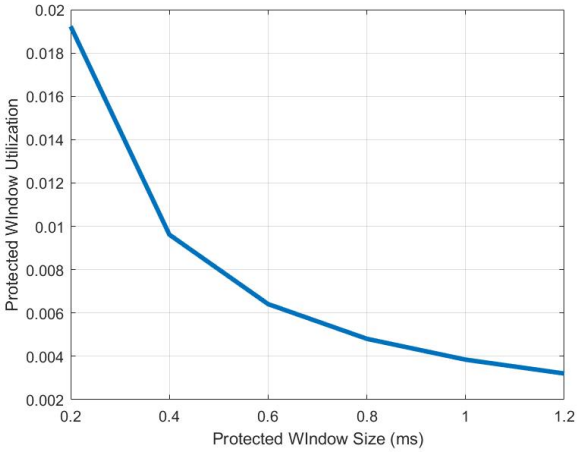

Fig. 5. The change of the protected window utilization against the protected window size.

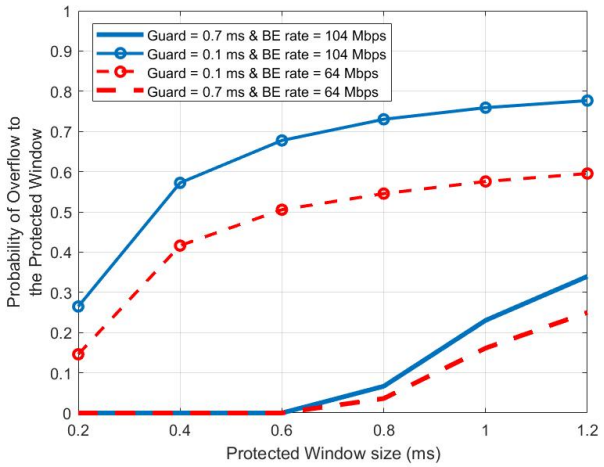

Fig. 6. The overflow probability of the BE traffic to the protected window against the protected window size for values of the guard interval length and BE Traffic rate.

B. The Impact of Protected Window Size

We study the impact of the protected window size on the various performance metrics when the token bucket data buffer size is 2000 B. In Fig. 5, we show the change of the protected window utilization against the protected window size. It is clear that decreasing the protected window size increases the protected window utilization. The protected window utilization is very low because the TC data packet is small but the protected window has to be selected much larger than the transmission time of the packet for two reasons. First, the protected window gives the TC packet the chance to access the channel where there will be randomness on the data generation. Second, it has to allow for the retransmission, if needed. Hence, narrowing the protected window increases the chances that a TC data packet can lose its chance to get to the protected window, where the TC packet can get delayed for a complete schedule period.

In Fig. 6, we show the overflow probability of the BE traffic to the protected window against the protected window size for various values of the guard interval lengths and BE Traffic rates. The impact of the guard interval on the overflow probability is shown earlier. However, the impact of the guard interval becomes less important for lower BE traffic rates because the chance of having buffered data still being transmitted during the guard interval is low. On the other hand, although the overflow probability is zero or very close to zero in the case of smaller protected window sizes at the low BE traffic rate case, it is only beneficial to use smaller protected windows when the jitter in processing the data and the TC traffic generation is comparably low to the protected window size.

In Fig. 7, the channel utilization is shown against the protected window size for values of the guard interval length and BE traffic rate. Basically, the improvement of the channel utilization in this case is due to moving a part of the protected window to the BE traffic window. As a result, the curves are almost linear against the protected window size.

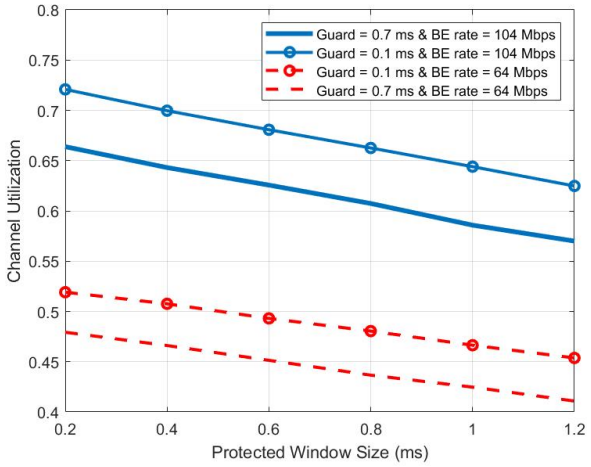

Fig. 7. The channel utilization against the protected window size for values of the guard interval length and BE traffic rate.

C. The Impact of BE Traffic Rate

We study the impact of the BE traffic rate on the various performance metrics when the guard interval is 0.7 ms. In Fig. 8, we observe that the overflow probability is zero for the 2000 B buffer size, and it is generally very low at lower BE traffic rates. However, increasing the BE traffic rate above specific values increases the overflow probability drastically, which is one of the main disadvantages of this software-based TSN implementation as it cannot apply the TSN schedule over the whole network stack. However, this is only true at the high data rates of the BE traffic that can be controlled at the application domain.

In Fig. 8, we show the channel utilization against the BE traffic rate for values of the BE token bucket data buffer size. We observe that the enhancement rate of the channel utilization

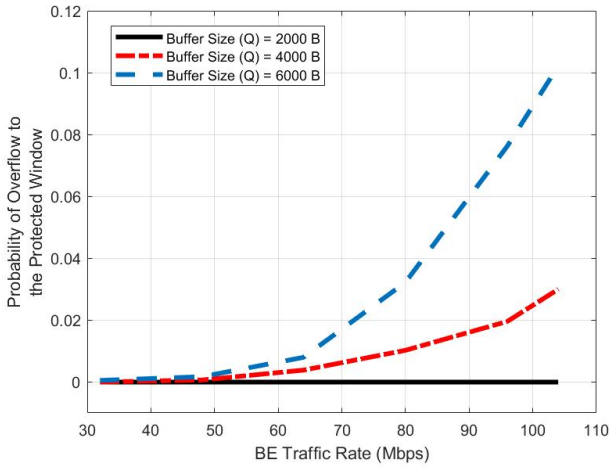

Fig. 8. The overflow probability of the BE traffic to the protected window against the BE traffic rate for values of the BE token bucket data buffer size.

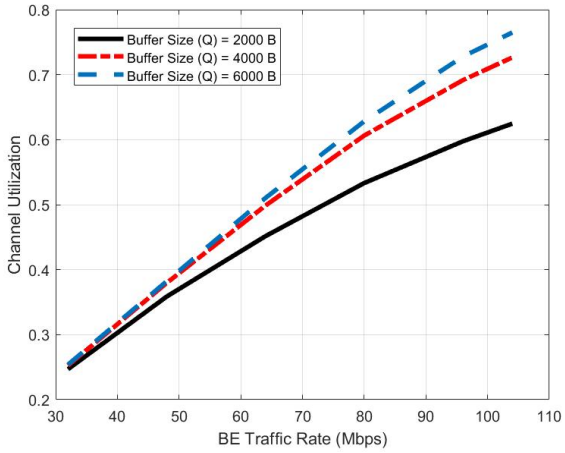

Fig. 9. The channel utilization against the BE traffic rate for values of the BE token bucket data buffer size.

decreases, due to the increase in the token bucket buffer size. As a result, the degradation of the TSN performance due to the increased BE traffic overflow probability cannot be justified by the increase in channel utilization. Hence, selecting a lower token bucket buffer size is more suitable for this TSN implementation.

IV. Conclusions

In this paper, we introduced a numerical analysis of a software-based WTSN implementation that performs software-based TSN scheduling while utilizing the underlying Wi-Fi priority EDCA queues. The scheduling was performed in a Linux Kernel using the Qdisc traffic shaping utility. This implementation is shown to achieve WTSN latency requirements in a practical way, utilizing the existing Wi-Fi chipsets. This provides the advantage of utilizing already deployed legacy Wi-Fi chipsets, but it comes with some limitations. The analysis presented in this paper demonstrates the impact of the Token Bucket algorithm parameters on latency performance. We found that using smaller BE data buffer sizes is recommended to achieve the required latency, but at the expense of channel bandwidth efficiency (wireless channel utilization). Moreover, high data rates of BE traffic can overflow to the TSN protected window, which degrades the TC traffic performance significantly by delaying or dropping TC traffic. Overall, this work identifies the advantages and limitations of the current software-based WTSN implementations and provides a tool for application-based tuning of the Qdisc parameters to achieve a desired level of performance.

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] R. Candell, K. Montgomery, M. Kashef Hany, S. Sudhakaran, and D. Cavalcanti, "Scheduling for Time-Critical Applications Utilizing TCP in Software-Based 802.1Qbv Wireless TSN," in *2023 IEEE 19th International Conference on Factory Communication Systems (WFCS)*, 2023, pp. 1–8.

[2] Y. Liu, M. Kashef, K. B. Lee, L. Benmohamed, and R. Candell, "Wireless Network Design for Emerging IIoT Applications: Reference Framework and Use Cases," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1166–1192, 2019.

[3] T. Stüber, L. Osswald, S. Lindner, and M. Menth, "A Survey of Scheduling Algorithms for the Time-Aware Shaper in Time-Sensitive Networking (TSN)," *IEEE Access*, vol. 11, pp. 61 192–61 233, 2023.

[4] D. Cavalcanti, C. Cordeiro, M. Smith, and A. Regev, "WiFi TSN: Enabling Deterministic Wireless Connectivity over 802.11," *IEEE Communications Standards Magazine*, vol. 6, no. 4, pp. 22–29, 2022.

[5] S. Avallone, P. Imputato, and D. Magrin, "Controlled Channel Access for IEEE 802.11-Based Wireless TSN Networks," *IEEE Internet of Things Magazine*, vol. 6, no. 1, pp. 90–95, 2023.

[6] "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic," *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015)*, pp. 1–57, 2016.

[7] N. Reusch, L. Zhao, S. S. Craciunas, and P. Pop, "Window-Based Schedule Synthesis for Industrial IEEE 802.1Qbv TSN Networks," in *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*, 2020, pp. 1–4.

[8] S. Sudhakaran, K. Montgomery, M. Kashef, D. Cavalcanti, and R. Candell, "Wireless Time Sensitive Networking Impact on an Industrial Collaborative Robotic Workcell," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 10, pp. 7351–7360, 2022.

[9] Ubuntu Manuals, "TAPRIO - time aware priority shaper." [Online]. Available: https://manpages.ubuntu.com/manpages/focal/en/man8/tc-taprio.8.html

[10] E. Zhang and L. Xu, "Capacity and Token Rate Estimation for Networks with Token Bucket Shapers," *Computer Networks*, vol. 88, pp. 1–11, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128615001735

[11] M. Eisen, M. M. Rashid, A. Ribeiro, and D. Cavalcanti, "Scheduling Low Latency Traffic for Wireless Control Systems in 5G Networks," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[12] "IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, pp. 1–1076, 2007.