Fixed-Priority Scheduling and Controller Co-Design for Time-Sensitive Networks

Xiaotian Dai[1], Shuai Zhao[1], Yu Jiang[2], Xun Jiao[3], Xiaobo Sharon Hu[4], Wanli Chang[1]
[1]University of York, UK [2]Tsinghua University, China [3]Villanova University, USA [4]University of Notre Dame, USA
[1]{xiaotian.dai, shuai.zhao, wanli.chang}@york.ac.uk
[2]jy1989@mail.tsinghua.edu.cn [3]xun.jiao@villanova.edu [4]shu@nd.edu

ABSTRACT

Time-sensitive networking (TSN) is a set of standardised communication protocols developed under the IEEE 802.1 working group. TSN aims to support deterministic communication based on network schedules that are distributively configured. It is widely considered as the future in-vehicle network solution for highly automated driving, where the requirement on timing guarantee is alongside the demand of high communication bandwidth. In this work, we study a setting of periodic control and non-control packets, with implicit and arbitrary deadlines, respectively. As the FIFO (first-in, first-out) queues in the 802.1Qbv switch incur long delay in the worst case, which prevents the control tasks from achieving short sampling periods and thus impedes control performance optimisation, we propose the first fixed-priority scheduling (FPS) approach for TSN by leveraging its gate control features. In this context, we develop a finer-grained frame-level response time analysis, which provides a tighter bound than the conventional packet-level analysis. Building upon FPS and the above analysis, we formulate a co-design optimisation problem to decide the sampling periods and poles of real-time controllers with settling time as the objective to minimise, whilst satisfying the schedulability constraint.

CCS CONCEPTS

• **Computer systems organisation → Embedded and cyber-physical systems**; **Real-time systems**; • **Networks → Network protocols**.

KEYWORDS

Time-sensitive networks, real-time communication, fixed-priority scheduling, scheduling-controller co-design

ACM Reference Format:
Xiaotian Dai, Shuai Zhao, Yu Jiang, Xun Jiao, Xiaobo Sharon Hu, Wanli Chang. 2020. Fixed-Priority Scheduling and Controller Co-Design for Time-Sensitive Networks. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '20), November 2–5, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3400302.3415715

1 INTRODUCTION

In recent years, Ethernet, as a data link layer protocol, has evolved from standard computer networks to applications of industrial automation (e.g., PROFINET, EtherNet/IP, EtherCat, and Sercos III [18]), aerospace (e.g., AFDX [3]), energy and power (e.g., IEC 61850 over HSR/PRP), as well as in-vehicle communication (e.g., deterministic real-time Ethernet [19]). In the emerging safety-critical systems such as highly automated vehicles, a large volume of messages with mixed types need to be transmitted on the same infrastructure, which requires deterministic and predictable timing to guarantee safety. Traditional real-time networks use non-standard Ethernet to enable high-bandwidth deterministic communication, which prohibits connectivity between different protocols and components from different vendors, as well as increases uncertainty and difficulty in timing and hazard analysis. Therefore, a network protocol is demanded to satisfy requirements on both deterministic end-to-end latency and high traffic load.

Time-sensitive networking (TSN), proposed as an IEEE standard, offers an interoperable and flexible deterministic Ethernet-based solution [12]. It is widely considered as the network solution for future automobiles. The IEEE 802.1 TSN standard includes a wide range of subsets, in which one of the most important protocols is the 802.1Qbv [5, 11, 20]. The IEEE 802.1Qbv supports time-aware shaper (TAS) using TDMA (time-division multiple access)-scheduled queues to access the egress port — controlled by a gate switching logic that is driven by a synchronised global timer and a look-up scheduling table.

Control loops are often involved in the safety-critical systems, where guarantees are required on both timing of communication and control performances (measured by settling time). In general, short sampling periods enable the potential to achieve good control performance with frequent interactions between the controller and the plant. The state-of-the-art network scheduling techniques for TSN (e.g., [2, 13, 20]) cannot be directly applied, as they consider neither the hard real-time constraints on network packets nor the control performance of the system.

Main contributions: To address the above needs, we present the first integrated solution of network scheduling and controller co-design for TSN 802.1Qbv. Both periodic control and non-control packets are considered to best capture the real-world scenarios. The goal is to achieve timing predictability and maximise the control performance. Specifically, as the FIFO (first-in, first-out) queues in the 802.1Qbv switch incur long delay in the worst case, which prevents the control tasks from achieving short sampling periods and thus impedes control performance optimisation, we propose the first fixed-priority scheduling (FPS) approach for TSN. The schedules computed offline can be implemented in the gate control list of the switch and executed accordingly. We develop a finer-grained analysis for the proposed scheduling approach at the frame level, which bounds the worst-case response time of the network packets more tightly than the conventional packet-level analysis. Based on FPS and the analysis, we formulate a co-design optimisation problem to decide the sampling periods and poles of real-time controllers. The goal is to maximise the overall control performance measured by settling time, whilst satisfying the schedulability constraint.

Organisation of the paper: A brief background of TSN is given in Section 2, which includes the internal structure of a switch and the necessary notions; Section 3 presents the frame-level FPS method, as well as a corresponding schedulability analysis; In Section 4, the real-time controller design under performance and network constraints is discussed. Finally, evaluation is reported in Section 5, followed by some remarks in Section 6 to conclude the paper.

2 BACKGROUND ON TIME-SENSITIVE NETWORKING

Time-sensitive networking is an enabler for Ethernet-based communication services that were not originally built to support hard real-time guarantees, such as OPC Unified Architecture (OPC-UA)[1] and Distributed Data Service (DDS)[2]. The objective of TSN is to reduce the worst-case end-to-end latency for critical traffics. Here we briefly discuss the IEEE 802.1Qbv TSN (referred to as Qbv in the following text). A diagrammatic view of a Qbv-enabled switch is depicted in Figure 1. From the figure, it can be seen that a Qbv TSN switch consists of the following major components:

- **Scheduled FIFO queues**: In a Qbv-enabled TSN switch, there are eight independent time-divided FIFO queues which are controlled by transmission gates. The incoming traffic is filtered by the packet filtering unit which sends a packet to its designated queue. This information is encoded as Class of Service (CoS) in the priority code point (PCP) header in the Ethernet frame.
- **Gate control list (GCL)**: The GCL can trigger gate-open and gate-close events periodically with a gate control cycle. The time granularity between events can be as low as $1ns$ depending on the specific implementation. The schedule is located in a GCL look-up table that is distributively configured to each TSN node. If multiple gates are opened at the same time, the policy in the priority selection unit will determine which queue is forwarded to the egress port first.
- **Time synchronisation**: To allow time-divided transmission that is distributed through the network, a timer is globally synchronised with all the switches in the same network using precision time protocols (PTPs), e.g., IEEE 802.1AS or IEEE 802.1AS-Rev.

The mechanisms of Qbv TSN improve the flexibility in terms of traffic schedule and control. It enables interoperability between standard-compliant industrial devices thus allowing open data exchange. It also removes the need for physical separation of critical and non-critical communication networks. However, in a different

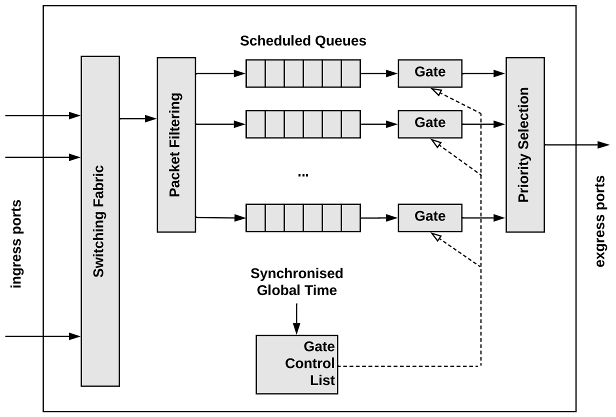

Figure 1: An overview structure of a 802.1Qbv-capable TSN switch.

aspect these introduce increased design complexity that needs to be elaborately handled.

3 TRAFFIC SCHEDULING OF TSN IN CONTROL SYSTEMS

In this work, we propose an integrated solution that solves the controller-network co-design problem. Scheduling on a single TSN switch is considered and can be extended to the entire network. As we focus on the scheduling aspect, it is assumed the network communication is ideal: (i) the depth of the queues is sufficient, i.e., no traffic overflows; (ii) the channel is error-free and has a constant transmission rate. These ease the analysis and helps to understand the nature of the problem. Relaxing them in practice needs limited modifications and will be discussed in future work. The network is subjected to two basic traffic types: scheduled and unscheduled traffic, depending on a certain level of quality-of-service (QoS) is required or not. In this work, we focus on scheduled traffic and leave unscheduled traffic be transmitted using residual bandwidth with best effort.

TSN provides time synchronisation and time-division transmission, which enables global scheduling through GCLs [20]. Although the schedule of TSN can be designed by hand, it soon becomes impractical as the network turns complex and more packets are added to the network. In this section, we specify the scheduling policy adopted for TSN while control systems are considered. The proposed schedule minimises the blocking of packets (including ones sent by control tasks), to improve schedulability and control performance. We then introduce a fine-grained response time analysis that bounds the worst-case latency of packets in a single Qbv switch. Below we first discuss the system model.

System model: The system contains N periodic packets[3] $\Gamma = \{\tau_1, \tau_2, ..., \tau_N\}$, including both control (Γ_c) and non-control packets (Γ_{nc}) sent by tasks from the application. Each packet τ_i is modelled as a 7-tuple $\{L_i, C_i, T_i, D_i, P_i, R_i, \Lambda_i\}$, representing the worst-case length of the packet L_i, transmission time C_i, period T_i, deadline D_i,

[1]https://opcfoundation.org/about/opc-technologies/opc-ua/
[2]https://www.omg.org/spec/DDS/1.4/PDF

[3]Continuously released periodic packets will form a flow. For simplicity, we use these two terms interchangeably.

priority P_i, worst-case latency R_i and the set of frames Λ_i in each release, respectively. Frames are transmitted in a non-preemptive fashion. A global packet transmission rate v is applied to all packets, thus $C_i = L_i/v$ for τ_i. Each control packet is assigned with an implicit deadline i.e., $D_i = T_i$. To provide a more general network model for the system, the non-control packets can have arbitrary deadlines without any constraint imposed. As a consequence, at a given time instant there could be several instances of a non-control packet waiting for transmission in the switch. The priorities of all packets are assigned according to the deadline monotonic algorithm ($P_i > P_j$ if $D_i < D_j$), and each packet has a unique priority. In addition, the Maximum Transmission Unit (MTU) is considered, denoted as M, which defines the maximum data size allowed in a single transmission. For the ease of presentation, we denote M as the transmission time for sending data with a size equal to one MTU. Thus, each packet could be divided into a set of successive frames, i.e., $\Lambda_i = \{\lambda_i^1, \lambda_i^2, ..., \lambda_i^m\}$, with $m = \lceil L_i/M \rceil$. For a given frame λ_i^j, it inherits the analytical properties of τ_i (i.e., T_i, D_i and P_i), and has its own data length, L_i^j, and transmission time, C_i^j.

3.1 Scheduling Network Packets in TSN

In a typical Qbv switch, the network packets are queued by their arriving time (i.e., FIFO queuing) and are transmitted non-preemptively [11]. Traditionally, the synthesis of GCL schedule is performed using Satisfiability Modulo Theories (SMT) [5, 13] or Integer Linear Programming (ILP) [2]. The defined end-to-end latency imposes zero-jitter, however, with significantly reduced solution space. The scheduling in TSN networks with Quality-of-Service (QoS) requirements can be either performed at the queue level [20] or packet level [14]. With the queue-level scheduling, each FIFO queue in the Qbv switch is assigned with a priority, and packets in a queue with a higher priority are always transmitted first. However, as packets in each queue are transmitted strictly in a FIFO order, packets under the queue-level scheduling approach can incur substantial blocking, where packets with a tighter deadline but at the end of a queue cannot be favoured. That is, with the queue-level scheduling, packets with different deadlines in the same FIFO queue are treated equally without concerning individual temporal requirements. For control systems, such a scheduling is not appropriate, as the delay for transmitting control packets can introduce significant impact on the control performance of the system. Thus, the packet-level (more precisely, the frame-level) scheduling is adopted to provide a finer-grained schedule, where each packet (and its frames) is scheduled strictly by its priority.

However, even with the packet-level scheduling, packets can still incur additional delay due to the FIFO queuing, as the actual transmission largely depends on the arriving time of the packets. In the worst-case, a late-arrived packet with a high priority can be blocked by all the released packets with lower priorities. To minimise the delay due to FIFO queuing, an alternative is to perform the scheduling off-line (i.e., prior to execution), with the complete knowledge of all packets in the system[4]. The offline scheduling can be performed by assuming all packets are arrived at the same time, with a packets transmission order obtained based on their priorities. If packets have different arrival times during run-time, a simple mechanism that defers the queuing of the early-arrived low-priority frames can be adopted, to maintain the queuing order obtained from the offline FPS-NP without imposing extra latency to packet transmission (see Section 3.2 for deferred queuing). By maintaining the offline packets transmission order during run-time, the blocking time of each packet during transmission can be minimised to one frame only, i.e., identical to the classic non-preemptive fixed-priority scheduling (FPS-NP) [8].

[4]Such an approach is feasible as the packets are deterministic i.e., the packets sent by each task are known a prior with periodic release.

Based on the above discussion, to provide a fine-grained schedule and to minimise the delay due to the queuing problems, the scheduling adopted in this work is conducted before runtime on the frames of each packet in one hyper-period, with the scheduling decisions encoded into the GCL. Once a schedule is obtained, the frames can be statically allocated to the FIFO queues according to the schedule while the scheduling decisions can be mapped to the GCL to control the gates of all queues to achieve the desired execution order. To this end, the scheduling on TSN can be successfully mapped to the traditional FPS-NP, in which each packet is scheduled strictly by its priority and can be blocked maximum once during the entire transmission.

With the described scheduling approach, we avoid the packets queuing problem and can achieve the minimised delay for all packets, in the context of a Qbv switch. This is crucial for control systems as the resulting control performance can be affected by transmission delay for the control packets. The experimental results given in Section 5 provide evidence for this claim and demonstrate the impact of packets transmission delay on the system schedulability and control performance. To our best knowledge, this is one of the earliest work targeting at control systems in which the timeliness and performance are sensitive to the transmission delay of certain critical (i.e., control and non-control) packets. For the non-control packets, meeting their timing requirements is essential for guaranteeing the system correctness, whereas minimising transmission delay of the time-triggered control packets are essential crucial for control performance.

For unscheduled packet flows that do not have a temporal requirements, the traffics can be scheduled using residual bandwidth left by the critical traffics with time-aware shapers [16, 17] and queue partitioning. Supporting such flows has been well-described by the above work, and will not be re-presented in this paper. Targeting at such systems, a complete scheduling solution is proposed that minimises the transmission delay for all packets, in the context of the TSN Qbv switch. Last but not least, different from [5], our approach makes no assumption on the isolation of incoming packets and the construction of the GCL, e.g., isolating certain queues for a specific packet type, to provide a more general approach for using TSN in control systems.

3.2 Deferred Queue

As described in Section 3.1, for packets with different arriving times, a mechanism is required to delay the queuing of the early-arrived low priority packets so that the minimised blocking can be guaranteed. To achieve this, a *deferred queue* with priority ordering is introduced into the Qbv switch, which is integrated into the packet filtering unit (see Figure 1) for holding early-arrived packets temporarily, until they can be added into the scheduled queues with a correct order.

Assuming simultaneous release for all packets at the start of the system, the offline FPS-NP schedule can produce a well-planned transmission order for all packet instances released in one hyperperiod, in which each packet (a set of successive frames) is scheduled strictly based on priority. For this schedule, the blocking of each packet is minimised, as in the worst case, the ready packet with the highest priority can start transmitting after the currently transmitting frame of a low priority packet has completed. During run-time, this offline scheduling order is encoded into the priority filtering unit, which provides a reference of the expect order for incoming packets.

For each incoming packets, the priority filter examines whether this packets arrives by the expected order, i.e., all its previous packets with a higher priority have arrived. If so, this packet is dispatched to the scheduled queues immediately, at which it will be select to transmit by GCL. Otherwise (i.e., certain previous high priority packets haven't arrived yet), this packet is hold by the priority filter until (a) the missing packets arrives or (b) the scheduled queues are empty and this packet has the highest priority among all the deferred packets.

Note that the condition (b) can lead to a transmission order different from the expected one, as certain packets can be transmitted before a late-arriving higher priority packet. However, this does not introduce extra delay and can help increasing the throughput. With the deferred queuing, it is possible that all scheduled queues are empty while some packets are stored in the priority filter. Under this situation, the priority filter selects the packet at the head of the queue (i.e., with the highest priority) and send its frames into the scheduled queue for a direct transmission one by one, until a higher priority packet arrives. This guarantees that the transmission never stops as long as there exist waiting packets (either in the priority filter or the scheduled queues). In addition, for the late arriving high priority packet, its blocking is still at most C_i^j, where it can be transmitted directly after the currently-transmitting frame.

3.3 Worst-Case Response Time Analysis

With the scheduling in TSN mapped to the traditional FPS-NP, the worst-case response time for transmitting a packet in a single Qbv switch can be obtained, which bounds the time duration from when the packet enters into the switch to when the packet is transmitted. Due to the different deadline constraints of the control and non-control packets (i.e., implicit and arbitrary deadlines respectively), different analysis techniques are applied for each packet type. However, as both control and non-control packets are scheduled strictly by the FPS-NP, the basic philosophy for analysing both types of packets is similar to that in [8], but with modifications and improvements in order to reflect the unique features of the Qbv switch and to support the analysis at the frame level.

The response time equation of a packet τ_i is given in the following equation for both control and non-control packets:

$$R_i = \max_{\forall \lambda_i^j \in \Lambda_i} \begin{cases} R_i^j(0), & \text{if } \tau_i \in \Gamma_c \\ \max_{n=0\ldots\left\lceil \frac{t_i+J_i}{T_i} \right\rceil -1} \left(R_i^j(n)\right), & \text{if } \tau_i \in \Gamma_{nc} \end{cases} \quad (1)$$

In Equation (1), $R_i^j(n)$ denotes the response time for transmitting the nth instance of frame λ_i^j in τ_i's busy period t_i, and J_i denotes the queuing time, i.e., the time window from when the first frame of τ_i reaches the Qbv Switch, until when the last frame is queued. $\left\lceil \frac{t_i+J_i}{T_i} \right\rceil$ gives the total number of times that a non-control packet can be sent within its busy period [8].

The analysis of a control packet is relatively straight forward, as at any given time, there can only exist one instance of a control packet in the system i.e., implicit deadlines. Thus, the worst-case response time of a control packet can be safely bounded by computing the maximum response time of all its frames[5]. However, for a non-control packet, multiple instances of each of its frames can co-exist due to the arbitrary deadline. Thus, the response time of a frame (with an arbitrary deadline) must be obtained by computing the maximum response time of all its instances within the busy period t_i.

Similar to [8], the busy period of a non-control packet is computed by Equation (2), where B_i gives the worst-case blocking that τ_i can experience due to transmitting a low priority frame and $hep(i)$ refers to all indices of packets that have equal or higher priorities than P_i, including i. The recursive calculation can starts with $t_i = B_i + C_i$, and is guaranteed to converge [8], given that the total utilization for packets in $hep(i)$ is less than 1, i.e., $\sum_{\forall j \in hep(i)} (C_j/T_j) \le 1$. We later decompose B_i in Equation (5).

$$t_i = B_i + \sum_{\forall k \in hep(i)} \left\lceil \frac{t_i + J_k}{T_k} \right\rceil C_k \quad (2)$$

The response time of a frame is bounded by Equation (3), in which J_i^j denotes the time to en-queue frame λ_i^j, W_i^j gives the maximum queuing delay that λ_i^j can incur in a FIFO queue before it is selected to be transmitted and C_i^j denotes its transmission time. The time for queuing λ_i^j into a FIFO queue also contains the enqueue time of frames of τ_i that are prior to λ_i^j in one transmission. In addition, for the non-control frames, $n \cdot T_i$ is subtracted as this is the arrival time of its nth instance, relative to the start of the busy period. Note, for control frames, n is always 0.

$$R_i^j(n) = \sum_{q \in [1,j]} J_i^q + W_i^j(n) + C_i^j - n \cdot T_i \quad (3)$$

Equation (4) gives the queuing delay W_i^j of frame λ_i^j, where $hp(i)$ returns a set of packets with a priority strictly higher than P_i. This equation is also applicable to either control or non-control frames, with $n = 0$ for all control frames. Figure 2 provides an example illustrating the worst-case delay of the third ($n = 2$) instance of the second frame (i.e., $j = 2$) in packet τ_i. As shown in the figure, in the worst case, the frame (in red) has to wait for five types of other frames to transmit before it can start, which are mapped to four types of delay, as follows. In the worst case, a frame can incur four sources of delay when waiting in a FIFO queue: (i) the blocking caused by a low-priority frame that is currently transmitting i.e., B_i; (ii) the delay by τ_i's frames prior to λ_i^j (with potential existence

[5]From Equation 1, the response time of a packet equals to the response time of its last frame in each transmission, which takes into account the delay for transmitting the previous frames in one transmission.

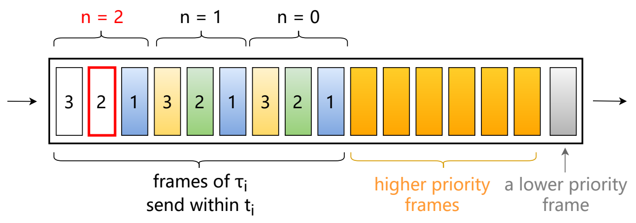

Figure 2: The worst-case delay of a frame, which is caused by a low priority frame (in grey), high priority packets (in orange), instances of τ_i's frames prior to λ_i^j (in blue), previous instances of λ_i^j (in green) and previous instance of τ_i's frames after λ_i^j (in yellow).

of multiple instances); (iii) the delay by previous instances of λ_i^j and the frames after λ_i^j in each τ_i's instance sent before λ_i^j; and (iv) the interference from the frames of each packet with a higher priority than P_i. Note that (iii) accounts for the delay cause by both the previous instances of λ_i^j itself and the frames after λ_i^j in previous instances. These delays are captured by the equation respectively.

$$
\begin{aligned}
W_i^j(n) =& B_i + (n+1) \cdot \sum_{q \in [1, j-1]} C_i^q + n \cdot \sum_{q \in [j, |\Lambda_i|]} C_i^q \\
&+ \sum_{\forall \lambda_k^q \in \Lambda_k, \forall k \in hp(i)} \left\lceil \frac{W_i^j(n) + J_k^q}{T_k} \right\rceil C_k^q
\end{aligned} \tag{4}
$$

Finally, B_i is given by Equation (5), where $lp(i)$ returns the packets with a priority lower than P_i. The maximum blocking time that τ_i (and any of its frames) can incur is the longest transmission time among the frames of all the lower priority packets.

$$
B_i = \max_{\forall \lambda_k^q \in \Lambda_k, \forall k \in lp(i)} (C_k^q) \tag{5}
$$

Equation (1-5) summarises the response time analysis for bounding the worst-case transmission latency (i.e., the response time) of packets in a Qbv switch for time-critical control systems. The analysis considers both implicit and arbitrary deadlines for different packet types and is fine-grained, which provides the worst-case transmission latency of each frame. Arguably, by intuition, a trivial modification that treats each frame as an independent task can be applied in an existing packet-level analysis (e.g., the one in [9]), to support the analysis at the frame-level. However, additional techniques are still required to guarantee the correct transmission order between frames that belong to the same packet and instance so that the transmission time of each individual frame can be obtained. This is achieved in our analysis by Equation (4), which carefully examines the transmission order of different types of frames (including the ones in τ_i) and provides a tighter upper bound compared to a packet-level analysis.

The proposed analysis and scheduling techniques for a single switch can be extended to support the network topology level with multiple switches and end-nodes. For the proposed method, it can be implemented in each switch. For a given switch, the proposed schedule takes all packets that will go through this switch and then produced a static schedule. In addition, the deferred queue is applied in each switch to handle the case in which low priority packets arrive earlier than expected. To compute the end-to-end worst-case transmission time of a packet τ_i that travels through more than one switches, the input packets of each of the switches should be given and the worst-case delay of τ_i in each switch can be effectively upper bounded by summing the worst-case delay it can incur in each switch by the above analysis.

However, with only one switch, the worst-case delay of a packet can be bounded by considering all the input packets with a synchronous release at the begin of the system. This assumption, however, may not hold in the scenario of multiple switches, in which the actual arrival time of a packet at a given switch depends on the delay it incurs at the previous switches. Thus, the analysing approach above would contain certain degree of pessimism as not all the input packets in a switch will cause a delay on τ_i, depending on their arrival times. In addition, as an offline scheduling technique, the proposed schedule would incur scalability issues when the number of switches and nodes increases. These are identified as desirable research directions that will be addressed in future work.

4 CONTROLLER SYNTHESIS AND PERIOD ALLOCATION

For a safety-critical autonomous system, for example, a self-driving car, the control functions are crucial and should always be a major concern. Further to the introduced scheduling and analysis that guarantee the timing of control packets, a well-designed controller is also required, in order to satisfy the control performance requirement and even maximise it under the schedulability constraint of the network.

Most real-time controllers targeting settling time (which will be formally defined later in this section) can run at different frequencies [1, 6, 7]. In the TSN context, this rate is bounded by (i) the maximum transmission capability; (ii) the lowest control performance requirement. Hence, there exists an optimised operational point that would produce acceptable network schedulability with maximised control performance.

4.1 Control Model

For a linear-time-invariant (LTI) controlled plant, its system dynamics can be described using the following differential equations:

$$
\begin{aligned}
\dot{x}(t) &= Ax(t) + Bu(t), \\
y(t) &= Hx(t)
\end{aligned} \tag{6}
$$

in which A, B and H are system matrices that represent the system physical properties; $x(t)$ is the system state(s); $y(t)$ is the system output(s) and $u(t)$ is the control input(s). Assuming the sampling time is T_s and the sensor-to-actuator delay is within one sampling period, at discrete time instant k, the system dynamics evolve with the following equations:

$$
\begin{aligned}
x(k+1) &= A_d x(k) + B_d u(k-1), \\
y(k) &= Hx(k)
\end{aligned} \tag{7}
$$

where $u(-1) = 0$ for $k = 0$ and

$$
A_d = e^{A \cdot T_s}, B_d = \int_0^{T_s} e^{A\tau} d\tau \cdot B \tag{8}
$$

To further simplify the equation, define an augmented variable z as: $z(k) = \begin{bmatrix} x(k) & u(k-1) \end{bmatrix}^T$, and substitute $x(k)$, $u(k)$ with $z(k)$ in Equation (7):

$$z(k+1) = \begin{bmatrix} A_d & B_d \\ 0 & 0 \end{bmatrix} z(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \tag{9}$$

Assuming a full state-feedback controller is used, the control input $u(k)$ is calculated by:

$$u(k) = -Kz(k) + Fr(k) \tag{10}$$

where K is the feedback gain, F is the feedforward gain and $r(k)$ is the reference. By combining Equation (9) and (10), the system equation therefore becomes:

$$z(k+1) = \underbrace{(A_d - B_d K)}_{A_{cl}} z(k) + B_d F r(k) \tag{11}$$

To satisfy control stability, all the eigenvalues of the closed loop dynamic matrix, i.e. A_{cl} in Equation (11), have to be inside the unit circle. The exact value of A_d and B_d is dependent on the sampling period T_s as seen from Equation (8), which is equal to the period of the control packet, T_i. This control model will be used through the rest of this paper.

4.2 Problem Definition

We use settling time (t_s) as the index of quality-of-control (QoC), which is widely used in control engineering as a compulsory design requirement [4]. Settling time is defined as the time duration from when a control system is subjected to a disturbance to when it enters steady-state, i.e., the current output has reached and stays within 5% deviation of the targeted output. There is an upper bound requirement on the settling time, e.g., the settling time of a control system should not be longer than 0.5 seconds.

Finding an optimal period is crucial for (i) guaranteeing the performance of the controller itself; and (ii) ensuring enough residual time slots for non-control-related packets so they can also meet their deadlines. Based on the aforementioned objectives and constraints, the period assignment problem can be solved as an optimization problem, which is formulated as follows:

$$\begin{aligned} \text{minimise} \quad & \mathcal{J} = \sum_{\mathbb{D}} w_j \cdot t^*_{s,j} \\ \text{subject to} \quad & R_i \le D_i \\ & t_{s,j} \le t^+_{s,j} \\ & |u_j(k)| \le u_{max} \\ & T_i = n \cdot t_{gcd}, n \in N^+ \\ \text{where} \quad & i \in \Gamma, \quad j \in \Gamma_c \end{aligned} \tag{12}$$

where $w_j \in (0,1]$ is the weight (i.e., relative importance) of the corresponding control task and $\sum w_j = 1$; $t^*_{s,j} \in [0,1]$ is the normalised settling time of the jth controller; $\mathbb{D}$ represents the solution space of all poles that can ensure control stability; $t_{s,j}$ is the settling time of the jth controller, and $t^+_{s,j}$ is the maximum allowed settling time; $u_j(k)$ is input at discrete instance k, which is constrained by u_{max} as the maximum input threshold; The last constraint defines the time-granularity of a feasible period. To benefit from harmonic periods and to reduce the size of the GCL table, each T_i must be an integer multiple of t_{gcd}, the greatest common divisor of all the packet periods. This is in accordance with common practice.

4.3 Solving the Network and Control Co-Design Problem

In a typical control application, while the periods of non-control-related packets are inflexible, the control-related packets often have adjustable periods. This additional flexibility allows fine tuning of controller periods to achieve the best overall performance (defined as in Equation (12)). To solve the defined problem, a controller's period and its corresponding parameters under that period both have to be decided. These two steps are dependent on each other but can be decomposed into two sub-problems, i.e., the optimization process needs to (i) find the feasible periods that can satisfy schedulability constraints; (ii) find the controller parameters under the feasible periods that would satisfy control stability and minimal performance requirement, and on top of that, maximise the control performance as much as possible.

For the first problem, due to the existence of harmonic periods and that the number of control tasks is often small, the search space is manageable and thus can be solved through exhaustive search. For larger scale problems, heuristic methods can be used instead to find the feasible period configurations.

For the second problem, as pole placement for the minimum settling time under input constraints is a non-convex and non-linear problem, the solution space cannot be searched easily. We use Particle Swarm Optimisation (PSO) to find the optimal controller parameters (by pole placement [4]) under certain sampling period that can minimise the settling time, while given the control performance and input saturation as constraints. PSO is a population-based optimization approach for iterative improvement of candidate solutions given a non-linear non-convex objective function and a metric of quality [15].

The optimization process is given in Algorithm 1. The solution space is first formulated in Line 4. The schedulability is then tested (Line 6) to obtain potential period configurations, and under each period configuration, the optimal poles of each control task can be found through PSO (Line 8). To speedup the process, the optimal poles under the feasible range of periods can be obtained in advance. The identified configuration is appended into the feasible solutions provided that the minimum control performance and the input constraints are both satisfied (Line 10-13). Finally, the best candidate that has the minimum $\mathcal{J}$ is selected from all the feasible solutions (Line 16-20). No feasible solution is found if $S^* = \emptyset$, in which case the algorithm fails to find a solution that satisfies all the constraints.

5 EVALUATION

To evaluate the proposed co-design method, experiments are implemented in MATLAB (R2019b) and are running on a desktop PC. The experiment scripts and data can be publicly accessed[6]. To demonstrate the feasibility of the proposed method, we evaluate our approach on synthetically generated packets using UUnifast [10]. The network transmission speed v is 100 Mbps. The greatest

[6]The MATLAB code used in the experiments can be accessed at the following link: https://github.com/automaticdai/research-sched-tsn.

Algorithm 1: Periods and control poles assignment

```
Input: $\Gamma = \{\Gamma_c, \Gamma_{nc}\}$
Output: schedulability, $S^*$
Initialise: feasible and best solutions: $\mathbb{S}^f = \emptyset$, $S^* = \emptyset$
/* construct candidate solutions:                          */
formulate the solution space: $\mathbb{S} = \{S_1, S_2, ..., S_n\}$.
/* explore each candidate:                                 */
for $S_k$ in $\mathbb{S}$ do
    if RTA_schedulability($\Gamma^k$) is True then
        for $j$ in $\Gamma_c^k$ do
            $\{t_{s,j}, u_j\}$ = pso_find_control_parameters($T_j$)
        end
        if $\forall j$ in $\Gamma_c^k : t_{s,j} \leq t^+_{s,j}$ and $|u_j| \leq u_{max}$ then
            $\mathcal{J}_k = \sum w_j \cdot t_{s,j}$
            $S_k \rightarrow \mathbb{S}^f$
        end
    end
end
/* find the best candidate solution:                       */
for $S_k$ in $\mathbb{S}^f$ do
    if $\mathcal{J}_k < \mathcal{J}^*$ then
        $S^* = S_k$
    end
end
/* return feasibility:                                     */
if $S^*$ is not $\emptyset$ then
    return (feasible, $S^*$)
else
    return (infeasible, $\emptyset$)
end
```

common divisor of the periods, t_{gcd}, is 100 μs. The transmission time C_i of all control packets is 120 μs (which can fit into one MTU). The enqueue time J_i is set to be $C_i/100$. The control packets have implicit deadline with $D_i = T_i$, and for non-control packets, the deadline is assigned randomly in [0.5, 1] multiplying its period. The transmission rate of non-control packets ranges from 0.5 *ms* to 200 *ms*, i.e., 5-2000 *Hz*. To control the size of the GCL table, we constrain the period of non-control packets to be within a pre-defined set of harmonic periods (in *ms*): {0.5, 1, 2, 5, 10, 20, 50, 100, 200}. The controlled system consists of a number of direct current (DC) motors which are commonly used in autonomous and robotic systems, e.g., motion control, manipulator and joints. The dynamics of the DC motors are modelled as a second-order plant in the form of time-domain state-space model. In our setup, we have three motor systems with slightly different parameters to control, of which the parameters are given in Table 1. The controller is designed with pole-placement and the system is fully observable and controllable.

5.1 Evaluation of Network Scheduling Performance

A concrete example of a packet set scheduled by the proposed method is given in Table 2. The packets are ordered with their deadlines and the calculated latency is on the last column. It can be seen that the packet set is schedulable as $\forall i : R_i \leq D_i$. To further evaluate the effectiveness of the scheduling model, the proposed method (*P-DM*) is compared with queue-level scheduling. Specifically, we consider the following two policies:

- *Q-RND*: Each queue has its own priority. Packet priority is assigned randomly, i.e., each packet is assigned to a random queue;
- *Q-DM*: Each queue has its own priority. Each packet is assigned to a queue according to its deadline (with deadline monotonic);

For each policy, we have three network scenarios: (i) L – the network is lightly loaded with a total utilisation $\sum U = 0.5$; (ii) M – network is medium-loaded ($\sum U = 0.7$); and (iii) H – network is heavily loaded ($\sum U = 0.9$). Under each scenario, we also have two total numbers of packets: 10 and 20.

For each case, we independently generate 10,000 sets of packets according to the properties introduced at the beginning of this section. To quantify the schedulability, we use schedulable ratio Φ, defined as the number of schedulable packets divided by the total number of packets:

$$\Phi = \frac{\#(schedulable\ packets)}{\#(all\ packets)}$$

Using the random generated packet sets, the result in terms of schedulable packets is given in Table 3. From the result, it can be seen that the random priority policy *Q-RND* can barely find any solution, except a few schedules are found in *L-10*. Comparing *P-DM* with *Q-DM*, the two methods have competitive performance when the network is lightly loaded. For the medium and heavy-load cases, the *P-DM* is significantly better. Comparing *H-10* and *H-20* for *P-DM*, it is observed that the later case produced a higher schedulability. As the total utilization is constrained, this could be due to the fact that shorter interference duration of higher priority packets in (*L/M/H-20*) is preferred to the longer case in our method.

5.2 Control Performance Maximisation

It is common for a control system to have non-control timing-critical traffics that communicate considerable amount of data alongside the control packets. In this evaluation, we consider a real-world scenario in which the non-control packets are pre-defined, and the control packets with flexible transmission rate need to be fitted into the schedule at a later stage. The schedule should not violate

Table 1: Parameters of the controlled plants used in the experiment

Plant	A	B	H	w_i
p_1	$\begin{bmatrix} -10 & 1 \\ -0.02 & -2 \end{bmatrix}$	$\begin{bmatrix} 0 & 2 \end{bmatrix}^T$	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	0.5
p_2	$\begin{bmatrix} -9.167 & 0.833 \\ -0.019 & -1.961 \end{bmatrix}$	$\begin{bmatrix} 0 & 1.961 \end{bmatrix}^T$	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	0.3
p_3	$\begin{bmatrix} -8.571 & 0.714 \\ -0.019 & -1.923 \end{bmatrix}$	$\begin{bmatrix} 0 & 1.923 \end{bmatrix}^T$	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	0.2

Table 2: Compare Packet-Level and Queue-Level Scheduling (unit: μs)

Packet	C_i	T_i	D_i	J_i	$\mathrm{R_i}$
τ_0	37	1,000	598	1	**158**
τ_1	11	1,000	625	1	**169**
τ_2	87	2,000	1,840	1	**256**
τ_3	438	10,000	6,271	5	**700**
τ_4	145	10,000	6,749	2	**841**
τ_5	515	50,000	31,437	6	**1,410**
τ_6	668	50,000	45,357	7	**2,215**
τ_7	183	200,000	124,352	2	**2,390**
τ_8	5,335	200,000	192,926	54	**8,105**

Table 3: Evaluation of Scheduling Policies (index by Φ)

	L-10	*L-20*	*M-10*	*M-20*	*H-10*	*H-20*
P-DM	0.999	1.000	0.992	0.999	0.619	0.816
Q-RND	0.002	—	—	—	—	—
Q-DM	0.924	0.970	0.741	0.808	0.147	0.101

the schedulability of non-control traffics while achieve the highest possible control performance.

Specifically, we study the cases where the network is heavily loaded with a total utilization of non-control packets from 0.70 to 0.95. For each utilisation, 2,000 random generated non-control packets sets are produced. The number of non-control packets is fixed to 10. The input constraint for all controllers is $u_{max} = 24$. The other parameters follow the configurations introduced at the beginning.

The results are shown in Figure 3 and 4. For both figures, the x-axis is the total utilisation of non-control traffics, i.e., $\sum U_{nc}$. Figure 3 represents the percentage of feasible solutions that satisfy both schedulability and control requirements. From the figure it can be seen that as the network load increases, the percentage of feasible solutions drops quadratically but still at an acceptable level. For example, the feasible percentage is above 60% when $U_{nc} = 0.9$. Even for the extreme case when U_{nc} is 0.95, the method still manages to find more than 25% feasible solutions. Figure 4 shows the minimum/average/maximum cost of all the feasible solutions. In the figure, the min/avg/max cost all increases as the network becomes more loaded. However, the average and minimum control cost increases much slower than the maximum, which indicates the effectiveness of the method as the method is still able to find relative good controller periods and poles.

6 CONCLUSION

Timing-sensitive networking provides a potential solution to enhance real-time communication, which satisfies both real-time and high bandwidth requirements. However, the mechanisms of TSN require an elaborate design to make it fully beneficial. In this paper, we proposed an integrated network and control co-design method on IEEE 802.1 Qbv time-sensitive network. This work can be used for automotive and autonomous systems in which the timing determinism in communication contributes a major part in safety assurance

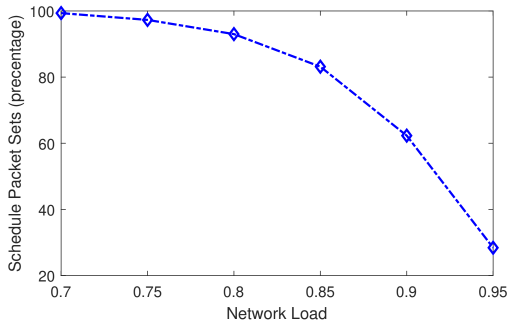

Figure 3: Scheduling performance — percentage of feasible solutions, Φ (y-axis) versus total utilization of non-control traffics, U_{nc} (x-axis).

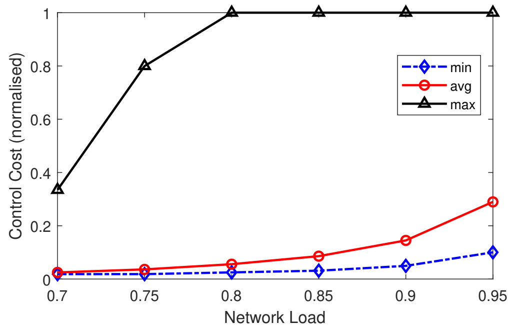

Figure 4: Quality-of-control measured by $\mathcal{J}$ (lower is better) in Equation (12) versus network load of non-control packets U_{nc}. Avg: average; Min: minimum; Max: maximum.

and verification processes. Specifically, we introduced a network scheduling model using non-preemptive fixed-priority scheduling (FPS-NP) and the mapping of the schedule into the TSN gate control list. The schedulability of the network is discussed using non-preemptive response-time analysis with the consideration of multi frames and unconstrained deadlines. An optimisation method is also proposed that could find the feasible solution with maximised overall quality of control constrained by network schedulability. We demonstrated our methods through extensive number of experiments. Future work that further improves this method includes: (i) exploration of dependability, for example, scheduling of flows with dependency modelled using directed acyclic graph (DAG); (ii) extend the scheduling model and analysis to the network topology level with a large number of switches and end-nodes.

REFERENCES

[1] K-E Arzén, Anton Cervin, Johan Eker, and Lui Sha. 2000. An introduction to control and scheduling co-design. In *Proceedings of the 39th IEEE Conference on Decision and Control*, Vol. 5. IEEE, 4865–4870.
[2] Bharat Bansal. 2018. *Divide-and-conquer scheduling for time-sensitive networks.* Master's thesis. University of Stuttgart, Germany.
[3] Henri Bauer, Jean-Luc Scharbarg, and Christian Fraboul. 2010. Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach. *IEEE Transactions on Industrial informatics* 6, 4 (2010), 521–533.
[4] Wanli Chang and Samarjit Chakraborty. 2016. Resource-aware automotive control systems design: A cyber-physical systems approach. *Foundations and*

Trends in Electronic Design Automation 10, 4 (2016), 249–369.

[5] Silviu S Craciunas, Ramon Serna Oliver, Martin Chmelík, and Wilfried Steiner. 2016. Scheduling real-time communication in IEEE 802.1 Qbv time sensitive networks. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*. ACM, 183–192.

[6] Xiaotian Dai and Alan Burns. 2020. Period adaptation of real-time control tasks with fixed-priority scheduling in cyber-physical systems. *Journal of Systems Architecture* 103 (2020), 101691.

[7] Xiaotian Dai, Wanli Chang, Shuai Zhao, and Alan Burns. 2019. A Dual-Mode Strategy for Performance-Maximisation and Resource-Efficient CPS Design. *ACM Transactions on Embedded Computing Systems (TECS)* 18, 5s (2019), 85.

[8] Robert I Davis, Steffen Kollmann, Victor Pollex, and Frank Slomka. 2011. Controller area network (CAN) schedulability analysis with FIFO queues. In *2011 23rd Euromicro Conference on Real-Time Systems*. IEEE, 45–56.

[9] Robert I Davis, Steffen Kollmann, Victor Pollex, and Frank Slomka. 2013. Schedulability analysis for Controller Area Network (CAN) with FIFO queues priority queues and gateways. *Real-Time Systems* 49, 1 (2013), 73–116.

[10] Paul Emberson, Roger Stafford, and Robert I Davis. 2010. Techniques for the synthesis of multiprocessor tasksets. In *proceedings 1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2010)*. 6–11.

[11] IEEE 802.1 Task Group. 2016. *Standard for Local and Metropolitan Area Networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic*. Standard. IEEE.

[12] Stephan Kehrer, Oliver Kleineberg, and Donal Heffernan. 2014. A comparison of fault-tolerance concepts for IEEE 802.1 Time Sensitive Networks (TSN). In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. IEEE, 1–8.

[13] Ramon Serna Oliver, Silviu S Craciunas, and Wilfried Steiner. 2018. IEEE 802.1 Qbv gate control list synthesis using array theory encoding. In *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 13–24.

[14] Giuseppe Piro, Luigi Alfredo Grieco, Gennaro Boggia, Rossella Fortuna, and Pietro Camarda. 2011. Two-level downlink scheduling for real-time multimedia services in LTE networks. *IEEE Transactions on Multimedia* 13, 5 (2011), 1052–1065.

[15] Yuhui Shi et al. 2001. Particle swarm optimization: developments, applications and resources. In *Proceedings of the 2001 congress on evolutionary computation*, Vol. 1. IEEE, 81–86.

[16] Sivakumar Thangamuthu, Nicola Concer, Pieter JL Cuijpers, and Johan J Lukkien. 2015. Analysis of ethernet-switch traffic shapers for in-vehicle networking applications. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 55–60.

[17] Daniel Thiele, Rolf Ernst, and Jonas Diemer. 2015. Formal worst-case timing analysis of Ethernet TSN's time-aware and peristaltic shapers. In *2015 IEEE Vehicular Networking Conference (VNC)*. IEEE, 251–258.

[18] J-P Thomesse. 2005. Fieldbus technology and industrial automation. In *2005 IEEE conference on emerging technologies and factory automation*, Vol. 1. IEEE, 651–653.

[19] Tsung-Yu Tsai, Yao-Liang Chung, and Zsehong Tsai. 2010. Introduction to packet scheduling algorithms for communication networks. In *Communications and Networking*. IntechOpen.

[20] Luxi Zhao, Paul Pop, and Silviu S Craciunas. 2018. Worst-case latency analysis for IEEE 802.1 Qbv time sensitive networks using network calculus. *IEEE Access* 6 (2018), 41803–41815.