Brief Industry Paper: Response Time Evaluation of Cross-Domain Communication in CAN-FD and TSN

Wenhong Ma[1], Xiaoyi Huang[1], Dongsheng Wei[1], Renfa Li[1], Yunfei Zhang[2], Guoqi Xie[1], Wanli Chang[1]
[1] Key Laboratory for Embedded and Network Computing of Hunan Province, Hunan University, China
[2] Tencent Future Network Lab, Tencent Corporation, China

***Abstract*—With the advancement of intelligence and networked automotive, the domain-centralized architecture, which employs time sensitive networking (TSN) as the inter-domain backbone network and control area network with flexible data rate (CAN-FD) as the intra-domain network, has garnered significant attention. However, cross-domain end-to-end communication involves multiple components, and significant disparities between TSN and CAN-FD render response time analysis within domain-centralized architecture for mixed-critical traffic exceptionally complex. In this paper, we develop a cross-domain with TSN and CAN-FD end-to-end response time evaluation tool, which analyzes the response time of mixed-critical traffic under different design options segment by segment. We specifically analyze the waiting times of different messages in the domain control unit when faced with the design options of one-to-one and multi-to-one conversion of CAN-FD and TSN frames. The proposed evaluation tool can be easily extended to different design options to support more application scenarios. Theoretical computational analysis and real hardware measurements show the effectiveness of our tool.**

I. Introduction

Intelligence and networking promote the evolution of automotive electronic and electrical (E/E) architecture from distributed to domain (zonal) centralized architecture [1]. In automotive distributed architecture, introducing a new function demands an extra ECU along with the corresponding wiring harness tailored for that specific function, which makes the automotive E/E architecture very bloated. Some have exceeded 100 ECUs and have about 150 million lines of code [2]. The deployment and upgrading of intelligent applications are extremely difficult. As shown in Figure 1, a central switch connects multiple domains in automotive domain centralized architecture, while the core of each domain is the domain control unit (DCU) with more complex functions and more powerful performance, responsible for governing and relaying communication messages from multiple ECUs in the domain/zone. This architecture greatly simplifies the automotive electronic system and reduces the number of ECUs and wiring harnesses, which has attracted widespread attention [3].

This work was supported in part by the National Natural Science Foundation of China under Grants 62133014, 61932010, 61972139, 62141212, and 62272155; in part by the Natural Science Foundation of Hunan Province under Grants 2023JJ40172 and 2022JJ10021; in part by the Changsha Municipal Natural Science Foundation under the Grant kq2208039; and in part by the Natural Science Foundation of Chongqing under Grants cstc2021jcyj-msxmX0461.

Corresponding Authors: Wanli Chang (wanli.chang.rts@gmail.com), Guoqi Xie (xgqman@hnu.edu.cn)

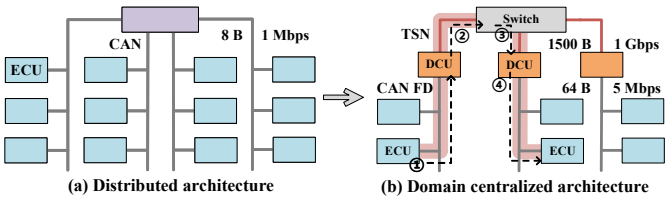

Fig. 1: Automotive E/E Architecture

The in-vehicle network (IVN) is a heterogeneous network interconnected through gateways, utilizing various bus technologies tailored to specific functional requirements and characteristics for cost-efficiency. Figure 2 shows common in-vehicle communication protocols. There are two major trends in the current IVN. On one hand, the traditional bus, exemplified by the Control Area Network (CAN), is progressively advancing toward increased bandwidth and larger payload capacity through a bottom-up approach. After CAN has become the largest in-vehicle bus today, CAN with flexible data rate (CAN-FD) is well compatible with CAN, and has the advantages of higher bandwidth and payload, and has been gradually applied to the IVN [4], [5]. On the other hand, the top-down technology for applying Ethernet to automobiles is also maturing. Automotive Ethernet, represented by Time Sensitive Networking (TSN), is evolving towards determinism and reliability, and is gradually being adopted by manufacturers and applied in backbone networks, audio, and video [6], [7].

The domain-centralized architecture using TSN as the backbone network of inter-domains and CAN-FD as the intra-domain network has received widespread attention [8]. As shown in Figure 1(b), cross-domain end-to-end communication in the domain-centralized architecture is divided into four stages. Different design options in each stage make its response time analysis (RTA) more complicated. Moreover, due to the evident distinctions between TSN and CAN-FD, RTA of mixed-critical traffic faces consequent waiting time in gateway-capable DCUs when converting these two types of frames. For instance, DCU converts CAN-FD to TSN frames and faces one-to-one and multi-to-one design options [9], which causes different waiting time for mixed critical traffic with different real-time requirements. Analyzing the impact of different design options on response time and establishing response time boundaries are essential and challenging.

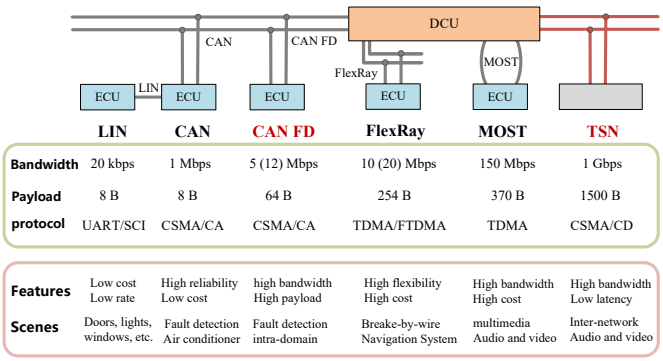

Fig. 2: The common in-vehicle communication protocols

In this paper, we implement a response-time evaluation tool of cross-domain end-to-end communication with TSN and CAN-FD for mixed-critical traffic. The developed tool breaks down the end-to-end response time of domain-centralized architecture into distinct segments, enabling a step-by-step analysis of the response time for different critical messages under specific design choices. In particular, we analyze the waiting time of mixed critical messages in the DCU under different conversion strategies, including one-to-one or multi-to-one conversion from CAN-FD to TSN. Theoretical computational analysis and real hardware measurements show the effectiveness of our tool.

II. Architecture and Models

System architecture. Figure 3 illustrates a typical domain-centralized architecture, where a switch links multiple DCUs via TSN, and each DCU connects to multiple ECUs within the domain using CAN-FD. The DCU utilizes separate buffers to store frames awaiting transmission and frames received from other components. For example, the DCU receives CAN-FD frames from within the domain and stores them in the *CAN-FD_RX_buffer*, while it stores TSN frames from outside the domain in the *TSN_RX_buffer*. In cross-domain communication, the frame conversion module in the DCU converts the TSN frames in the buffer into CAN-FD frames and stores them in *CAN-FD_TX_buffer*, and the reverse frame conversion is stored in *TSN_TX_buffer*.

Message model. In the automotive system, various functional modules may be distributed across different domains, necessitating communication among them to achieve the overall system functionality. Components in the system communicate through IVN in the form of messages. Cross-domain end-to-end communication refers to the process of message transmission from one ECU to another across these different domains. This communication link encompasses all use cases of other component types of communications, such as ECU to DCU, DCU to switch, etc. The message set in the system is modeled as $M = \{m_1, \cdots, m_i, \cdots, m_{|M|}\}$. Each message m_i has attributes $\langle ty_i, src_i, des_i, p_i, s_i, d_i \rangle$, which respectively represent the message type, source ECU, destination ECU, period, size, and deadline. ty_i equals 1 indicates that message m_i is a critical message, otherwise it is a non-critical message. The source

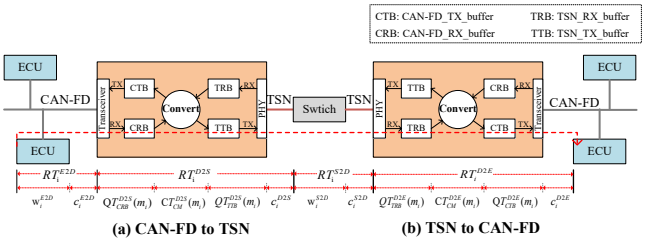

Fig. 3: System architecture and cross-domain end-to-end communication

ECU src_i activates the message m_i with size s_i periodically at p_i intervals and transmits it to the destination ECU des_i.

Design specifications. In resource-constrained systems, there is often a trade-off between performance and overhead to ensure design goals. The response time RT_i is the length of time from src_i activating to des_i receiving m_i. Critical messages have hard real-time constraints, which make it preferable to guarantee higher performance (low response time) rather than low overhead. Non-critical messages have soft real-time constraints, which makes it necessary to explore more design options to balance performance and overhead.

III. RTA of Cross-Domain End-to-End Communication

Cross-domain end-to-end communication is divided into four segments as shown in Figure 3. Correspondingly, the response time RT_i of message m_i is divided into: 1) response time RT_i^{E2D} from source ECU to DCU, 2) time RT_i^{D2S} from DCU to switch, 3) time RT_i^{S2D} from switch to DCU, 4) time RT_i^{D2E} from DCU to destination ECU; and is calculated as follows,

$$RT_i = RT_i^{E2D} + RT_i^{D2S} + RT_i^{S2D} + RT_i^{D2E}. \quad (1)$$

Each of the above four segments can be further divided into processing time, queuing time, transmission time, and propagation time. Considering the strong processing capability of modern devices and the fast data propagation rate of CAN-FD and TSN, and the processing time or propagation time are related to the code logic and physical characteristics of the device, this study primarily focuses on queuing time and transmission time for RTA, but it can be readily extended to include processing time and propagation time.

A. RTA from ECU to DCU

A DCU and multiple ECUs in the domain are mounted on an asynchronous and non-preemptive CAN-FD bus. The components mounted on the bus do not have a unified clock. Even with careful design, messages may collide due to jitter or external influences during long-term operation. Therefore, the response time analysis in intra-domain message m_i needs to consider the waiting time caused by other messages. In this research, we consider using the worst-case response time proposed in [10] to calculate RT_i^{E2D} of m_i,

$$RT_i^{E2D} = c_i^{E2D} + w_i^{E2D}, \quad (2)$$

where transmission time c_i [11] is

$$c_i^{E2D} = 32\mathrm{t_a} + \left(28 + 5\left\lceil \frac{s_i - 16}{64} \right\rceil + 10 s_i\right)\mathrm{t_d} \tag{3}$$

and waiting time w_i is

$$w_i^{E2D}(n+1) = B_i^{E2D} + \sum_{h \in hp(i)} \left\lceil \frac{w_i^{E2D}(n) + \mathrm{t_a}}{p_h} \right\rceil c_h^{E2D}. \tag{4}$$

$\mathrm{t_a}$ and $\mathrm{t_d}$ are the single-bit transmission times of the CAN-FD frame in the arbitration phase and data phase. Refer to Appendix A to calculate the c_i^{E2D}. We prioritize messages that are critical and have shorter deadlines by assigning them higher priority (Please refer to Appendix B). $hp(i)$ is the subscript of the message group with a priority higher than m_i in the source domain. B_i^{E2D} is maximum blocking time of a low priority message that has won arbitration before m_i is activated,

$$B_i^{E2D} = \max_{l \in lp(i)} \left(c_l^{E2D}\right), \tag{5}$$

where $lp(i)$ is the set of message subscripts with priority lower than μ_i. Eq. (4) is a recursive equation, starts at $w_i^{E2D}(0) = c_i^{E2D}$ and ends at $w_i^{E2D}(n+1) = w_i^{E2D}(n)$, and it is easily extended to use busy period in [10] to analyze response time.

B. RTA from DCU to Switch

The response time from the DCU to the switch includes the transmission time and waiting time within the DCU,

$$RT_i^{D2S} = c_i^{D2S} + w_i^{D2S}. \tag{6}$$

The DCU with gateway function converts the CAN-FD frames in the *CAN-FD_RX_buffer* into TSN frames through the frame conversion module and stores them in the *TSN_TX_buffer*. The waiting time w_i^{D2S} in DCU is further split into the queuing time $QT_{CRB}^{D2S}(m_i)$ in *CAN-FD_RX_buffer*, the conversion time $CT_{CM}^{D2S}(m_i)$ of the conversion module, and the queuing time $QT_{TTB}^{D2S}(m_i)$ in *TSN_TX_buffer*,

$$w_i^{D2S} = QT_{CRB}^{D2S}(m_i) + CT_{CM}^{D2S}(m_i) + QT_{TTB}^{D2S}(m_i), \tag{7}$$

where $CT_{CM}^{D2S}(m_i)$ is considered a constant related to the computing performance of the device.

The value of $QT_{CRB}^{D2S}(m_i)$ is associated with the frame conversion strategy (Appendix B). As shown in Figure 4(a), there are two conversion strategies from CAN-FD to TSN: one-to-one and multi-to-one. The rate at which a TSN frame is transmitted out from the DCU is significantly higher than the rate at which a CAN-FD frame is transmitted into the DCU. In order to ensure the transmission performance of critical messages in the design specification, it is natural to choose a one-to-one method to convert a critical CAN-FD frame into a TSN frame. Under this design solution, the $QT_{CRB}^{D2S}(m_i)$ value of the critical frame is equal to 0. For non-critical messages, DCU puts multiple CAN-FD frames into one TSN frame in a multi-to-one manner to reduce the bandwidth overhead at the frame head and tailer and the calculation overhead caused by too many short frames. This manner makes the earlier CAN-FD frame have to wait for the last CAN-FD frame in the same TSN frame,

$$QT_{CRB}^{D2S}(m_i) = \max_{m_j \in \mathcal{M}_i} \left\{ LAT_j^{E2D} - EAT_i^{E2D} \right\}, \tag{8}$$

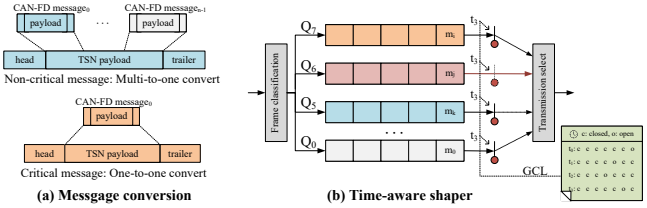

Fig. 4: Message conversion and TAS

where $\mathcal{M}_i$ is the TSN frame to which m_i is mapped, EAT_i^{E2D} is the earliest arrival time of m_i from ECU to DCU, and LAT_j^{E2D} is the latest arrival time of m_j. In this study, we consider EAT_i^{E2D} equal to c_i^{E2D} and LAT_j^{E2D} equal to RT_j^{D2S} to obtain the worst-case queuing time $QT_{CRB}^{D2S}(m_i)$.

The $QT_{TTB}^{D2S}(m_i)$ is correlated with the communication mechanism of TSN, which achieves deterministic transmission through the time synchronization mechanism in IEEE 802.1AS and the time-aware shaper (TAS) in IEEE 802.1Qbv. The clock synchronization mechanism allows the DCU and the switch to have the same clock, which provides the basis for predictable communication timing at both devices. As shown in Figure 4(b), TAS uses different priority queues to isolate different types of messages, where each queue is associated with a gate. The configurable gate control list (GCL) is referenced by the DCU to open and close gates to achieve precise message scheduling. Based on the above mechanism, it is easy to implement no-waiting scheduling [12] in the *TSN_TX_buffer* of DCU by integer linear programming, i.e., *TSN_TX_buffer* forwards the TSN frame immediately after receiving it through the configured GCL without waiting, i.e., $QT_{TTB}^{D2S}(m_i) = 0$.

Since m_i is put into the TSN frame $\mathcal{M}_i$, the transmission time c_i^{D2S} of the message m_i is consistent with $\mathcal{M}_i$,

$$c_i^{D2S} = \left(176 + \max\left(336, \sum_{m_j \in \mathcal{M}_i} \left(60 + 5\left\lceil \frac{s_j - 16}{64} \right\rceil + 10 s_j\right)\right)\right) t_T, \tag{9}$$

where t_T is the transmission rate of TSN. The minimum length of a TSN frame is 64 bytes (including head and trailer), which implies that its payload is extended to 336 bits if it is shorter. Please refer to Appendix A for a description of Eq. (9).

C. RTA from Switch to DCU

The response time from the switch to the DCU is divided into transmission time and waiting time in the switch,

$$RT_i^{S2D} = c_i^{S2D} + w_i^{S2D}, \tag{10}$$

where transmission time c_i^{S2D} is obtained by Eq. (9).

The value of w_i^{S2D} is correlated with the configuration of the TSN switch. This study assumes that each egress of the switch is configured with GCL. Since no-wait scheduling is used from DCU to switch in the previous stage, there is a time interval between messages arriving at the switch, that is, packets can be transmitted to the destination DCU immediately when they arrive at the switch, so the queuing time w_i^{S2D} is 0. Further, the RTA can be easily extended to other configuration such as Cyclic Queuing and Forwarding (Please refer to Appendix B).

D. RTA from DCU to ECU

The response time RT_i^{D2E} from DCU to ECU is calculated as

$$RT_i^{D2E} = c_i^{D2E} + w_i^{D2E}, \tag{11}$$

where transmission time c_i^{D2E} is obtained by Eq. (3), and waiting time is divided into

$$w_i^{D2E} = QT_{TRB}^{D2E}(m_i) + CT_{CM}^{D2E}(m_i) + QT_{CTB}^{D2E}(m_i). \tag{12}$$

The conversion time $CT_{CM}^{D2E}(m_i)$ is a small constant. The time required for transmitting a TSN frame into the DCU, including the frame interval, exceeds the $CT_{CM}^{D2E}(m_i)$. When a TSN frame enters the buffer, the DCU directly converts the TSN frame into one or multi CAN-FD frame, and the waiting time $QT_{TRB}^{D2E}(m_i)$ in the *TSN_RX_buffer* to 0.

The DCU uses a priority queue in *CAN-FD_TX_buffer* to schedule converted CAN-FD frames, with critical frames with shorter deadlines being transmitted first. CAN-FD frames utilize the ID within the frame for arbitration, with lower ID values indicating higher priority. The DCU in the destination domain does not reassign priorities to CAN-FD frames; it retains the ID of the source ECU that activated the CAN-FD frame. The waiting time $QT_{CTB}^{D2E}(m_i)$ can be calculated by

$$QT_{CTB}^{D2E}(m_i) = B_i^{D2E} + \sum_{h \in hp(i)} \left\lceil \frac{QT_{CTB}^{D2E}(m_i) + \mathrm{t_a}}{t_h} \right\rceil c_h^{D2E}, \tag{13}$$

which faces the blocking time B_i^{D2E} of the low-priority message and interference time of high-priority messages from the destination domain. Please note that Eq. (13) is a recursive equation, similar to Eq. (4). However, because they involve different high and low priority sets, the values of these two equations differ.

IV. Experimental Results

We use real-world automotive benchmarks [13] to generate message sets with message sizes of 20, 40, 60, 80, 100, and 120. The generated message set is randomly distributed among the four ECUs shown in Figure 3, and whether the message is critical is also randomly selected. CAN-FD operates with 500 kbps arbitration phases rate and 2 Mbps data phase rate, whereas TSN has a 100 Mbps. The calculation of response time is implemented by Python[1] on a Win10 PC (i5-12400, 32 GB main memory). We further verify the response time on the hardware platform shown in Figure 5, which uses NXP SJA1105Q-EVB as the switch, LS1028A as the DCU, and S32K148-EVB as the ECU.

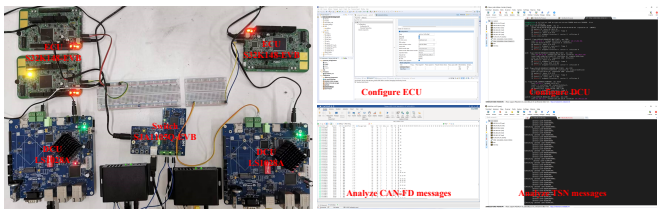

Fig. 5: Hardware platform

[1]The code is open sourced in "https://gitee.com/wenhongma/RTA_CDC".

Figure 6 shows the response time distribution for different message set sizes, where the values of different segments are specifically determined by the analysis method in Section III. The response time of the S2D segment is extremely short due to the no-wait scheduling configuration of TSN. The D2S segment is related to E2D according to Eq. (8). For instance, the point exceeding 30 ms in E2D in Figure 6(f) leads to the upper whisker of the D2S segment also exceeding 30 ms. As shown in Table I, we further compare the measured and analyzed waiting time w_i^{D2S}. The test case shows the waiting time in the D2S segment for 13 cross-domain messages of the 20 messages. All measured values fall within the bounds of analyzed values.

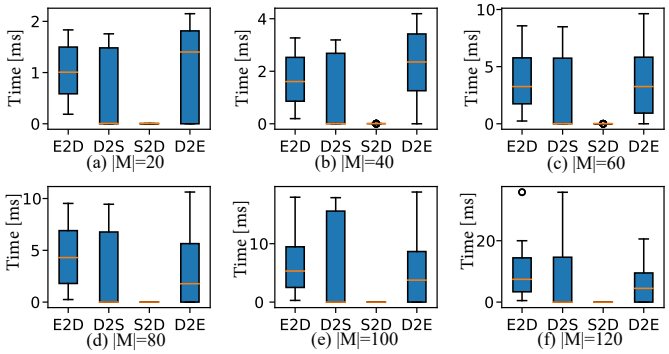

Fig. 6: Response time distribution of different message set sizes

TABLE I: Comparison of measured and analytical of w_i^{D2S}

TSN message	CAN-FD message	ty_i	s_i (bytes)	p_i (ms)	Mea-w_i^{D2S} (μs)	Ana-w_i^{D2S} (μs)
1	1	0	2	10	790.4	1477
	2	0	1	10	393.6	1482
	3	0	2	10	0.48	1477
2	4	0	2	100	863.2	1746
	5	0	4	100	448.3	1736
	6	0	1	100	0.4	1751
3	7	1	2	100	0.36	1
4	8	1	4	20	0.48	1
5	9	1	1	10	0.32	1
6	10	1	1	20	0.32	1
7	11	1	2	20	0.37	1
8	12	1	2	10	0.44	1
9	13	1	1	10	0.41	1

V. Conclusion

In this work, we explore the impact of different design options on response time, especially the waiting time in the buffer. Based on this, we develop a response-time evaluation tool for domain-centralized in-vehicle heterogeneous networks, which performs detailed response time calculations for mixed-critical traffic from ECU to DCU, DCU to switch, switch to DCU, and DCU to ECU. The tool can be easily adapted to different configurations and benefits the design exploration and timing assurance for low-latency and highly reliable communications. In the future, we plan to use the time analysis of frames conversion with this tool to optimize mixed-critical traffic conversion strategies and scheduling approaches.

REFERENCES

[1] V. Bandur, G. Selim, V. Pantelic, and M. Lawford, "Making the case for centralized automotive E/E architectures," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1230–1245, 2021.

[2] O. Burkacky, J. Deichmann, G. Doll, and C. Knochenhauer, "Rethinking car software and electronics architecture," *McKinsey & Company*, p. 11, 2018.

[3] L. Deng, G. Xie, H. Liu, Y. Han, R. Li, and K. Li, "A survey of real-time ethernet modeling and design methodologies: From AVB to TSN," *ACM Computing Surveys (CSUR)*, vol. 55, no. 2, pp. 1–36, 2022.

[4] W. Zeng, M. A. Khalid, and S. Chowdhury, "In-vehicle networks outlook: Achievements and challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1552–1571, 2016.

[5] W. Ma, G. Xie, R. Li, and W. Chang, "Optimality-guaranteed design space pruning for CAN-FD frame packing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[6] J. Wang, J. Liu, and N. Kato, "Networking and communications in autonomous driving: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1243–1274, 2018.

[7] W. Ma, X. Xiao, G. Xie, N. Guan, Y. Jiang, and W. Chang, "Fault tolerance in time-sensitive networking with mixed-critical traffic," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[8] V. Bandur, R. Kapinski, V. Pantelic, M. Lawford, and B. Wasacz, "Aspects of migrating from decentralized to centralized E/E architectures," SAE Technical Paper, Tech. Rep., 2022.

[9] A. Kern, D. Reinhard, T. Streichert, and J. Teich, "Gateway strategies for embedding of automotive CAN-frames into ethernet-packets and vice versa," in *Proceedings of the 24th international conference on Architecture of computing systems*, 2011, pp. 259–270.

[10] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller area network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, pp. 239–272, 2007.

[11] U. D. Bordoloi and S. Samii, "The frame packing problem for CAN-FD," in *2014 IEEE Real-Time Systems Symposium*. IEEE, 2014, pp. 284–293.

[12] F. Dürr and N. G. Nayak, "No-wait packet scheduling for IEEE time-sensitive networks (TSN)," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, 2016, pp. 203–212.

[13] S. Kramer, D. Ziegenbein, and A. Hamann, "Real world automotive benchmarks for free," in *Proceedings of the 6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*, vol. 130, 2015.

[14] N. C. Audsley, "On priority assignment in fixed priority scheduling," *Information Processing Letters*, vol. 79, no. 1, pp. 39–44, 2001.

APPENDIX A
ANALYSIS TRANSMISSION TIME

Transmission Time Analysis in Eq. (3)**.** Figure 7 shows a CAN-FD frame, which contains two arbitration phases and a data phase, and their single-bit transmission times are t_a and t_d respectively. The calculation of the CAN-FD worst-case transmission time depends on the bit numbers of the different phases. The available payload in the data phase of the frame, denoted s_i, include 0-8, 12, 16, 20, 24, 32, 48, and 64 bytes. CAN-FD frames use different bit stuffing rules in different fields: 1) The CRC field uses mandatory bit stuffing, which makes the CRC field of a CAN-FD frame with a payload of fewer than 16 bytes have 21 (including 17 bit CRC and 4 bit stuff) after bit stuffing, while larger than 16 bytes are 26; 2) The part of the CAN-FD frame from the beginning to the end of the payload section follows the bit stuffing mechanism, i.e. if the same five bits are transmitted in sequence, the communication controller transmits a stuff bit opposite to the value of the consecutive bits. Therefore, under the worst-case bit stuffing scenario, arbitration phases have 32 bits, while the data phase has $28 + 5\lceil \frac{s_i-16}{64} \rceil + 10s_i$.

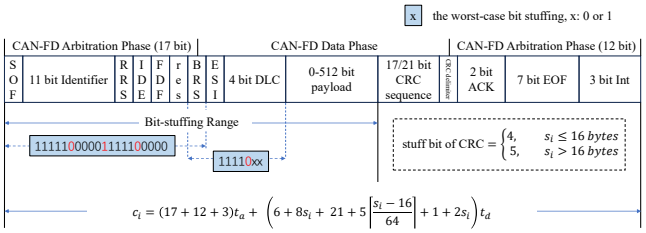

Fig. 7: CAN-FD message transmission time with bit-stuffing mechanism

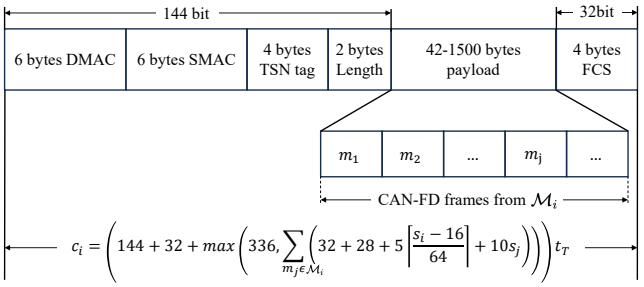

Fig. 8: TSN message transmission time under the multi-to-one strategy

Transmission Time Analysis in Eq. (9)**.** The structure of a TSN frame is shown in Figure 8. The Ethernet standard specifies that the minimum length of a frame is 64 bytes, including header, payload, and trailer. Since TSN is an extension to the standard Ethernet protocol that inserts a 4-byte VLAN tag into a standard Ethernet frame, the minimum payload of a TSN frame is 42 bytes (336 bits). In the multi-to-one strategy, the entire CAN-FD frame is treated as part of the payload of a TSN frame, so the number of bits transmitted in a TSN frame is $\sum_{m_j \in \mathcal{M}_i}(32 + 28 + 5\left\lceil \frac{s_j-16}{64} \right\rceil + 10s_j)$. If the payload of a TSN frame is less than 336 bits after frame aggregation, it is extended to 336 bits.

APPENDIX B
DISCUSS AVAILABLE DESIGN OPTIONS

This section discusses the design options available for the four segments of cross-domain communication in Section III.

1) Design options from ECU to DCU. During this segment, response time can be influenced by design options such as RTA and priority assignment methods. Due to space limitations, we consider Eq. (4) to calculate the waiting time. In fact, the complete response time analysis approach using busy periods in [10] can be used. Nonetheless, this approach results in overly conservative response times in Eqs. (4) and (8) for cross-domain communication. Employing a more precise RTA method at this segment allows for more accurate upper bounds. In addition, the priority assignment also has an impact on the RTA. We use the priority assignment algorithm that combines criticality with Earliest Deadline First (EDF) in this segment. It can be considered to be extended to the optimal priority assignment algorithm in [14].

2) Design options from DCU to switch. The DCU converts CAN-FD frames into TSN frames in this segment. Mixed critical messages in this segment are used in different strategies to ensure their requirements. For critical messages, a one-to-one conversion strategy is executed, and the DCU assigns a high-priority queue and a forwarding strategy as fast as possible to ensure its real-time requirements. For non-critical messages, a multi-to-one conversion strategy is used to balance bandwidth overhead and quality of service. In this segment, the design options of frame conversion strategies, priority assignment, and offset assignment need to be considered.

3) Design options from switch to DCU. When analyzing the response time from the switch to the DCU, the switch adopted TAS and configured GCL for no-wait scheduling. The switch can be easily extended to accommodate other design options, like CQF, which employs a cyclic timer and two transmission queues. It divides the global time into equal time slots, and the two queues alternate between sending and receiving messages. The switch sends the message in the next time slot at the latest after receiving it. Therefore, the maximum waiting time of m_i in switch configured with CQF is $w_i^{S2D} = 2 \times SL$, where SL is the time slot length.

4) Design options from DCU to ECU. At this segment, the DCU immediately unpacks the TSN frame into a CAN-FD frame after receiving the message. As the ID value of CAN-FD in the first segment is retained during this segment, the priority of critical messages remains higher than that of non-critical messages, allowing them to be activated first. Furthermore, the transmission rate of TSN significantly surpasses that of CAN-FD. Frequent arrivals of non-critical messages may lead to an accumulation of CAN-FD messages in the buffer, potentially resulting in network congestion. The congestion-aware unpacking mechanism and the priority assignment algorithm for CAN-FD frames can be synergized to achieve better traffic control.