IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 7, 2021, accepted August 26, 2021, date of publication September 6, 2021, date of current version September 28, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3110585

Critical Offset Optimizations for Overlapping-Based Time-Triggered Windows in Time-Sensitive Network

KHALED M. SHALGHUM[1,2,3], (Student Member, IEEE),
NOR K. NOORDIN[1,3], (Senior Member, IEEE),
ADUWATI SALI[1,3], (Senior Member, IEEE),
AND FAZIRULHISYAM HASHIM[1,3], (Member, IEEE)
[1]Department of Computer and Communication Engineering, Universiti Putra Malaysia, Serdang 43400, Malaysia
[2]Department of Electrical and Electronic Engineering, Azzaytuna University, Tarhuna, Libya
[3]Wireless and Photonic Networks Research Centre of Excellence (WiPNet), Faculty of Engineering, Universiti Putra Malaysia, Serdang 43400, Malaysia

Corresponding authors: Khaled M. Shalghum (kshalghum@gmail.com) and Nor K. Noordin (nknordin@upm.edu.my)

This work was supported by UPM Putra Berimpak under Fundamental Research Grant 9584300.

ABSTRACT Deterministic and low latency communications are increasingly becoming essential requirements for several safety-critical applications, such as automotive and automation industries. Time-sensitive networking (TSN) is an element of the new IEEE 802.1 standards that introduced Ethernet-based amendments to support these applications. One of these enhancements was presented in IEEE 802.1Qbv to define the time-aware shaping (TAS) technique for time-triggered (TT) traffic scheduling. The TAS mechanism is window-based scheduling using a gating system controlled by the gate control list (GCL) schedules in all nodes. Although several scheduling algorithms have been proposed to investigate the effects of window-related parameters on network performance, the offset difference (OD) between the same-class windows in the adjoining nodes has not been optimized yet. These optimizations are extremely crucial to implement less pessimistic latency schedules. This paper proposes an optimized flexible window-overlapping scheduling (OFWOS) algorithm that optimizes TT window offsets based on latency evaluations considering the overlapping between different priority windows at the same node. First, we formulate the GCL timings as mathematical forms under variable ODs between the same-priority windows. Then, an analytical model is implemented using the network calculus (NC) approach to express the worst-case end-to-end delay (WCD) for TT flows and evaluated using a realistic vehicular use case. The OFWOS model optimizes OD under all overlapping situations between TT windows at the same node. Compared with the latest works of 3-hop and 30-hop TSN connections, the OFWOS reduces the WCD bounds by 8.4% and 32.6%, respectively, accomplishing less pessimistic end-to-end latency bounds.

INDEX TERMS Network calculus, safety-critical real-time systems, scheduling algorithm, time-sensitive network (TSN), worst-case latency performance.

I. INTRODUCTION

Safety-critical real-time applications such as in the automotive and automation industries require deterministic and low latency performance for end-to-end data transmissions. Failing to comply with these requirements may cause dangerous situations for society. Therefore, many technologies have been proposed to support these applications. One of which is the Ethernet network, as it has enough bandwidth and feasible cost for real-time scenarios. Although multiple Ethernet-based protocols have been previously introduced, such as Audio/Video Bridging (AVB) Ethernet and time-triggered (TT) Ethernet, they cannot manage safety-critical transmissions and achieve the requirements.

As an extension to TT-Ethernet protocol is the time-sensitive networking (TSN) standardized by IEEE TSN task group which was introduced to support safety-critical environments. The TSN features include timing, network

The associate editor coordinating the review of this manuscript and approving it for publication was Chakchai So-In.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

management, access control, and reliability aspects to support TT flows targeting deterministic and low latency, extremely low jitter, and no congestion loss [1]. In the presence of TT traffic, the TSN framework is designed to serve AVB traffic with lower QoS requirements and Best Effort (BE) flows with no QoS guarantees. These extensions interested many experts and companies to espouse the TSN technology.

As defined in the IEEE 802.1Qbv standard [2], the TSN framework integrates TT flows using a time-aware shaping (TAS) technique which operates as a time-gating mechanism controlled by the gate control list (GCL) scheduled in each networking node. These predefined schedules (GCLs) control accessing TT flows through the physical links with a global synchronization constraint. For unscheduled traffic (AVB and BE) configurations, the TSN switching applies the credit-based shaping (CBS) technique based on TT schedules, as introduced in IEEE 802.1Qav [3].

Designing appropriate GCLs in all selected nodes while guaranteeing the quality-of-service (QoS) requirements for critical time flows is a complex and vital problem. The designer has to pay the highest attention to two significant aspects; the TT latency requirements and the impact on unscheduled real-time traffic [4]. Although several safe analytical models have been proposed to speed TT scheduling, they still have a degree of pessimism in latency evaluations. Thus, decreasing the experienced end-to-end latency for urgent data under a safe formal analysis is an attractive goal in the TSN scheduling design.

As the TAS mechanism is a window-based scheduling technique, numerous works have considered the transmission window specifications and their impacts on the overall network performances [5]–[8]. Under fixed priority traffic assumption, window offsets will undoubtedly affect the end-to-end delay for the associated queues. Although the proposed algorithms in [7], [8] have proved the impact of window offsets on the worst-case latency bounds, no one has optimized the offset difference between the same priority windows in the adjoining nodes. Furthermore, the GCL schedules are offline based implemented, thus it is fundamental to evaluate and optimize the offset difference between the same priority transmission windows. More details will also be discussed in Section II later.

The trusted and safe latency representations are analytically based as all worst-case situations can be included through the design stage. Thus, analytical models are pivotal in safety-critical real-time systems. One of the preferred and trusted analytical approaches in real-time systems is the Network Calculus (NC) [9], as it produces safe and less pessimistic latency evaluations [10], [11]. For this reason, the NC approach is adopted here to formulate the proposed model.

In this paper, we design a GCL pattern for all selected nodes on the end-to-end transmission path by considering the offset difference (*OD*) between the same priority windows in the adjacent nodes. Based on the proposed algorithm in [8], it is assumed that the TT open windows overlap by a ratio (*OR*) in each selected node. Accordingly, the *OD* and *OR* values are considered as design factors in GCL implementations. Under these assumptions, a closed-form expression is determined for the worst-case end-to-end delay (*WCD*) bound for a specific TT priority queue. The *OD* effect is examined and evaluated comprehensively considering all expected window situations. These evaluations can be considered as a helpful guide for TSN designers to implement tighter and more trusted GCL schedules. Thus, the paper contributions can be summarized as:

- An optimized flexible window-overlapping scheduling (OFWOS) algorithm optimizes the *OD* between the same priority windows in the adjoining nodes is proposed. The GCL schedules are generated under overlapping-based TT windows in the same selected node.
- Worst-case latency bound is formulated for a targeted TT queue using Network Calculus (NC) approach. The OFWOS model is evaluated using a realistic vehicle use case. Under non-overlapping-based windows, the preferred *OD* range is determined according to the end-to-end latency performance. Critical *OD* optimizations are determined and discussed by considering all overlapping situations between TT windows.
- The OFWOS algorithm achieves less pessimistic *WCD* bounds compared with the previous related works under all window-overlapping scenarios. Accordingly, the OFWOS performance saves more bandwidth for unscheduled transmissions under overlapping-based GCLs.

The remainder of the paper is structured as follows. Section II introduces related works, and Section III presents a relevant background on TSN switching fundamentals and the network calculus theory for worst-case latency computations. The OFWOS system model and associated design decisions are described in Section IV. The worst-case end-to-end latency analysis for the OFWOS algorithm and related performance evaluations are presented with critical discussions in Sections V and VI, respectively. Finally, Section VII concludes the paper.

II. RELATED WORKS

The IEEE 802.1 TSN task group has standardized the TAS mechanism to serve the TT streams and the CBS technique for the unscheduled traffic (AVB and BE) in IEEE 802.1Qbv [2] and IEEE 802.1Qav [3], respectively. The scheduling synthesis is considered as an NP-complete (non-deterministic polynomial-time) problem [12]. For this reason, the TAS-based schedules are proposed as offline implementations, where the timing alignments are statically configured in GCLs to guarantee deterministic frame transmission. Under fully synchronized TSN elements, several TAS-based scheduling models have been presented to improve the schedulability and latency performances for real-time traffic.

Several models have proposed a no-wait scheduling approach for TT forwarding. In [13], Atallah *et al.* implemented the no-wait scheduling model using an iterated integer linear programmable technique by forwarding incoming traffic through multiple disjoint transmission paths. After addressing several scheduling-based solutions, the authors improved the success rate from 47% to 90% compared with the random flows partitioning technique. Jin *et al.* [14] proposed a joint algorithm of appropriate message fragmentation and no-wait scheduling to increase schedulability up to 50% further than previous algorithms. In [15], Dürr *et al.* presented a Tabu search algorithm to compute efficient no-wait schedules with less number of guard bands. The guard bands were reduced by 24% on average, resulting in less bandwidth wastage.

Fixed GCLs in all network elements produce fluctuated performances for scheduled and unscheduled traffic. For this reason, a degree of adaptability in performance-related parameters was proposed. For example, to ensure latency deadlines for critical time flows, Nasrallah *et al.* [16] implemented GCL schedules with adaptive window durations for scheduled and unscheduled transmissions, depending on related latency threshold and the instantaneous loading condition. The simulation results confirmed that increasing window length decreases queuing delays for the corresponding traffic type and vice versa. Moreover, they compared the performances of TAS and asynchronous time shaping (ATS) techniques under different loading conditions. However, the authors assigned only two priority queues at the egress port without differentiating hard and soft real-time transmissions when using TAS. Kim *et al.* [17] assigned a timeslot as a protection interval for emergency transmissions in each cycle. The simulation findings guaranteed targeted services for emergency data with minimal effect on the scheduled traffic. The authors in [18], [19] recommended adapting GCL-timings according to the network resources availability and instantaneous traffic intensity, while [20], [20] proposed a scheduling algorithm with online updating to integrate incoming flows efficiently. [13], [21], [22] recommended joint routing and scheduling algorithms targeting proper network resources and proper load balancing. However, increasing the GCL adaptability requires an excellent control system for the overall network elements. Additionally, most recent works reported in [23]–[25] recommended the software-defined network (SDN) protocol to be used to manage TSN scheduling updates. However, the reported works are implemented using simulation approaches, which cannot include all worst-case transmission cases leading to unsafe performance evaluations.

The argument surrounding these aforementioned approaches is that prioritizing TT traffic by designing GCLs with tough timing constraints will degrade the performance of unscheduled real-time traffic. In particular, the AVB flows may lose the QoS requirements. For this reason, implementing appropriate GCL designs that guarantee TT requirements with minimal impact on AVB traffic is an essential and critical issue. Gavrilut *et al.* [26], [27] proposed to include AVB traffic with TT flows in the schedule. The authors used a greedy randomized adaptive search procedure (GRASP) to implement the algorithm and confirmed feasible AVB scheduling under a simple network scenario. However, the presented simulation could not guarantee feasible schedules for AVB under high traffic intensity and under more complicated conditions use cases. Zhang *et al.* [28] reduced the effect of higher priority flows on the lower priority traffic by selecting an appropriate transmission cycle and proper scheduling unit. On the other hand, the researchers in [4], [29]–[31] presented formal AVB latency analysis based on predefined TT schedules. The authors demonstrated some improvements in the CBS limitations. However, all these algorithms have been implemented under complete isolation between TT windows, resulting in more bandwidth wastage.

Window-related metrics were also considered in the scheduling designs and examined in several works. In [32], Craciunas *et al.* confirmed that the transmission window must be fit within the framing period and completely isolated in serving incoming TT frames. They emphasized that the least window duration must be greater than the frame offset plus the frame size to ensure successful transmission. In [12], the authors measured the schedulability and scalability using the latency deadlines and the window lengths for scheduled streams.

Other window-based scheduling algorithms were presented in [5]–[8], [33], [34]. In [33], [34], the authors implemented their scheduling algorithms with no synchronization requirement for end systems. However, the prerequisite is that all non-terminal nodes must be synchronized with each other. In [5], Zhao *et al.* suggested allowing TT windows to overlap, leading to improving the overall solution space. In [6], Zhang *et al.* used the analytical model in [5] to evaluate the preemption effect on the worst-case TT latency performance. Although the presented model in [5] and [6] has enhanced the solution space compared with full-isolated windows, the TT window overlapping was not optimized with the targeted latency deadlines. Moreover, the algorithms were implemented with node-based design without considering the offset difference between the same priority windows in the adjoining nodes [5], [6], [33], [34]. Zhao *et al.* [7] proposed an analytical model based on completely isolated TT windows by considering the offset difference between the same class windows in adjoining nodes. Also, after achieving a critical latency improvement compared to [7], Shalghum *et al.* [8] optimized the overlap between TT windows based on three aspects; the priority of the overlapped queues, the position of the overlap, and the overlapping ratio. Although the proposed models in [7] and [8] have reduced the worst-case TT latency compared to the models in [5], [6], the window offsets were statically selected without critical optimization with related network metrics and latency behavior. The algorithms in [7] and [8] also did not confirm that the chosen window offsets meet the lowest worst-case end-to-end latency bound. Thus, a critical optimization is required to evaluate the whole

TABLE 1. Design-Based differentiation between relevant studies and the proposed model with the associated achievements.

Reference	Shaping technique	Latency evaluation	Latency evaluation approach	Worst-case considerations	Window overlapping	Relative Offsets between Nodes	Key window-based achievements
[12], [32]	TAS	No	-	Yes	Not considered	Not considered	Deriving window constraints
[33]	TAS	Yes	Simulation	No	Not considered	Not considered	Synchronization improvement
[34]	TAS	Yes	Analytical	Yes	Not optimized	Not considered	No need for window isolation
[16]	TAS/ ATS	Yes	Simulation	No	Not considered	Not considered	Latency improvement with an adaptive windows size
[5]	TAS	Yes	Analytical	Yes	Not optimized	Not considered	Enhanced solution space
[6]	TAS	Yes	Analytical	Yes	Not optimized	Not considered	Preemption improvement
[7]	TAS	Yes	Analytical	Yes	Not considered	Not optimized	Latency reduction compared to [5] and [6]
[8]	TAS	Yes	Analytical	Yes	Optimized	Not optimized	Less WCD compared to [7]
Proposed model	TAS	Yes	Analytical	Yes	Optimized	Optimized	Less WCD and more overlapping flexibility than [8]

possible range of offset difference between the same priority windows on the adjacent nodes. As a summary of the most related previous works to our work, we briefly compare their assumptions, limitations, and key achievements in Table 1.

III. RELEVANT BACKGROUND

This section presents the fundamentals in TSN switching transmission and the NC approach implemented in our model.

A. TSN SWITCHING FUNDAMENTAL

The TSN structure consists of a number of nodes (V) connected by full-duplex data links. The set of nodes include end systems (ESs) (sources and destinations) and switches (SWs). The TSN task group has standardized the stream reservation protocol (SRP) [35] to manage and automatically establish appropriate connections between targeted ESs. Data forwarding between TSN nodes is done using the TAS mechanism introduced in IEEE 802.1Qbv protocol. In the TAS technique, each switch differentiates the incoming flows to eight priority queues as depicted in Fig. 1, with different timing constraints provided in the GCL. The GCL specifies the opening and closing times for the associated queue gate. Thus, the transmission mechanism utilizes a window-based technique under guaranteed synchronization requirement between nodes. If the queue-gate is closed, the associated frames are not allowed to be transmitted. Only one priority frame from the associated queue can be sent using the first-in-first-out (FIFO) mechanism. TT queues prioritize being transmitted before unscheduled traffic (AVB and BE) if their gates are opened simultaneously. But for AVB and BE queues, the CBS technique is used to share the available bandwidth.

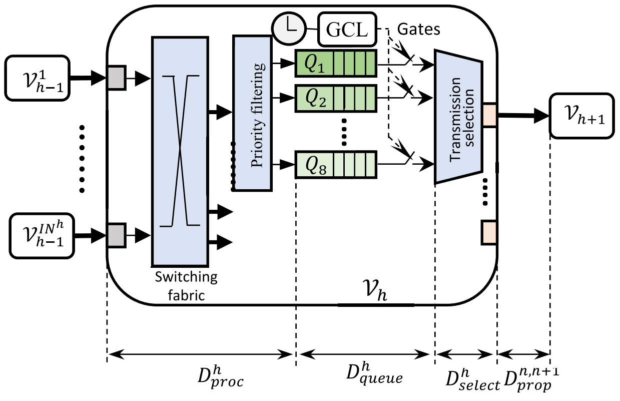


FIGURE 1. An IEEE 802.1Qbv capable switch with multiple input and output ports, with delay components between node-to-node connection.

The IEEE 802.1Qbv switch realizes several functions simultaneously for traffic arriving from the input ports (IN^h), including switching fabric, filtering, and traffic policy selection, as illustrated in Fig. 1. To achieve these functions, the frame experiences the processing delay (D_{proc}^{h}), queuing delay (D_{queue}^{h}), selection delay (D_{select}^{h}), and the propagation delay ($D_{prop}^{h,h+1}$) between nodes h and $h+1$, as presented in Fig. 1.

B. NETWORK CALCULUS BASICS

The NC approach is an effective tool to express the flow transmission properties in networking communications. It has been used widely to evaluate several important QoS parameters in real-time systems, such as worst-case latency bounds and network utilization. In this article, we use it to formulate worst-case end-to-end latency for TT traffic in TSN environment. To achieve that, two curves must be determined, the arrival curve and the service curve, to specify flow

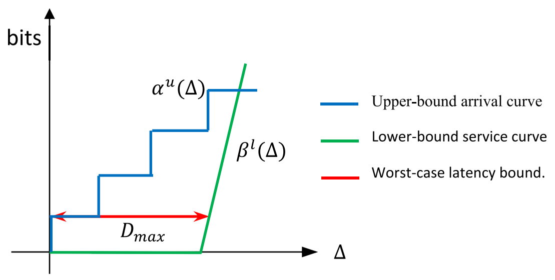


FIGURE 2. The worst-case latency bound using network calculus approach.

characteristics and network resources availability. The min-plus formulations are used to express these curves.

The arrival curve ($\alpha(t)$) specifies the stream arrival process ($R(t)$) by counting the input bits in a node until t, as follows [36],

$$R(t) \le \inf_{0\le\lambda\le t} \{R(\lambda) + \alpha(t-\lambda)\} = (R \otimes \alpha)(t) \quad \forall\lambda < t$$

where *inf* means infimum (maximum lower bound) and $\otimes$ represents the convolution of min-plus operation. The arrival curve is formulated using the triple $\langle p, j, d\rangle$, where p represents the transmission period of input flows, j is the jitter, and d denotes the minimum inter-arrival distance of frames in the associated flow. Accordingly, the upper- and lower-bounds of the arrival curve can be given as follows [36],

$$\alpha^l(\Delta) = \left\lfloor \frac{\Delta - j}{p} \right\rfloor \tag{1}$$

$$\alpha^u(\Delta) = \min\left\{ \left\lceil \frac{\Delta + j}{p} \right\rceil, \left\lceil \frac{\Delta}{d} \right\rceil \right\} \tag{2}$$

These bounds mean that during any interval (Δ), at least $\alpha^l(\Delta)$ and at most $\alpha^u(\Delta)$ stream events will arrive at the associated node. Hence, the upper-bound arrival curve is considered for worst-case latency calculations.

The service curve ($\beta(t)$) specifies the resources serving availability as a departure process ($R^*(t)$) by counting the bits outgoing from a node until t, as follows [36],

$$R^*(t) \le \inf_{0\le\lambda\le t} \{R(\lambda) + \beta(t-\lambda)\} = (R \otimes \beta)(t) \quad \forall\lambda < t$$

The incoming flows will be served depending on their rate (C) and the duration of open window intervals assigned for them. Accordingly, depending on the arrival instances at least $\beta^l(t)$ and at most $\beta^u(t)$ bits will be served at the associated node. However, the lower-bound service curve ($\beta^l(t)$) is considered to calculate worst-case latency.

Based on the above equations and as shown in Fig. 2, the worst-case latency bound can be computed in each node as the maximum horizontal distance (D_{max}) between the upper-bound arrival curve ($\alpha^u(t)$) and the lower-bound service curve ($\beta^l(t)$) [36], as follows,

$$D_{max} \overset{\text{def}}{=} Del\left(\alpha^u, \beta^l\right) = \sup_{\lambda\ge0}\left\{\inf\left\{\tau \ge 0 : \alpha^u(\lambda) \le \beta^l(\lambda+\tau)\right\}\right\} \tag{3}$$

IV. OFWOS SYSTEM MODEL AND DESIGN DECISIONS

TSN switching uses two priority-based traffic isolation mechanisms, namely spatial and temporal isolations, to distinguish between ingress flows. The spatial isolation is provided in each node by forwarding the incoming frames to one of eight priority queues with different gate-timing constraints at the output port. The temporal isolation is provided in the traffic schedules, i.e., GCLs. The GCL specifies the open and close times for each queue-gate, creating a window-based transmission mechanism.

Window specifications in the GCL pattern for the associated TT queue will undoubtedly affect the overall transmission performance. These specifications include the window duration, the gate events' period, the overlapping with other TT windows, and the window offsets. All these timings are offline-based design parameters in the TAS technique. Adjusting them could enhance or degrade the system performance, especially the end-to-end latency and the bandwidth availability for the associated queue.

In this work, we are interested in assessing the performance of the TAS technique under an optimized flexible window-overlapping scheduling algorithm. The OFWOS model optimizes the *OD* between same-priority windows in the adjacent nodes. The optimization process is provided for both scenarios, the overlapped and non-overlapped TT windows in the same node. In specific, the OFWOS algorithm works as a predesign stage that optimizes *OD* for each overlapping situation. Then, based on the worst-case latency performance and the targeted latency deadlines, the optimized *OS* and *OR* are selected to implement proper GCLs. The overlapping is considered under three relevant metrics: the priority, the position, and the ratio (*OR*). To achieve these objectives, OFWOS performs two main design stages, as illustrated in Fig. 3. In the first stage, as the best latency performance will be obtained under complete isolation between TT windows, end-to-end GCLs are implemented under non-overlapping windows and evaluated to determine the preferable *OD* range where the frame experiences the lowest worst-case latency. Secondly, under the preferable *OD* range, all overlapping situations are considered and evaluated to determine the optimal *OD*s accordingly. These two stages are explained below and illustrated in steps, as shown in Fig. 3.

Hence, latency calculations considered here are only for TT flows ($\mathcal{S}$). Each TT flow ($s_m \in \mathcal{S}$) consists of $\mathcal{F}_m$ frames, where m is the priority of the flow. When the frame arrives at node h, it will be selected to one of N_q^h TT queues ($Q_1^h, \ldots, Q_{N_q}^h$), where Q_1^h represents the highest priority queue, while $Q_{N_q}^h$ the lowest. The Q_m^h-frame forwarding to the egress port is controlled by a G_m^h gate which changes with a period T_m^h initializing open and closed events with lengths of W_m^h and $T_m^h - W_m^h$, respectively.

To realize the generality in the performance evaluation, we consider the kth queue in a node h (Q_k^h) as a targeted queue, where $1 \le k \le N_q^h$. As *OD* is the key factor of the

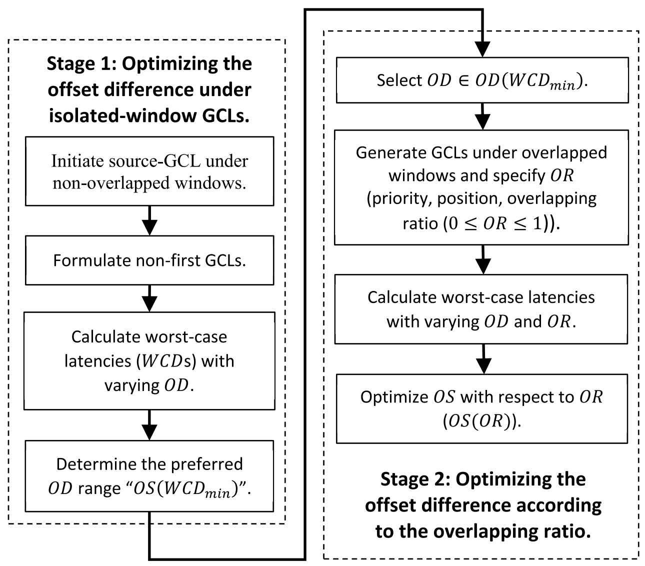

Stage 1: Optimizing the offset difference under isolated-window GCLs.
Initiate source-GCL under non-overlapped windows.
Formulate non-first GCLs.
Calculate worst-case latencies (WCDs) with varying OD.
Determine the preferred OD range "OS(WCD_min)".
Select OD ∈ OD(WCD_min).
Generate GCLs under overlapped windows and specify OR (priority, position, overlapping ratio (0 ≤ OR ≤ 1)).
Calculate worst-case latencies with varying OD and OR.
Optimize OS with respect to OR (OS(OR)).
Stage 2: Optimizing the offset difference according to the overlapping ratio.

FIGURE 3. The two main stages of the optimized flexible window-overlapping scheduling (OFWOS) algorithm.

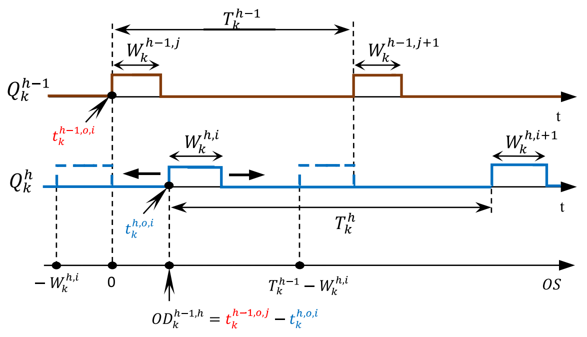


FIGURE 4. The variation range of the offset difference (*OD*) between the same-priority windows in the adjoining nodes.

proposed model, the whole range of *OD* must be considered. Accordingly, the *OD* between Q_k^{h-1} and Q_k^h windows will vary from $W_k^{h,i}$ to $T_k^{h-1} - W_k^{h,i}$ as shown in Fig. 4, to cover all situations where the i^{th} Q_k^h window may serve the flows that come from the j^{th} Q_k^{h-1} window. But this assumption is valid only if the transmission period of Q_k^{h-1} is not greater than Q_k^h period ($T_k^{h-1} < T_k^h$). However, if $T_k^{h-1} \geq T_k^h$, *OD* will vary from $W_k^{h,i}$ to $T_k^h - W_k^{h,i}$. Thus, the *OD* variation can be conditionally specified as,

$$OD_k^{h-1,h} \in \begin{cases} \left[-W_k^{h,i}, T_k^{h-1} - W_k^{h,i}\right] & \textit{if } T_k^{h-1} \leq T_k^h \\ \left[-W_k^{h,i}, T_k^h - W_k^{h,i}\right] & \textit{if } T_k^{h-1} > T_k^h \end{cases} \tag{4}$$

Next, the algorithm considers all overlapping situations between TT windows in each node *h*. The Q_k^h window may overlap with any Q_m^h window, as shown in Fig. 5. In the *i*-th cycle, the *OR* between Q_k^h and Q_m^h windows represents the time interval ($L_{k,m}^{h,i}$), when the Q_k^h and Q_m^h gates are simultaneously open, divided by the total duration of the Q_k^h open

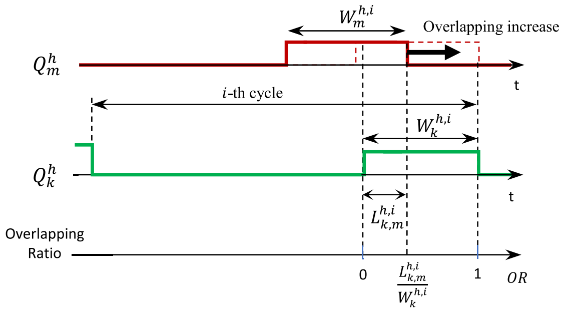


FIGURE 5. Overlapping scenario between different priority TT windows at the same node.

window ($W_k^{h,i}$), as follows [8],

$$OR_{k,m}^{h,i} = \frac{L_{k,m}^{h,i}}{W_k^{h,i}} \tag{5}$$

As presented in Fig. 5, the *OR* is adjusted from 0 to 1 to cover all the expected overlapping conditions. Note that "zero" overlapping means that the Q_k^h window is completely isolated, i.e., when the Q_k^h gate (G_k^h) is open, all other TT gates are closed, and "one" means that the window is completely overlapped, i.e., there is at least another TT gate is open when G_k^h is open, as clearly represented in Fig. 5.

Based on the above window considerations, we implement here the end-to-end GCLs, which will be used to formulate the worst-case end-to-end latency for the Q_k^h frames. Note that our GCL implementation is offline based on worst-case analysis, and the dynamism is not considered.

V. WORST-CASE LATENCY ANALYSIS OF THE OFWOS MODEL

In this section, we calculate the worst-case end-to-end delay (*WCD*) for the Q_k^h frames using the network calculus approach. The timing calculations are provided based on the scheduling constraints for each TT queue. These constraints are implemented in the GCL structure of each node. Accordingly, the required GCL schedules for all nodes in the selected path are formulated in the beginning. Then, as the worst-case analysis only considered the contention-free windows for data transmission, these windows are determined based on the updated GCLs. The contention-free intervals equal the related window durations, excluding the guard bands (block intervals) and the contention-based intervals (overlapping effect). The frame arrival effect must then be considered to formulate the experienced maximum waiting time and the arrival curve. The service curve is determined based on contention-free intervals and the maximum waiting time. Finally, the arrival and service curves are used to calculate *WCD* for the selected path. Note that latency calculations provided here considered both non-overlapping and overlapping-based GCLs at the same time. As illustrated in Fig. 3, the implemented forms will be used to calculate *WCD*s in the two design stages of the OFWOS algorithm. These analytical steps are presented below.

A. GCL FORMULATION

The GCL schedule determines the open and closed intervals for all queue gates. The OFWOS model implements the GCL as mathematical relationships for each node depending on the GCL of the previous node on the same selected path. In particular, the OFWOS initiates the source-GCL and, then, all other GCLs will be generated accordingly. The source-GCL is initialized by specifying the opening times for all priority gates ($t_m^{FN,o,1}$), where m represents a queue-priority in the source that has N_q^{FN} queues. Accordingly, the i^{th} opening and closing times for the first node will be easily formulated as,

$$\begin{aligned} &\forall 1 \le m \le N_q^{FN} \\ &t_m^{FN,o,i} = t_m^{FN,o,1} + (i-1)\,T_m^{FN} \\ &t_m^{FN,c,i} = t_m^{FN,o,1} + W_m^{FN} + (i-1)\,T_m^{FN} \end{aligned} \tag{6}$$

where T_m^{FN} denotes the Q_m^{FN} period, and W_m^{FN} the length of Q_m^{FN} open window. Additionally, the i^{th} opening and closing times for a non-first node h will be represented as,

$$\begin{aligned} &\forall 2 \le h < N \quad \& \quad \forall 1 \le m \le N_q^h \\ &t_m^{h,o,i} = t_m^{FN,o,1} + (i-1)\,T_m^h + \sum\nolimits_{j=2}^{h} OD_m^{j-1,j} \\ &t_m^{h,c,i} = t_m^{FN,o,1} + W_m^h + (i-1)\,T_m^h + \sum\nolimits_{j=2}^{h} OD_m^{j-1,j} \end{aligned} \tag{7}$$

where T_m^h and W_m^h represent the period and window length of Q_m^h, respectively. $OD_m^{j-1,j}$ denotes the offset difference between the initial m^{th} windows in nodes $j-1$ and j, and N is the number of nodes on the selected path from the source to the destination. Based on these parameters, the end-to-end GCLs will be specified. After implementing the path schedule using (6) and (7), the OFWOS algorithm calculates the worst-case end-to-end delay (WCD) for the Q_k^h frames using the network calculus (NC) approach.

B. CONTENTION-FREE INTERVAL CALCULATION

As mentioned, the contention-based and guard band intervals must be defined to determine the contention-free interval in each Q_k^h open window. The contention-based intervals consider the overlapping effects with other TT windows, and the guard band is located at the end of each TT window to ensure that the remaining time is enough to transmit the associated frame. All mathematical calculations related to the effects of these intervals are described in Appendix A.

C. MAXIMUM WAITING TIME CALCULATION

The waiting time is defined as the time the frame experiences in a node from its arrival instance at the associated queue until the starting time of the first contention-free interval. The maximum waiting time considers the worst-case frame arrival, whereby in the first node, the Q_k^h frame arrival instance is arbitrary during the related period (T_k^{FN}), while in the non-first node, the arrival instance is bounded by the contention-free interval for the corresponding priority queue in the previous node.

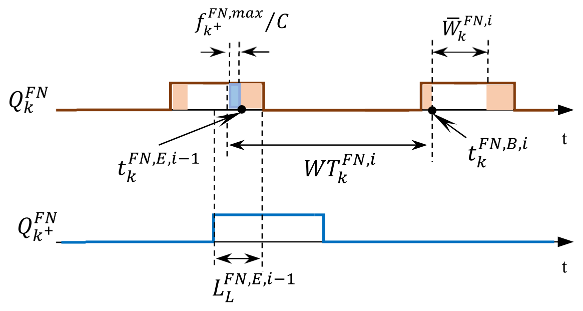


FIGURE 6. The maximum waiting time at the first node.

Fig. 6 illustrates the maximum waiting time in the first node that happens when the first Q_k^{FN} frame reaches the queue at the ending time of the contention-free interval. The most pessimistic arrival instance happens when a lower-priority ($Q_{k^+}^{FN}$) frame is in the transmission condition at the end of the contention-free interval. The non-preemption mechanism prevents any Q_k^{FN} frame to interrupt $Q_{k^+}^{FN}$ frame. After finishing the lower-priority transmission, the remaining time is not enough for the maximum size of Q_k^{FN} frame. Accordingly, the Q_k^{FN} and $Q_{k^+}^{FN}$ overlapping maximizes the waiting time if the overlap in the previous window ($L_L^{FN,E,i-1}$) is greater than the guard band interval ($f_k^{FN,max} \big/ C_{FN}$). Therefore, by taking the i^{th} contention-free window as a benchmark, the maximum waiting time can be given as [8],

$$WT_k^{FN,i} = t_k^{FN,B,i} - \left(t_k^{FN,E,i-1} - D_L^{FN,np,0}\right) \tag{8}$$

where

$$\forall t_m^{FN,o,i-1} < t_k^{FN,c,i-1} < t_m^{FN,c,i-1}$$

$$D_L^{FN,np,0} = \max_{k<k^+\le N_q}\left\{ min\left\{ \frac{f_{k^+}^{FN,max}}{C_{FN}}, L_L^{FN,E,i-1} \right\} \right\}$$

Otherwise $D_L^{FN,np,0} = 0$, and

$$L_L^{FN,E,i-1} = W_k^{FN,i-1} \cdot \max_{k<k^+\le N_q}\left\{ OR_{k,k^+}^{FN,E,i-1} \right\}$$

For a non-first node h, the frame arrival is bounded by the contention-free intervals for all preceding nodes connected to h ingress ports. At any arrival instance, the frame cannot be out of those intervals. As illustrated in Fig. 7, the earliest Q_k^h frame ingresses node$h$ at "$t_{k,in}^{h,i}$" and reaches the associated queue in the i^{th} cycle at $t_{k,queue}^{h,i}$, which can be given as,

$$\begin{aligned} t_{k,queue}^{h,i} &= t_{k,in}^{h,i} + D_{proc}^h \\ &= t_k^{h-1,B,i} + D_{k,select}^{h-1} + D_{prop}^{h-1,h} + D_{proc}^h \end{aligned} \tag{9}$$

where $t_k^{h-1,B,i}$ is the starting time of the i^{th} Q_k^{h-1} contention-free interval in node $h-1$, $D_{prop}^{h-1,h}$ is the propagation delay between $h-1$ and h, $D_{k,select}^{h-1} = f_k^{h-1,max}/C_{h-1}$ represents the time required to select the entire Q_k^{h-1} frame through the

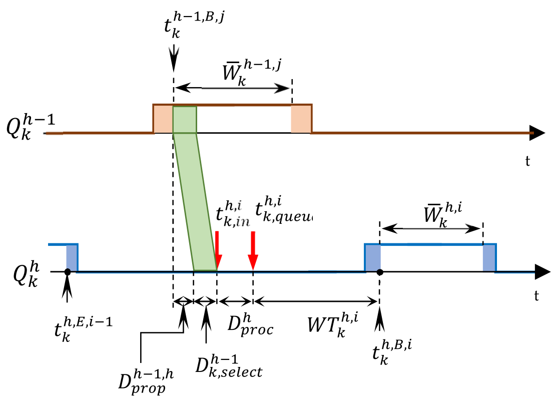


FIGURE 7. **The maximum waiting time at the non-first node in the selected path when the frame arrival is out of the associated contention-free interval.**

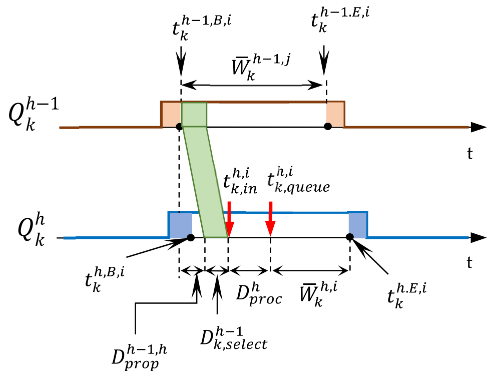


FIGURE 8. **The maximum waiting time at the non-first node in the selected path when the frame arrival is in the associated contention-free interval.**

link, and D_{proc}^h represents the processing delay required to reach the associated queue in node h after passing the input buffer, switching fabric, and priority filtering, as presented in Figs. 7 and 8.

As illustrated in Figs. 7 and 8, the waiting time highly depends on the offset difference between the same-priority windows in the adjacent nodes $h-1$ and h ($OD_k^{h-1,h}$). Thus, the maximum waiting time can be expressed depending on the time when the frame reaches the queue, as follows,

$$WT_k^{h,i} = \begin{cases} t_k^{h,B,i} - t_{k,queue}^h, & t_k^{h,E,i-1} < t_{k,queue}^h < t_k^{h,B,i} \\ 0, & t_k^{h,B,i} \le t_{k,queue}^h \le t_k^{h,E,i} \end{cases} \tag{10}$$

In addition, the first Q_k^h frame cannot be served before $t_{k,queue}^h$, therefore, the first contention-free interval will start at that time. It is worth noting that even if $t_k^{h,E,i} > t_k^{h,B,i}$, the length of the first Q_k^h contention-free interval in the non-first node may have more limitations depending on the frame arrival, as follows,

$$\bar{W}_k^{h,i} = \begin{cases} t_k^{h,E,i} - t_k^{h,B,i}, & t_k^{h,E,i-1} \le t_{k,queue}^h \le t_k^{h,B,i} \\ t_k^{h,E,i} - t_{k,queue}^h, & t_k^{h,B,i} < t_{k,queue}^h < t_k^{h,E,i} \end{cases} \tag{11}$$

D. WORST-CASE END-TO-END LATENCY FOR Q_k TRAFFIC

In the network calculus (NC) approach, the upper-bound arrival and lower-bound service curves are used to calculate worst-case end-to-end latency. In our algorithm, both two bounds must be defined for each node h in the selected path, i.e., $\alpha_k^{h,u}(t)$ and $\beta_k^{h,l}(t)$.

1) LOWER-BOUND SERVICE CURVE "$\beta_k^{h,l}(t)$" DETERMINATION

As illustrated before, under the worst-case transmission situation, the Q_k^h traffic obtains the service only at the contention-free intervals in the selected nodes. The duration of contention-free intervals changes in each cycle according to the overlapping situation with other TT windows. However, the associated pattern is repeated after the hyper-period, which equals the least common multiple of periods for all TT queues ($T_{GCL}^h = \underset{1\le m\le N_q}{LCM}\left(T_m^h\right)$). Thus, in each hyper-period, there is an M contention-free intervals ($M = T_{GCL}^h / T_k^h$). By considering the i^{th} cycle as a benchmark, the lower bound of the service curve can be given as [6],

$$\beta_k^{h,l,i}(t) = \sum_{j=i}^{i+M-1} \beta_k^{h,l,j,i}(t) \tag{12}$$

where

$$\beta_k^{h,l,j,i} = \beta_{T_{GCL}^h,\bar{W}_k^{h,j}}\left(t + T_{GCL}^h - \bar{W}_k^{h,j} - WT_k^{h,i} - R_k^{h,j,i}\right) \tag{13}$$

The term $\bar{W}_k^{h,j}$ is found using (29) and (11) and $WT_k^{h,i}$ using (8) and (10). Note that $\beta_{T,L}^h(t)$ is expressed for the TDMA protocol [6] as,

$$\beta_{T,L}^h(t) = C_h.max\left\{\left\lfloor\frac{t}{T}\right\rfloor L,\, t - \left\lceil\frac{t}{T}\right\rceil (T-L)\right\} \tag{14}$$

where C_h represents the link data rate. Moreover, the lower bounds of the service curve can be slightly different if the reference window is changed. Therefore, the lowest instantaneous values from all possible curves are considered as,

$$\beta_k^{h,l}(t) = \underset{1\le i\le M}{min}\left\{\beta_k^{h,l,i}(t)\right\} \tag{15}$$

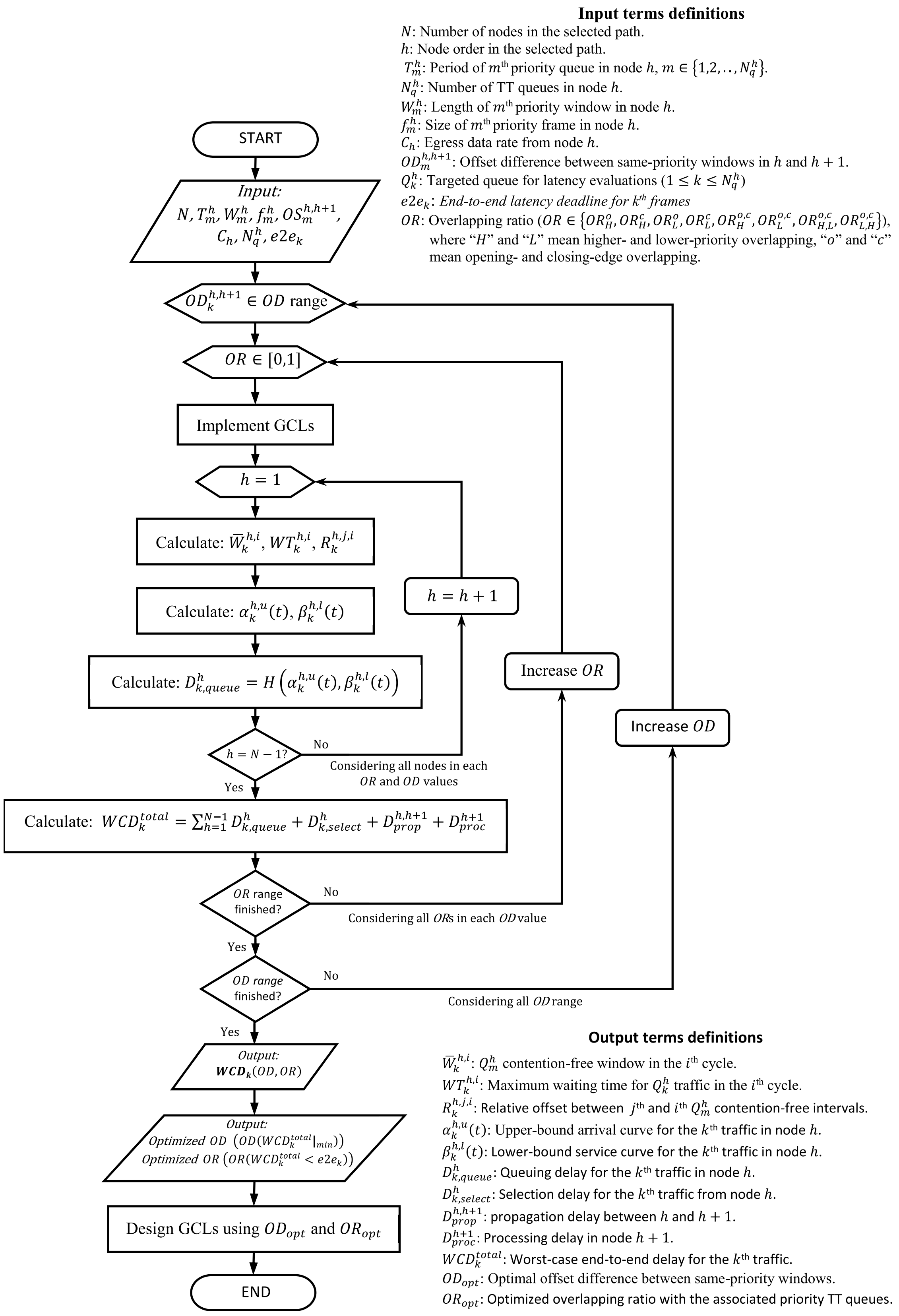
2) UPPER-BOUND ARRIVAL CURVE "$\alpha_k^{h,u}(t)$" DETERMINATION

For the first node (source), $\alpha_k^{FN,u}(t)$ is determined using (2). Subsequently, the accumulated upper-bound arrival curve in each non-first node h can be found using the input arrival curve at the previous node $h-1$, as follows.

$$\alpha_k^{h,u}(t) = \alpha_k^{h-1,u}\left(t + WCD_k^{h-1}\right) \tag{16}$$

where WCD_k^{h-1} represents the maximum latency that Q_k^{h-1} frame experiences to transfer from the k^{th} queue in $h-1$ to the k^{th} queue in h, i.e.,

$$WCD_k^{h-1} = D_{k,queue}^{h-1} + D_{k,select}^{h-1} + D_{prop}^{h-1,h} + D_{proc}^h \tag{17}$$


START
Input: $N, T_m^h, W_m^h, f_m^h, OS_m^{h,h+1}, C_h, N_q^h, e2e_k$
$OD_k^{h,h+1} \in OD$ range
$OR \in [0,1]$
Implement GCLs
$h = 1$
Calculate: $\overline{W}_k^{h,i}$, $WT_k^{h,i}$, $R_k^{h,j,i}$
Calculate: $\alpha_k^{h,u}(t)$, $\beta_k^{h,l}(t)$
Calculate: $D_{k,queue}^h = H\left(\alpha_k^{h,u}(t), \beta_k^{h,l}(t)\right)$
$h = N - 1$?
No
Yes
$h = h + 1$
Considering all nodes in each OR and OD values
Calculate: $WCD_k^{total} = \sum_{h=1}^{N-1} D_{k,queue}^h + D_{k,select}^h + D_{prop}^{h,h+1} + D_{proc}^{h+1}$
OR range finished?
No
Yes
Increase OR
Considering all ORs in each OD value
OD range finished?
No
Yes
Increase OD
Considering all OD range
Output: $\boldsymbol{WCD_k}(OD, OR)$
Output: Optimized OD $\left(OD(WCD_k^{total}|_{min})\right)$ Optimized OR $\left(OR(WCD_k^{total} < e2e_k)\right)$
Design GCLs using OD_{opt} and OR_{opt}
END

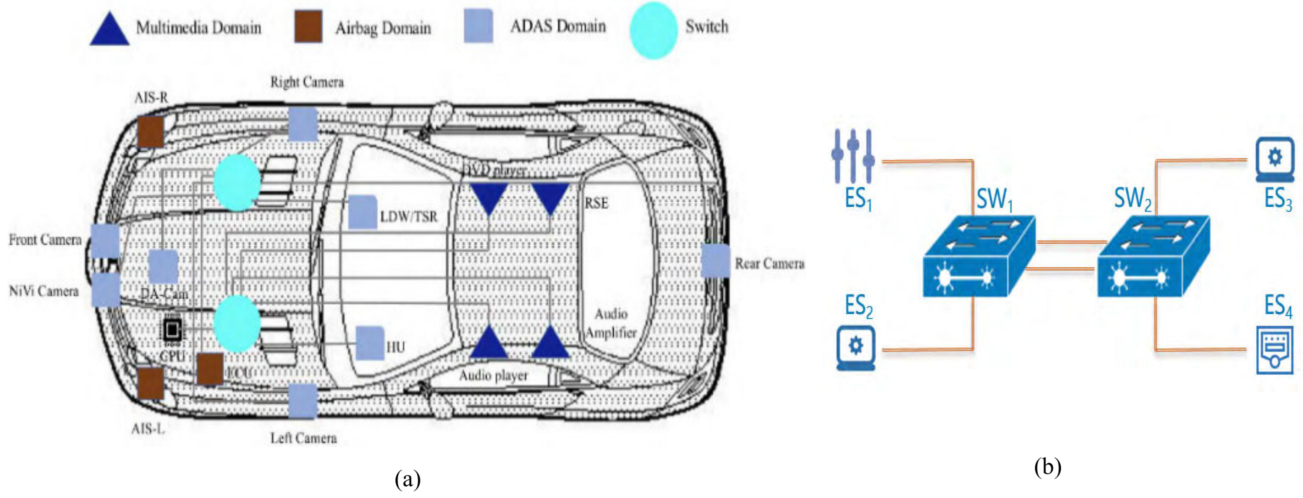
Input terms definitions

N: Number of nodes in the selected path.
h: Node order in the selected path.
T_m^h: Period of m^{th} priority queue in node h, $m \in \{1,2,..,N_q^h\}$.
N_q^h: Number of TT queues in node h.
W_m^h: Length of m^{th} priority window in node h.
f_m^h: Size of m^{th} priority frame in node h.
C_h: Egress data rate from node h.
$OD_m^{h,h+1}$: Offset difference between same-priority windows in h and $h+1$.
Q_k^h: Targeted queue for latency evaluations ($1 \le k \le N_q^h$)
$e2e_k$: *End-to-end latency deadline for k^{th} frames*
OR: Overlapping ratio ($OR \in \{OR_H^o, OR_H^c, OR_L^o, OR_L^c, OR_H^{o,c}, OR_L^{o,c}, OR_{H,L}^{o,c}, OR_{L,H}^{o,c}\}$), where "$H$" and "$L$" mean higher- and lower-priority overlapping, "o" and "c" mean opening- and closing-edge overlapping.

Output terms definitions

$\overline{W}_k^{h,i}$: Q_m^h contention-free window in the i^{th} cycle.
$WT_k^{h,i}$: Maximum waiting time for Q_k^h traffic in the i^{th} cycle.
$R_k^{h,j,i}$: Relative offset between j^{th} and i^{th} Q_m^h contention-free intervals.
$\alpha_k^{h,u}(t)$: Upper-bound arrival curve for the k^{th} traffic in node h.
$\beta_k^{h,l}(t)$: Lower-bound service curve for the k^{th} traffic in node h.
$D_{k,queue}^h$: Queuing delay for the k^{th} traffic in node h.
$D_{k,select}^h$: Selection delay for the k^{th} traffic from node h.
$D_{prop}^{h,h+1}$: propagation delay between h and $h+1$.
D_{proc}^{h+1}: Processing delay in node $h+1$.
WCD_k^{total}: Worst-case end-to-end delay for the k^{th} traffic.
OD_{opt}: Optimal offset difference between same-priority windows.
OR_{opt}: Optimized overlapping ratio with the associated priority TT queues.

FIGURE 9. A flow chart of the OFWOS algorithm.



LDW/TSR: Lane Departure Warning/Traffic Sign Recognition,
DA-Cam: Dashboard Camera,
AIS-R(L): Abbreviated Injury Scale-Right (or -Left),
RSE: Rear Seat Entertainment,
ES: End System,
HU: Head Up display,
NiVi Camera: Night Vision Camera,
ECU: Electronic Control Unit,
CPU: Central Processing Unit.
SW: Switch.

FIGURE 10. A graphical model for the applied use case: (a) a realistic vehicle use case with related networking components (b) representative model for the vehicular networking scenario.

where $D_{k,select}^{h-1} = f_k^{h-1,max} \big/ C_{h-1}$, $D_{prop}^{h-1,h}$ and D_{proc}^{h} are assumed here to be constant, while $D_{k,queue}^{h-1}$ represents the time delay that the Q_k^{h-1} frame experiences from the moment it reaches the queue until it arrives at the corresponding egress port. This delay contributes a significant part to the transmission delay, which is mainly dependent on the GCL design in the related node and is equal to the maximum horizontal distance between $\alpha_k^{h-1,u}(t)$ and $\beta_k^{h-1,l}(t)$ as follows,

$$D_{k,queue}^{h-1} = H\left(\alpha_k^{h-1,u}(t), \beta_k^{h-1,l}(t)\right) \tag{18}$$

Then, the upper bound end-to-end latency can be calculated by gathering the maximum latencies experienced in all nodes and links on the selected path as follows,

$$WCD_k^{total} = \sum_{h=1}^{N-1} WCD_k^h \tag{19}$$

where N is the number of nodes that the kth priority traffic passed through from the source to the destination. To simplify the design steps of the OFWOS algorithm, a flow chart summarizing the required computations in sequence to conclude an optimized GCL implementation is presented in Fig. 9 below.

VI. PERFORMANCE EVALUATION

A. CASE STUDY AND EXPERIMENTAL SETUP

In this section, we apply our proposed model on a TSN framework based on a realistic vehicle case as depicted in Fig. 10(a) [6], [37]. A simplified representative network structure for the vehicle is shown in Fig. 10(b). It consists of two switches and four end systems, which could be sensors, cameras, or actuators used in the connected vehicle to collect relevant information from the surrounding environment.

We assume that all physical links in Fig. 10(b) have the same data rate (1 Gbps) in all experiments. In our experiments, we consider the connection between ES$_1$ and ES$_4$ as the targeted path for latency evaluation. In each node, incoming traffic is differentiated into several priority queues. The initial opening-edge times ($t_m^{ES1,o,1}$) in the source-GCL (ES$_1$-GCL) for associated queues are tabulated in Table 2. The related closing-edge times and non-first GCLs are updated using (6) and (7) by assuming 20μs open window durations and 250μs periods for all TT queues. Note that the fourth priority is the targeted queue to assess our proposed algorithm under a changeable offset difference between same-priority windows in adjacent nodes. The initial largest sizes of Q_4^h and Q_{4+}^h frames are 400 and 300 bytes, respectively.

As we use NC theory to express our model, a Java API of the Real-Time Calculus (RTC) toolbox [38] is used to construct associated GCLs, and accordingly, calculate the worst-case latency bounds for Q_4^h. The RTC toolbox is an effective framework implemented based on min-plus and max-plus operators to integrate real-time systems. We use a computer with an Intel Core i7-3770 CPU at 3.40 GHz and RAM 12 GB to run the presented algorithm.

B. NUMERICAL RESULTS AND DISCUSSION

Several experiments are applied to the OFWOS algorithm to examine the effect of *OD* between the same-priority windows

TABLE 2. The initial GCL implementation.

TT- priority class	Opening-edge time in the first node (μs) ($t_m^{ES1,o,1}$)
$m = 1$	10
$m = 2$	30
$m = 3$	55
$m = 4$	90
$m = 5$	110
$m = 6$	150
$m = 7$	185
$m = 8$	210

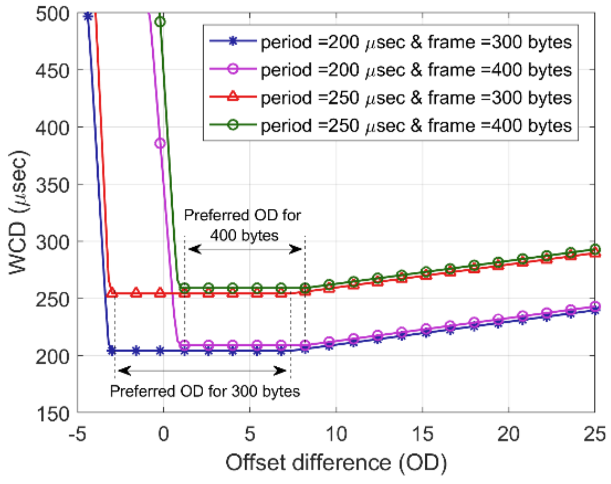
in the adjacent nodes under both complete-isolated and overlapped windows in GCLs' implementation.

1) FULL ISOLATION SCENARIO EVALUATIONS

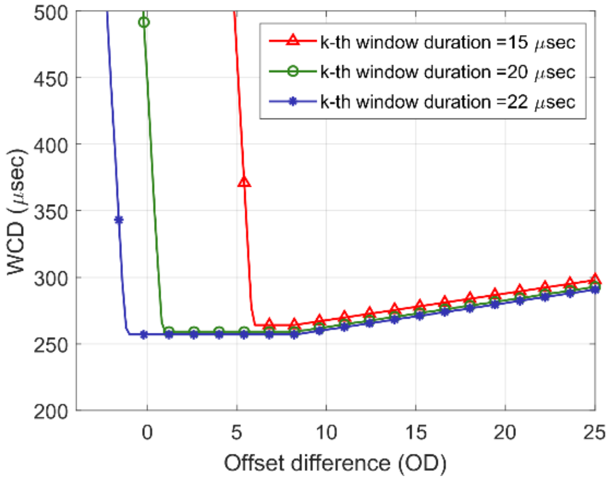
In this subsection, we implement complete-isolated windows in all GCLs as defined in Table 2. It means that the fourth-priority windows are completely isolated from other TT windows in all GCLs on the selected path. Under this assumption, the *WCD* is examined with varying *OD* between same-priority windows. All possible *OD* values are considered where the incoming traffic from the previous node (Q_4^{h-1} window) may receive any amount of service during the current window (Q_4^h window). These possible values are covered by the ranges specified in (4).

The *WCD* performance is evaluated under variable *OD* with considering three critical metrics: the frame size, the transmission period, and the window length. The Q_4^h frame size ($f_4^{h,max}$) and its period (T_4^h) are examined in Fig. 11(a), and the window length (W_4^h) in Fig. 11(b).

From the figures, it is obvious that the *OD* parameter significantly affects the latency performance. In all the cases tested, there is a preferred *OD* range ($OD\,|_{WCD_{min}}$) where the Q_4^h traffic experiences the lowest *WCD*. Accordingly, the optimal *OD* should be selected from this range. Additionally, Fig. 11(a) shows that the preferred *OD* range is not affected by changing the transmission period of the associated queue. Decreasing the transmission cycle from 250μs to 200μs has reduced the *WCD* by the same percentage in all *OD*s without any effect on the length of $OD\,|_{WCD_{min}}$ range. However, the Q_4^h frame size has a significant impact on the $OD\,|_{WCD_{min}}$ range, as observed from Fig. 11(a). Large frames reduce the $OD\,|_{WCD_{min}}$ range. For example, the lowest *WCD* is obtained during $1\mu s \leq OD \leq 8.2\mu s$ when $f_4^{h,max} = 400$ bytes, and during $-3\mu s \leq OD \leq 8.2\mu s$ when $f_4^{h,max} = 300$ bytes. In addition, the lowest *WCD* is obtained for long *OD* intervals under wide window durations, as shown in Fig. 11(b). For example, the lowest *WCD* is obtained under $OD \in [-1\mu s, 8.2\mu s]$ when $W_4^h = 22\mu s$, $OD \in [1\mu s, 8.2\mu s]$ when $W_4^h = 20\mu s$, and $OD \in [6\mu s, 8.2\mu s]$ when $W_4^h = 15\mu s$. Furthermore, as the *WCD* increases dramatically when



(a)



(b)

FIGURE 11. The worst-case end-to-end delay (WCD) as a function of offset difference (OD) between the same priority windows in the adjoining nodes under; (a) various transmission periods and different frame sizes, and (b) various window durations.

$OD < OD\,|_{WCD_{min}}$ and slowly when $OD > OD\,|_{WCD_{min}}$ in all test cases, the optimal *OD* must be carefully selected from the preferred range. In specific, it is safer to choose an *OD* that is near the largest limit of the $OD\,|_{WCD_{min}}$ range.

2) OVERLAPPED-WINDOWS SCENARIO EVALUATIONS

In this subsection, the impact of one-sided overlapping is examined using four different experiments, as illustrated in Table 3.

Experiments 1 and 2 are applied to evaluate the latency performance under opening- and closing-edge overlapping with lower-priority windows. Higher-priority overlapping is considered in Experiments 3 and 4. We assign the fifth-priority queue as a lower-priority and the second-priority queue as a higher-priority to overlap with the targeted queue (Q_4^h) by a variable *OR*, as stated in Table 3. As our proposed algorithm (OFWOS) is an extension to the FWOS model in [8], which also included the overlapping scenarios, the *WCD* evaluations

TABLE 3. Different experiments for one-sided overlapping scenarios with higher- and lower-priority TT windows.

Experiment	Priority	Open-Window Edges of the Overlapped Queue ($t_m^{h,o,i}, t_m^{h,c,i}$)
1 Opening-edge lower-priority overlapping	5	$t_5^{h,o,i} = t_4^{h,o,i} + W_4^h.OR_{4,5}^{h,B} - W_5^h$ $t_5^{h,c,i} = t_4^{h,o,i} + W_4^h.OR_{4,5}^{h,B}$
2 Closing-edge lower-priority overlapping	5	$t_5^{h,o,i} = t_4^{h,c,i} - W_4^h.OR_{4,5}^{h,E}$ $t_5^{h,c,i} = t_4^{h,c,i} - W_4^h.OR_{4,5}^{h,E} + W_5^h$
3 Opening-edge higher-priority overlapping	2	$t_2^{h,o,i} = t_4^{h,o,i} + W_4^h.OR_{4,2}^{h,B} - W_2^h$ $t_2^{h,c,i} = t_4^{h,o,i} + W_4^h.OR_{4,2}^{h,B}$
4 Closing-edge higher-priority overlapping	2	$t_2^{h,o,i} = t_4^{h,c,i} - W_4^h.OR_{4,2}^{h,E}$ $t_2^{h,c,i} = t_4^{h,c,i} - W_4^h.OR_{4,2}^{h,E} + W_2^h$

from our algorithm are compared with the FWOS results in all overlapping experiments. For all these overlapping scenarios (Experiments 1-4), we assume 20μs window length and 250μs period for all TT queues, and the sizes of Q_4^h and Q_{4+}^h frames are 400 and 300 bytes, respectively. These experiments are applied with varying *OD* between the same-priority windows in the adjacent nodes. The *OD* values are selected from the preferred *OD* range under non-overlapping GCLs ($OD\,|_{WCD_{min}}$). Based on the above window and frame settings, $1\mu s \leq OD\,|_{WCD_{min}} \leq 8.2\mu s$.

Experiments 1 and 2: These experiments examine *WCD* for Q_4^h traffic under variable lower-priority overlapping in the range $OR \in (0, 1)$. Fig. 12(a) presents the *WCD* performance under opening-edge overlap and Fig. 12(b) for the closing-edge overlapping case.

For the opening-edge overlapping, Fig. 12(a) shows that different *OD*s meet different *WCD*s for the same *OR*s. The $OD|_{WCD_{min}}$ range produces unequal latency performance. It is previously noticed that the performance improves as the *OD* nears the largest limit of $OD\,|_{WCD_{min}}$ interval ($OD = 8.2\mu s$). If $OD \leq 3.1\mu s$, the performance is worse than that in [8] ($OD = 20\mu s$). However, the optimal *OD* under opening-edge lower-priority overlapping can be conditionally defined as $3.4\mu s \leq OD_{L,opt}^{o} \leq 8.2\mu s$, where the latency has reduced by 23.7μs compared to [8]. For the closing-edge overlapping, Fig. 12(b) shows that all $OD|_{WCD_{min}}$ values experience the lowest *WCD*, which is less than that in [8] by 23.7μs, i.e., $1\mu s \leq OD_{L,opt}^{c} \leq 8.2\mu s$. These results confirm that the closing-edge overlapping has less impact on the latency than opening-edge overlapping.

Experiments 3 and 4: These experiments examine the effect of *OD* on the latency performance under opening- and closing-edge higher-priority overlapping, as depicted in Figs. 13(a) and (b), respectively.

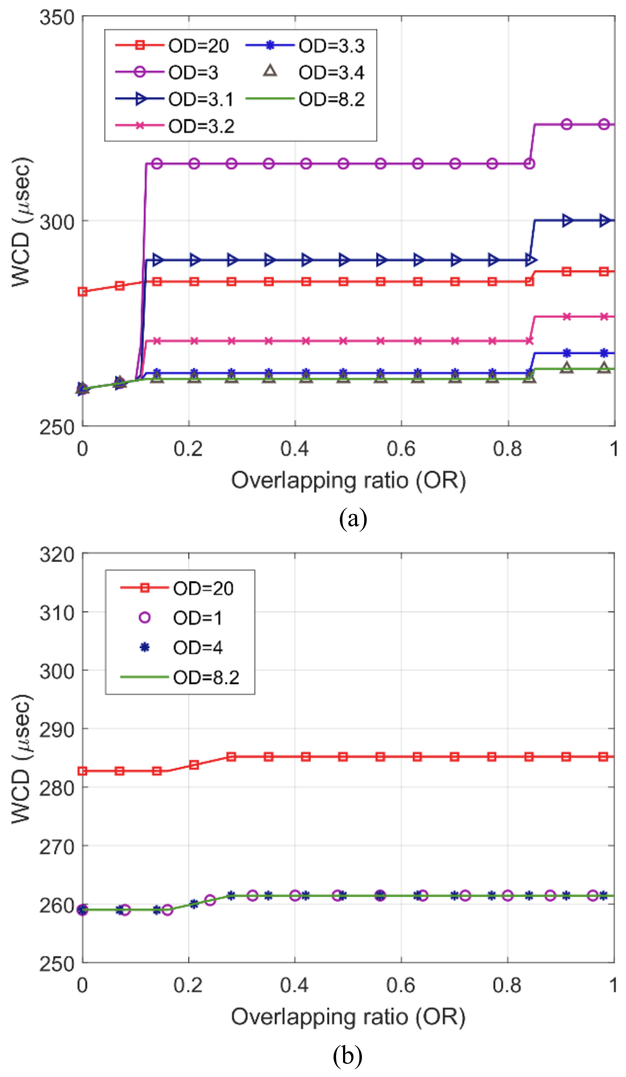


FIGURE 12. The worst-case end-to-end delay (WCD) vs. lower-priority overlapping ratio under various offset differences (ODs) between the same priority windows in the adjoining nodes; (a) opening-edge overlapping case, and (b) closing-edge overlapping case.

Figs. 13(a),(b) verify that higher-priority overlapping has a bigger impact on *WCD* performance than lower-priority. Furthermore, similar to the lower-priority case, the opening-edge overlap degrades *WCD* compared to closing-edge. For *OD* evaluations, using $OD\,|_{WCD_{min}}$ range ([1μs, 8.2μs]) reduces *WCD* by 23.7μs compared to [8] but only under small *OR*s. After certain *OR*s, most of $OD\,|_{WCD_{min}}$ values produce worse *WCD*s than [8]. For example, as shown in Fig. 13(a), 20μs *OD* (as presented in [8]) with opening-edge overlapping produces better *WCD* than; 6μs *OD* if $OR > 26\%$, 7μs *OD* if $OR > 31\%$, and 7.5μs *OD* if $OR > 34\%$. But the *WCD* performance enhances as the *OD* value nears the upper-limit of $OD\,|_{WCD_{min}}$ range ($OD = 8.2\mu s$), where the best *WCD* performance as observed in Figs. 13(a),(b) occurs under all higher-priority overlapping situations. Thus, this value is the optimal *OD* for opening- and closing-edge higher priority overlapping, i.e., $OD_{H,opt}^{o} = OD_{H,opt}^{c} = 8.2\mu s$.

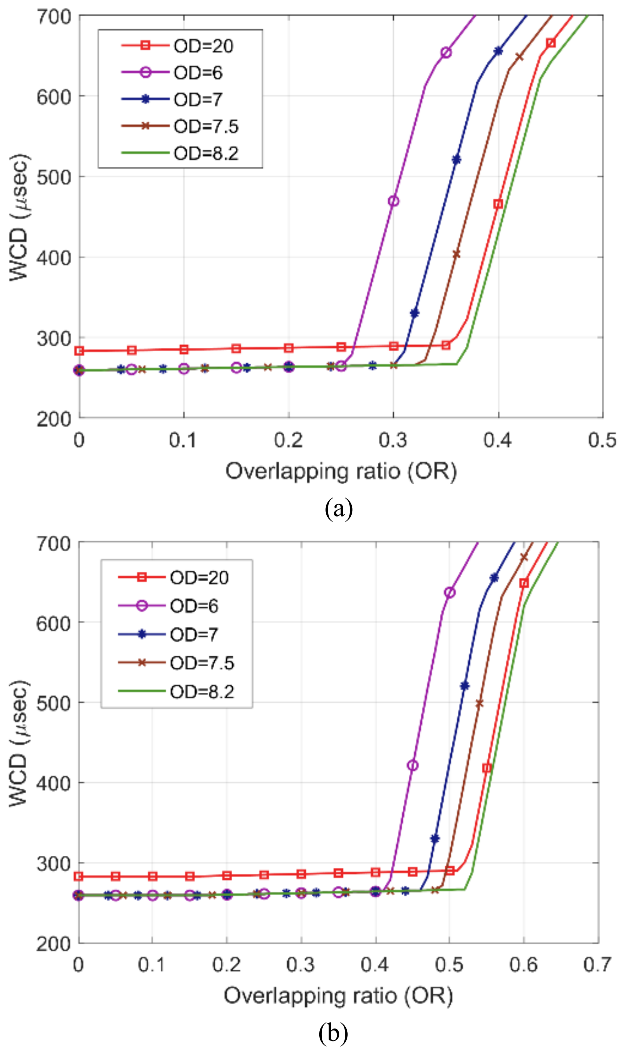


FIGURE 13. The worst-case end-to-end delay (WCD) vs. higher-priority overlapping ratio under various offset differences (ODs) between the same priority windows in the adjoining nodes; (a) opening-edge overlapping case, and (b) closing-edge overlapping case.

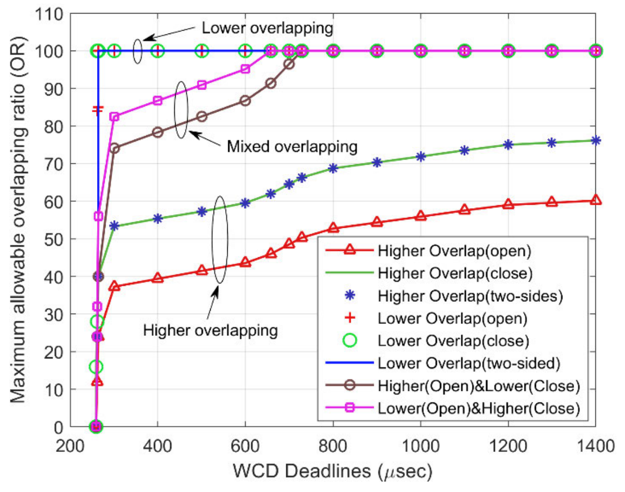


FIGURE 14. Maximum allowable one-sided and two-sided overlapping ratio (OR) vs. targeted WCD deadlines with any priority queue (lower-, higher-, or mixed-priority).

Therefore, the OFWOS algorithm optimizes the offset difference between same-priority windows in all GCL implementations (non-overlapped- and overlapped-window GCLs).

Based on the optimized *OD* and the *WCD* deadlines for the incoming flows, we can limit the overlapping between TT windows. Each TT frame should be selected to a queue where it must experience a *WCD* no higher than its deadline. For this reason, the overlapping between TT windows must be constrained when implementing the associated GCLs. Our algorithm calculates *WCD*s based on flexible window-overlapping GCLs, the *OR* that meets those *WCD*s should be considered a maximum allowable overlapping ratio that guarantees the targeted latency deadline for the associated queue.

Accordingly, the maximum allowable *OR* for each latency deadline can be determined easily using the above evaluations, as depicted in Fig. 14. This figure specifies the overlapping constraints based on the priority and the position of overlapping. From Fig. 14, it is observed that latency deadlines can be met under the largest/smallest *OR* with lower/higher priority queues, respectively. Mixed overlapping has an intermediate impact on *WCD* bounds. For all priority cases, the opening-edge overlap has a bigger influence on the latency than the closing-edge. For example, the latency deadline of $500\mu s$ can be met under at most 41.47% *OR* with opening-edge higher priority, 57.26% with closing-edge higher priority or two-sided higher priority (28.63% from each side), 82.53% with higher/lower (41.26% with opening-edge higher priority and 41.26% with closing-edge lower priority), 90.93% with lower/higher (45.46% with opening-edge lower priority and 45.46% with closing-edge higher priority), and 100% with any lower priority overlapping situation.

3) PERFORMANCE IMPROVEMENTS UNDER OVERLAPPING SITUATIONS

As described previously, allowing TT open windows to overlap can save more intervals for unscheduled transmissions (AVB and BE). These intervals will proportionally increase the available bandwidth for unscheduled traffic. The maximum BW that can be saved is obtained under the maximum allowable *OR* for each specific *WCD* deadline. Based on the results in Fig. 14, we illustrate, in Fig. 15, the maximum additional unscheduled BW percentage per link concerning the targeted *WCD* deadline. We assume two separate TT queue numbers (eight and six) at the egress port under the same *WCD* deadline. Both porosity and back-to-back configurations are considered for combining scheduled and unscheduled transmission windows in the GCL design for each latency deadline. The porosity configuration here means each two adjacent windows are overlapped and isolated from others by a blank for unscheduled transmissions (one-sided overlapping case). The back-to-back pattern denotes TT windows follow each other as a one time-block in the hyperperiod (two-sided overlapping case).

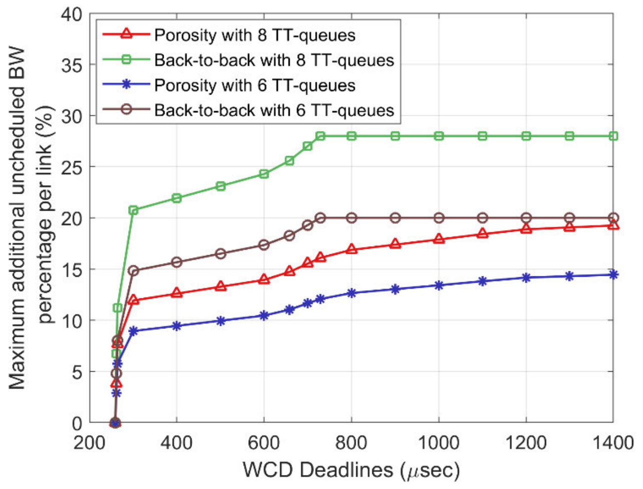


FIGURE 15. Maximum additional BW percentage per link for unscheduled traffic based on the maximum allowable *OR* between TT windows under specific latency deadlines with porosity and back-to-back configurations.

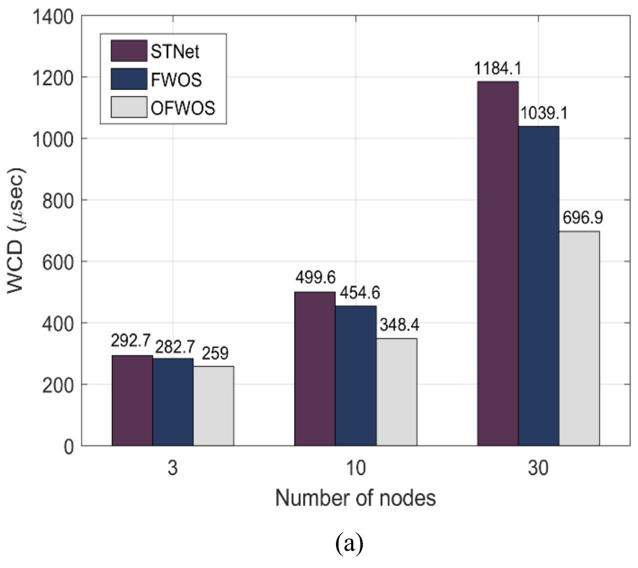
TABLE 4. The optimized *OD*s between the same priority windows in the adjacent nodes based on various overlapping situations among TT windows at the same node.

Overlapping condition	Optimum Offset Difference ($OD_{opt.}^{h-1,h}$) range
Non-overlapped-window GCLs	$[1\mu s, 8.2\mu s]$
Lower priority overlapping (opening)	$[3.4\mu s, 8.2\mu s]$
Lower priority overlapping (closing)	$[1\mu s, 8.2\mu s]$
Lower priority overlapping (two-sided)	$[3.4\mu s, 8.2\mu s]$
Higher priority overlapping	$8.2\mu s$
Mixed priority overlapping	$8.2\mu s$

As shown in Fig. 15, it is observed that the porosity style increases the BW with a percent lower than back-to-back, as the number of overlapping intervals in back-to-back is larger than that in the porosity style. Moreover, egress ports with eight TT queues produce more unscheduled BW than six as the overlapping intervals are larger. For example, with guaranteed $400\mu s$ latency bound for all TT queues, we can save additional BW for unscheduled transmissions using eight TT queues by 12.6% with the porosity style and 21.93% with back-to-back, and by 9.45% with the porosity and 15.67% with back-to-back using six TT queues. These improvements increase with more flexible *WCD* deadlines and, oppositely, decrease with stricter *WCD* deadlines, as observed in Fig. 15.

4) OVERALL *OD* OPTIMIZATIONS

Based on the evaluation results in the previous subsections, it is observed that the OFWOS optimization depends on the overlapping situations between TT windows. Also, adjusting frame and window specifications will result in different optimal values of *OD* even with non-overlapped window GCLs. Accordingly, Table 4 presents the optimized *OD*s with respect to the overlapping conditions under the assumption that $f_4^{h,max} = 400$ bytes, $f_{4+}^{h,max} = 300$ bytes, $W_4^h = 20\mu s$, and $T_m^h = 250\mu s$ for $1 \leq h \leq N$ and $1 \leq m \leq N_q^h$. Changing these parameters will undoubtedly produce different optimal values of *OD*. However, this table introduces a significant guide that can help TSN designers to implement more convenient GCLs.



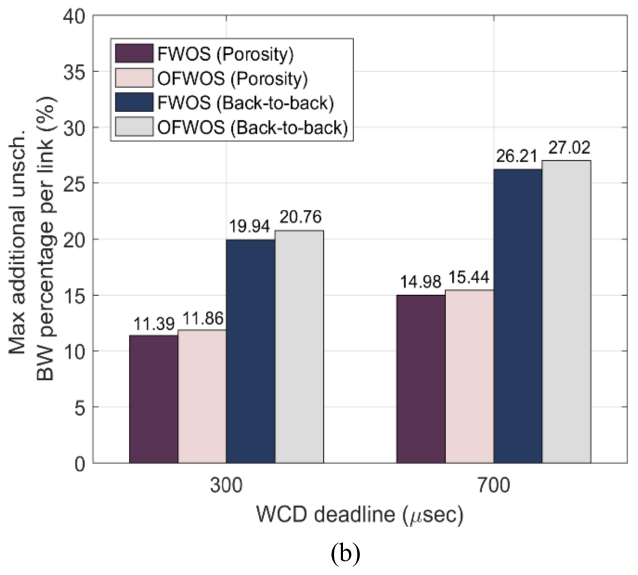


FIGURE 16. Comparison between OFWOS algorithm and related works; (a) WCD evaluations using OFWOS, STNet [7], and FWOS [8] based on a various number of nodes in the selected path and full isolated TT windows in each node. (b) Maximum additional BW percentage for unscheduled transmissions based on the maximum allowable one-sided and two-sided OR between TT windows under $300\mu s$ and $700\mu s$ *WCD* deadlines with porosity and back-to-back configurations.

5) COMPARISON WITH RELATED WORKS

The findings from this work are compared with the previous related works in [7] (STNet) and [8] (FWOS). As the work in [7] did not consider TT window overlapping, the comparison is done concerning *WCD* evaluations under a non-overlapping scenario, as depicted in Fig. 16(a). The authors in [7] also considered different *OD*s, including $OD = 0, 20, 45, 60, 85$, and $135\mu s$ arbitrarily chosen between adjacent nodes through the selected path. In [8], the authors considered $OD = 20, 40$, and $60\mu s$ applied separately on the whole path. These *OD* values produced different *WCD*s in both algorithms. The lowest *WCD*s were obtained with $OD = 20\mu s$ as addressed in [8]. Fig. 16(a) compares OFWOS with STNet and FWOS algorithms under different network scales (three-hop, 10-hop, and 30-hop end-to-end connections). In all topologies, OFWOS gives the lowest *WCD*s and the improvement increases under large-scale network topologies. For example, OFWOS reduces the *WCD*s in STNet by 11.5%, 30.3%, and 41.1% under three-hop, 10-hop, and 30-hop connections, respectively. Compared with FWOS, OFWOS minimizes the latency by 8.4%, 23.4%, and 32.6%, respectively.

For overlapping-based scenarios, the *WCD* improvements compared to [8] are addressed in Figs. 12 and 13. Based on these achievements, more QoS enhancements are obtained. As OFWOS obtains lower *WCD*s in all overlapping situations compared to [8], larger *OR*s are allowed using OFWOS in contrast to FWOS under the same *WCD* deadlines.

Fig. 16(b) presents the maximum additional BW percentage available for unscheduled transmissions in the associated link by assuming all TT queues require equal *WCD* deadlines. Two different *WCD* deadlines ($300\mu s$ and $700\mu s$) are selected. As shown in Fig. 16(b), OFWOS saves more unscheduled BW than FWOS in all cases. It is worthy to note that the results in Fig. 16(b) are provided by assuming eight TT queues in the egress port. It means that the number of overlapping intervals is four at the porosity style and seven at back-to-back. Accordingly, more BW can be saved for unscheduled transmissions if many TT queues are assigned.

VII. CONCLUSION

Urgent data in several extremely time-sensitive applications require strict QoS guarantees. The most essential requirement is the determinism for end-to-end communications. Recently, the TSN standards have been introduced based on Ethernet networking to support such real-time environments. For traffic scheduling, a time-aware shaping technique is presented as a window-based mechanism in IEEE 802.1Qbv standard. Although several window-based scheduling algorithms have been proposed to serve TT flows, the offset difference between the same-class windows has not been optimized yet.

In this paper, an optimized flexible window-overlapping scheduling algorithm is proposed with a complete *WCD* analysis for TT traffic under variable *OD*. The OFWOS model optimizes *OD* under all overlapping metrics; priority, position, and the ratio (*OR*). A realistic vehicular scenario is used to evaluate the performance of OFWOS. Initially, the latency evaluation is performed under non-overlapping GCLs to determine a preferred *OD* range, where the *WCD* is the lowest. The preferred *OD* range is examined by adjusting several relevant transmission parameters, such as window length, period, and frame size. The presented results confirm that window duration and frame size have a considerable impact on the preferred *OD* range. Wide windows or short frames meet a wider *OD* range, where *WCD* is the lowest, and vice versa. Although decreasing the transmission period reduces *WCD*s, the preferred *OD* domain stays the same. After that, the OFWOS performance is examined by varying all overlapping metrics, which led to an optimal *OD* for each overlapping condition. Compared to previous related works, the OFWOS algorithm achieves less pessimistic *WCD*s, leading to an increasing unscheduled BW under overlapping-based GCL designs. The latency reduction also increases in large-scale network topologies. For example, OFWOS obtains 8.4% and 32.6% latency reduction compared to the latest related work for three-hop and 30-hop TSN connections, respectively. Accordingly, combining the *OD* and *OR* optimizations in GCLs implementations can assist interested vehicular designers to suitably serve incoming data from related devices, such as sensors, cameras, and actuators.

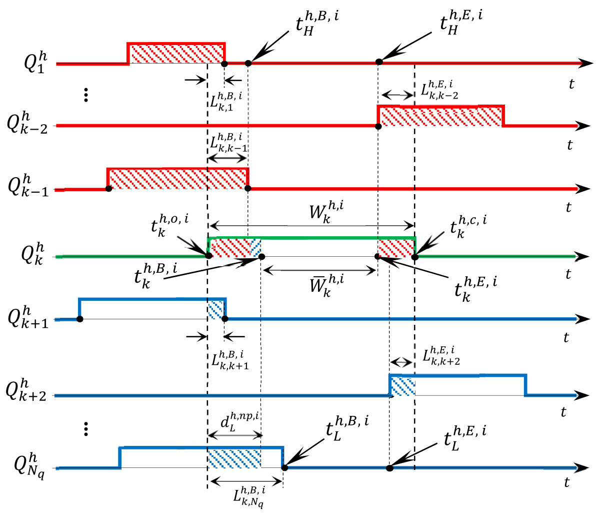


FIGURE 17. Opening- and closing-edge window-overlapping with higher- and lower-priority TT queues.

APPENDIX A CONTENTION-FREE INTERVAL CALCULATION

A. OVERLAPPING CONSIDERATIONS

Window-overlapping is considered here under three related aspects: priority, position, and overlapping ratio. Fig. 17 shows all overlapping situations of Q_k^h window with other TT queues in the i^{th} transmission cycle. The related timing analysis is summarized below, and further details can be read in [8].

1) OPENING-EDGE OVERLAPPING

The opening-edge overlapping is considered for the targeted Q_k^h window if there is another TT window is open when the Q_k^h gate (G_k^h) changes from closed event to open, as shown in Fig. 17. The *OR* at the opening-edge with higher and lower-priorities can be formulated respectively as,

$$OR_{k,H}^{h,B,i} = \frac{\max\limits_{1\leq k^{-}\leq k-1}\left\{L_{k,k^{-}}^{h,B,i}\right\}}{W_{k}^{h,i}} = \frac{L_{k,H}^{h,B,i}}{W_{k}^{h,i}} \tag{20}$$

$$OR_{k,L}^{h,B,i} = \frac{\max\limits_{k+1\leq k^{+}\leq N_q}\left\{L_{k,k^{+}}^{h,B,i}\right\}}{W_{k}^{h,i}} = \frac{L_{k,L}^{h,B,i}}{W_{k}^{h,i}} \tag{21}$$

In the higher-priority overlapping case, the contention-free interval starts at the closing-edge of the higher-priority window that has the largest overlap with Q_k^h window. Thus, the starting time of the contention-free window can be given as,

$$t_H^{h,B,i} = t_k^{h,o,i} + L_{k,H}^{h,B,i} = t_k^{h,o,i} + OR_{k,H}^{h,B,i}.W_k^{h,i} \tag{22}$$

In the lower-priority overlapping case, the Q_k^h frame has to wait for just the Q_{k+}^h frames that in the transmission status if the nun-preemption technique is applied between TT transmissions. Accordingly, the contention-free interval will start at,

$$t_L^{h,B,i} = t_k^{h,o,i} + d_L^{h,np,i} \tag{23}$$

where

$$d_L^{h,np,i} = \begin{cases} \min\left\{f_{k+}^{h,max}, L_{k,L}^{h,B,i}\right\}, & t_{k+}^{h,o,i} < t_k^{h,o,i} < t_{k+}^{h,c,i} \\ 0, & Otherwise \end{cases}$$

and $f_{k+}^{h,max}$ represents the maximum size of lower-priority frames that their windows overlap with Q_k^h window.

2) CLOSING-EDGE OVERLAPPING

Similar to the opening-edge overlapping case and as depicted in Fig. 17, the *OR* at the closing-edge with higher- and lower-priorities can be formulated respectively as,

$$OR_{k,H}^{h,E,i} = \frac{\max\limits_{1\leq k^{-}\leq k-1}\left\{L_{k,k^{-}}^{h,E,i}\right\}}{W_{k}^{h,i}} = \frac{L_{k,H}^{h,E,i}}{W_{k}^{h,i}} \tag{24}$$

$$OR_{k,L}^{h,E,i} = \frac{\max\limits_{k+1\leq k^{+}\leq N_q}\left\{L_{k,k^{+}}^{h,E,i}\right\}}{W_{k}^{h,i}} = \frac{L_{k,L}^{h,E,i}}{W_{k}^{h,i}} \tag{25}$$

In the higher-priority overlapping case, the contention-free interval ends at the opening-edge of the higher-priority window that has the largest overlapping with Q_k^h window. Thus, the ending time of the contention-free window can be given as,

$$t_H^{h,E,i} = t_k^{h,c,i} - L_{k,H}^{h,E,i} = t_k^{h,c,i} - OR_{k,H}^{h,E,i}.W_k^{h,i} \tag{26}$$

In the lower-priority overlapping case, the contention-free interval is not influenced as Q_k^h has a higher priority than k^+queues. However, the worst-case waiting time of the associated frame is enlarged, as discussed in Subsection V-C.

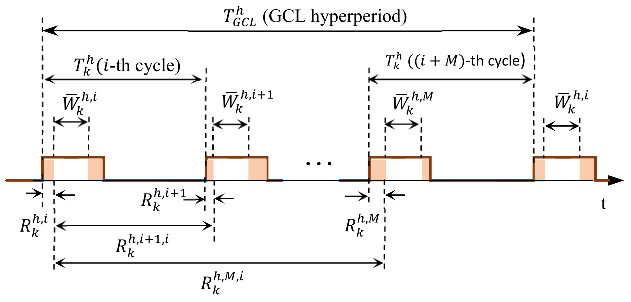


FIGURE 18. Relative offsets of contention-free windows to the benchmark in the hyper-period.

B. GAURD BAND EFFECT

At the end of each TT window, a block interval (guard band) ensures the remaining time is enough to transmit the associated frame. If the frame arrives during this interval, the service is not guaranteed. This interval equals the time required to send the largest size of Q_k^h frame ($d_k^{h,gb} = f_k^{h,max}/C_h$). Accordingly, the contention-free interval will end at,

$$t_k^{h,gb,i} = t_k^{h,c,i} - d_k^{h,gb} \tag{27}$$

By accumulating the overlapping and guard band considerations discussed above, the starting and ending times of the i^{th} Q_k^h contention-free window are constrained as,

$$\begin{aligned} t_k^{h,B,i} &= \max\left\{t_L^{h,B,i}, t_H^{h,B,i}\right\} \\ t_k^{h,E,i} &= \min\left\{t_k^{h,gb,i}, t_H^{h,E,i}\right\} \end{aligned} \tag{28}$$

Thus, the length of the i^{th} Q_k^h contention-free window can be given as,

$$\bar{W}_k^{h,i} = \begin{cases} \max\left\{t_k^{h,E,i} - t_k^{h,B,i}, \frac{f_k^{h,max}}{C_h}\right\}, & t_k^{h,B,i} < t_k^{h,E,i} \\ 0, & t_k^{h,B,i} \geq t_k^{h,E,i}. \end{cases} \tag{29}$$

By considering the i^{th} window as a reference, as shown in Fig. 18, the relative offset of the i^{th} contention-free interval from the opening-edge can be represented as,

$$R_k^{h,i} = \begin{cases} t_k^{h,B,i} - t_k^{h,o,i}, & \bar{W}_k^{h,i} \neq 0 \\ 0, & \bar{W}_k^{h,i} = 0 \end{cases} \tag{30}$$

Also, by considering the i^{th} window as a reference, the relative offset of the j^{th} contention-free interval is formed as,

$$R_k^{h,j,i} = (j-i).T_k^h - R_k^{h,i} + R_k^{h,j} \tag{31}$$

REFERENCES

[1] N. Finn, "Introduction to time-sensitive networking," *IEEE Commun. Standards Mag.*, vol. 2, no. 2, pp. 22–28, Jun. 2018, doi: 10.1109/mcomstd.2018.1700076.

[2] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic*, Standard 802.1QbvTM-2015, IEEE Standards Association, 2015, doi: 10.1109/IEEESTD.2016.8613095.

[3] *Virtual Bridged Local Area Networks-Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams*, D. Standard and M. A. Networks, New York, NY, USA, Dec. 2009, pp. 1–87, doi: 10.1109/IEEESTD.2009.5375704.

[4] L. Zhao, P. Pop, Z. Zheng, H. Daigmorte, and M. Boyer, "Latency analysis of multiple classes of AVB traffic in TSN with standard credit behavior using network calculus," *IEEE Trans. Ind. Electron.*, vol. 46, no. 10, pp. 1–11, Oct. 2020, doi: 10.1109/TIE.2020.3021638.

[5] L. Zhao, P. Pop, and S. S. Craciunas, "Worst-case latency analysis for IEEE 802.1Qbv time sensitive networks using network calculus," *IEEE Access*, vol. 6, pp. 41803–41815, 2018, doi: 10.1109/ACCESS.2018.2858767.

[6] P. Zhang, Y. Liu, J. Shi, Y. Huang, and Y. Zhao, "A feasibility analysis framework of time-sensitive networking using real-time calculus," *IEEE Access*, vol. 7, pp. 90069–90081, 2019, doi: 10.1109/ACCESS.2019.2927516.

[7] L. Zhao, P. Pop, Z. Gong, and B. Fang, "Improving latency analysis for flexible window-based GCL scheduling in TSN networks by integration of consecutive nodes offsets," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 1–11, Oct. 2020, doi: 10.1109/JIOT.2020.3031932.

[8] K. M. Shalghum, N. K. Noordin, A. Sali, and F. Hashim, "Network calculus-based latency for time-triggered traffic under flexible window-overlapping scheduling (FWOS) in a time-sensitive network (TSN)," *Appl. Sci.*, vol. 11, no. 9, p. 3896, Apr. 2021.

[9] J.-Y. and P. T. Le Boudec. (2004). *NETWORK CALCULUS A Theory of Deterministic Queuing Systems for the Internet*. [Online]. Available: https://portal.acm.org/citation.cfm?id=1755809

[10] L. Maile, K.-S. Hielscher, and R. German, "Network calculus results for TSN: An introduction," in *Proc. Inf. Commun. Technol. Conf. (ICTC)*, May 2020, pp. 131–140.

[11] A. Finzi, A. Mifdaoui, F. Frances, and E. Lochin, "Network calculus-based timing analysis of AFDX networks with strict priority and TSN/BLS shapers," in *Proc. IEEE 13th Int. Symp. Ind. Embedded Syst. (SIES)*, Jun. 2018, pp. 1–10, doi: 10.1109/SIES.2018.8442080.

[12] S. S. Craciunas and R. S. Oliver, "An overview of scheduling mechanisms for time-sensitive networks," in *Proc. Real-Time Summer School LÉcole dÉté Temps Réel (ETR*, 2017, pp. 1–7.

[13] A. A. Atallah, G. B. Hamad, and O. A. Mohamed, "Routing and scheduling of time-triggered traffic in time-sensitive networks," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4525–4534, Jul. 2020, doi: 10.1109/TII.2019.2950887.

[14] X. Jin, C. Xia, N. Guan, and P. Zeng, "Joint algorithm of message fragmentation and no-wait scheduling for time-sensitive networks," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 2, pp. 478–490, Feb. 2021, doi: 10.1109/JAS.2021.1003844.

[15] F. Dürr and N. G. Nayak, "No-wait packet scheduling for IEEE time-sensitive networks (TSN)," in *Proc. 24th Int. Conf. Real-Time Netw. Syst. (RTNS)*, 2016, pp. 203–212, doi: 10.1145/2997465.2997494.

[16] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. Elbakoury, "Performance comparison of IEEE 802.1 TSN time aware shaper (TAS) and asynchronous traffic shaper (ATS)," *IEEE Access*, vol. 7, pp. 44165–44181, 2019, doi: 10.1109/ACCESS.2019.2908613.

[17] M. Kim, J. Min, D. Hyeon, and J. Paek, "TAS scheduling for real-time forwarding of emergency event traffic in TSN," in *Proc. Int. Conf. ICT Converg.*, Oct. 2020, pp. 1111–1113, doi: 10.1109/ICTC49870.2020.9289458.

[18] D. Hellmanns, A. Glavackij, J. Falk, R. Hummen, S. Kehrer, and F. Durr, "Scaling TSN scheduling for factory automation networks," in *Proc. IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, Apr. 2020, pp. 1–8, doi: 10.1109/WFCS47810.2020.9114415.

[19] M. H. Farzaneh, S. Kugele, and A. Knoll, "A graphical modeling tool supporting automated schedule synthesis for time-sensitive networking," in *Proc. 22nd IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2017, pp. 1–8, doi: 10.1109/ETFA.2017.8247599.

[20] Q. Yu, H. Wan, X. Zhao, Y. Gao, and M. Gu, "Online scheduling for dynamic VM migration in multicast time-sensitive networks," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 3778–3788, Jun. 2020, doi: 10.1109/TII.2019.2925538.

[21] Q. Yu and M. Gu, "Adaptive group routing and scheduling in multicast time-sensitive networks," *IEEE Access*, vol. 8, pp. 37855–37865, 2020, doi: 10.1109/ACCESS.2020.2974580.

[22] L. Xu, Q. Xu, Y. Zhang, J. Zhang, and C. Chen, "Co-design approach of scheduling and routing in time sensitive networking," in *Proc. IEEE Conf. Ind. Cyberphys. Syst. (ICPS)*, Jun. 2020, pp. 111–116, doi: 10.1109/ICPS48405.2020.9274702.

[23] N. G. Nayak, F. Dürr, and K. Rothermel, "Time-sensitive software-defined network (TSSDN) for real-time applications," in *Proc. 24th Int. Conf. Real-Time Netw. Syst. (RTNS)*, 2016, pp. 19–21, doi: 10.1145/2997465.2997487.

[24] N. G. Nayak, F. Dürr, and K. Rothermel, "Incremental flow scheduling and routing in time-sensitive software-defined networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2066–2075, May 2018, doi: 10.1109/TII.2017.2782235.

[25] Z. Pang, X. Huang, Z. Li, S. Zhang, Y. Xu, H. Wan, and X. Zhao, "Flow scheduling for conflict-free network updates in time-sensitive software-defined networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 3, pp. 1668–1678, Mar. 2021, doi: 10.1109/TII.2020.2998224.

[26] V. Gavriluţ, L. Zhao, M. L. Raagaard, and P. Pop, "AVB-aware routing and scheduling of time-triggered traffic for TSN," *IEEE Access*, vol. 6, pp. 75229–75243, 2018, doi: 10.1109/ACCESS.2018.2883644.

[27] V. Gavrilut and P. Pop, "Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications," in *Proc. 14th IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, Jun. 2018, pp. 1–4, doi: 10.1109/WFCS.2018.8402374.

[28] J. Zhang, Q. Xu, X. Lu, Y. Zhang, and C. Chen, "Coordinated data transmission in time-sensitive networking for mixed time-sensitive applications," in *Proc. 46th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2020, pp. 3805–3810, doi: 10.1109/IECON43393.2020.9254565.

[29] J. Ren, D. Yang, E. Cui, and K. Gong, "An analytical latency model for AVB traffic in TSN considering time-triggered traffic," in *Proc. Int. Conf. Commun. Technol. (ICCT)*, Oct. 2020, pp. 938–943, doi: 10.1109/ICCT50939.2020.9295841.

[30] N. Diaz and N. Gautam, "On latency for non-scheduled traffic in TSN," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.

[31] L. Zhao, P. Pop, Z. Zheng, and Q. Li, "Timing analysis of AVB traffic in TSN networks using network calculus," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2018, pp. 25–36, doi: 10.1109/RTAS.2018.00009.

[32] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks," in *Proc. 24th Int. Conf. Real-Time Netw. Syst. (RTNS)*, 2016, pp. 183–192, doi: 10.1145/2997465.2997470.

[33] D. Hellmanns, J. Falk, and A. Glavackij, "On the performance of stream-based, class-based time-aware shaping and frame preemption in TSN," in *Proc. IEEE Int. Conf. Ind. Technol.*, Feb. 2020, pp. 298–303.

[34] N. Reusch, L. Zhao, S. S. Craciunas, and P. Pop, "Window-based schedule synthesis for industrial IEEE 802.1Qbv TSN networks," in *Proc. 16th IEEE Int. Conf. Factory Commun. Syst. (WFCS)*, Apr. 2020, pp. 14–17.

[35] W. Steiner, P. G. Peon, M. Gutierrez, A. Mehmed, G. Rodriguez-Navas, E. Lisova, and F. Pozo, "Next generation real-time networks based on IT technologies," in *Proc. IEEE 21st Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2016, pp. 1–8, doi: 10.1109/ETFA.2016.7733580.

[36] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse, "System architecture evaluation using modular performance analysis: A case study," *Int. J. Softw. Tools Technol. Transf.*, vol. 8, no. 6, pp. 649–667, 2006, doi: 10.1007/s10009-006-0019-5.

[37] M. H. Farzaneh and A. Knoll, "An ontology-based plug- and-play approach for in-vehicle time-sensitive networking (TSN)," in *Proc. IEEE 7th Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Oct. 2016, pp. 1–8, doi: 10.1109/IEMCON.2016.7746299.

[38] E. Wanderler and L. Thiele. (2006). *Real-Time Calculus (RTC) Toolbox*. [Online]. Available: https://www.mpa.ethz.ch/Rtctoolbox

KHALED M. SHALGHUM (Student Member, IEEE) received the B.Sc. and M.Sc. degrees in communication engineering from the Electrical and Electronic Engineering Department, University of Tripoli, Tripoli, Libya, in 2000 and 2008, respectively. Currently, he is pursuing the Ph.D. degree with Universiti Putra Malaysia (UPM). From 2002 to 2008, he worked as an Engineer and later as a Lecturer at the Faculty of Technical Engineering, Mesalata, Libya. From 2010 to 2018, he worked as a Lecturer at the Electrical and Electronic Engineering Department, Azzaytuna University, Tarhuna, Libya. In 2018, he was offered a scholarship to complete his graduate study with UPM from Libyan Ministry of Education. His research interests include wireless communications, broadband wireless access networks, WiMAX technology, time-sensitive networks, and latency performance evaluation.

NOR K. NOORDIN (Senior Member, IEEE) received the B.Sc. degree in electrical engineering from The University of Alabama, USA, in 1987, the M.E. degree from Universiti Teknologi Malaysia, and the Ph.D. degree from Universiti Putra Malaysia. She is currently working as a Professor with the Department of Computer and Communication System Engineering, Universiti Putra Malaysia. She has published more than 300 journals, book chapters, and conference papers. She has led many research projects. Her research interests include wireless communications and network systems.

ADUWATI SALI (Senior Member, IEEE) received the B.E. degree in electrical electronics engineering from The University of Edinburgh, U.K., in 1999, the M.Sc. degree in communications and networks engineering from Universiti Putra Malaysia, in 2002, and the Ph.D. degree in mobile and satellite communications from the University of Surrey, U.K., in 2009. She was the Deputy Director of the Research Management Centre (RMC), Universiti Putra Malaysia, from 2016 to 2019. She is currently a Professor with the Department of Computer and Communication System Engineering and a Researcher with the Wireless and Photonic Network Research Center of Excellence (WiPNET), Universiti Putra Malaysia. Her research interests include mobile and satellite communication systems.

FAZIRULHISYAM HASHIM (Member, IEEE) received the M.Sc. degree from Universiti Sains Malaysia and the Ph.D. degree in telecommunication engineering from The University of Sydney, Australia. He is currently an Associate Professor with the Department of Computer and Communication Systems Engineering, Universiti Putra Malaysia. His research interests include heterogeneous wireless communication systems and network security.