frontiers | Frontiers in The Internet of Things

TYPE Hypothesis and Theory
PUBLISHED 18 September 2024
DOI 10.3389/friot.2024.1433903

OPEN ACCESS

EDITED BY
Petros Spachos,
University of Guelph, Canada

REVIEWED BY
Luis Lino Ferreira,
Instituto Superior de Engenharia do Porto (ISEP), Portugal
Marc Jayson Baucas,
University of Guelph, Canada

*CORRESPONDENCE
Stefanos Tziampazis,
stefanos.tziampazis@gsame.uni-stuttgart.de

RECEIVED 16 May 2024
ACCEPTED 03 September 2024
PUBLISHED 18 September 2024

CITATION
Tziampazis S, Hirmer P and Weyrich M (2024) A hybrid framework for latency compensation in remote testing of automotive electronic control units.
Front. Internet. Things 3:1433903.
doi: 10.3389/friot.2024.1433903

A hybrid framework for latency compensation in remote testing of automotive electronic control units

Stefanos Tziampazis[1,2]*, Pascal Hirmer[2] and Michael Weyrich[3]

[1]Graduate School of Excellence advanced Manufacturing Engineering (GSaME), University of Stuttgart, Stuttgart, Germany, [2]Mercedes-Benz AG, Stuttgart, Germany, [3]Institute of Industrial Automation and Software Engineering, University of Stuttgart, Stuttgart, Germany

As the number of electronic control units (ECUs) in vehicles continues to grow, exchanging physical components between original equipment manufacturers (OEMs) and ECU suppliers presents logistical challenges, impeding the pace of development and leading to accumulation of costs. In this work, we introduce a conceptual framework that enables remote testing of geographically dispersed ECUs over the Internet. Pertinent to the Internet of Things (IoT) sphere, where interconnection of distributed components is hampered by the inherent challenges of latency, we propose a hybrid synchronous methodology that combines asynchronous test management with time synchronization mechanisms to mitigate the delay impact within the distributed environment. Additionally, we discuss challenges and prospects associated with this approach.

KEYWORDS

electronic control unit, automotive, distributed, testing, synchronization, networking, latency

1 Introduction

As technological advancements continually reshape the industry's landscape European Parliament et al. (2021), automotive development, characterized by a globalized nature, necessitates alliances between original equipment manufacturers (OEMs) and external suppliers. Collaborations and outsourcing agreements are vital to keeping abreast of evolving trends, sharing the burden of research, and matching turnaround times Ciravegna and Pilkington (2013). An example of a field that intrinsically relies on collaborative efforts is ECU development, with electronic control units (ECUs) being embedded devices regulating multiple vehicle functions, including engine performance, transmission control, safety systems, etc. Here, OEMs commonly rely on external suppliers to undertake their manufacturing, engineering, and optimization in alignment with the desired specifications. Throughout the development process, ECUs are subjected to testing, either in standalone or group configurations. The latter involves assessing their interdependent operation, a testing phase known as *integration testing* Sobotka and Novák (2013).

To date, integration testing has been conducted at the premises of the automotive OEM, necessitating that all external ECU suppliers ship their devices to a centralized location. From the OEM's perspective, this poses a challenge due to the combination of multiple suppliers and their dispersed geographical regions. This logistical complexity results in increased costs and time pressure, compounding the challenges of meeting the next vehicle release schedule.

In our previous work Tziampazis et al. (2023), we detailed a refined ECU development V-model while discussing how integration testing of physical ECUs between automotive OEMs and their suppliers over the Internet could enhance agility in outsourced development. From a synergistic viewpoint, utilizing distributed physical resources as opposed to a single, centralized testing facility—which would otherwise necessitate co-locating all components—proves advantageous in managing the development schedule and strategically planning costs. In the present article, we build on our previous theoretical research by emphasizing practical considerations, specifically targeting the challenge of long-distance latency, caused by geographic spans between interconnected resources, as well as issues of timeliness. The limitation of latency impacts the feasibility of remote testing across a wide range of applications, especially in dynamic scenarios with temporal constraints or scheduling requirements.

In view of this restriction, we present a hybrid *framework* for remote ECU testing that combines asynchronous test management, where events are processed independently by the testing authority subsequent to their occurrence, with synchronous timestamping techniques for time referencing. Latency introduced in the distributed links is calculated and corrected post-event through the exchange of timestamps across the distributed nodes. This expands the spectrum of test cases that can be evaluated remotely, in that it allows for the assessment of temporal characteristics (e.g., signal response times) in addition to purely functional attributes (e.g., signal occurrences). In the context of this paper, we define *framework* as a structured approach that encompasses the architecture of a distributed setup of components (e.g., ECUs, gateways, interconnection links), the communication protocols or entities (e.g., messages, data exchange formats, synchronization methods), and the methodological components (e.g., algorithms, processes) that facilitate distributed communication and testing.

We envisage that this work provides new insights into integrating the realms of the Internet of Things (IoT) and automotive ECU testing. By conceptualizing automotive ECUs as IoT components and framing remote ECU testing as a distributed IoT process, key challenges such as latency, timeliness, and synchronization emerge as common issues in both domains. Our proposed method to ameliorate the impact of latency through a hybrid approach of synchronicity and asynchronicity could hold potential for application in a range of distributed IoT testing processes that are not bound by strict real-time requirements.

The remainder of the article is structured as follows: Section 2 explores typical synchronization principles and mechanisms, Section 3 covers related work, Section 4 presents the proposed hybrid framework, alongside an exemplary test scenario, represented by the corresponding mathematical model, Section 5 discusses further considerations and practical constraints, and finally, Section 6 concludes and outlines future directions.

2 Time synchronization background

This section examines the principles of time synchronization and its implementation across three protocols: Network Time Protocol (NTP), Precision Time Protocol (PTP), and Controller Area Network (CAN). These protocols are deployed in multiple applications with varying precision requirements, each offering distinct approaches to time synchronization.

Frequently, *synchronization* refers to imparting a reference from one entity to another with the former typically referred to as the master and the latter as the slave. It is classified into three distinct types Lévesque and Tipper (2016): 1) *time synchronization*, which involves aligning the clocks of different devices to a common time reference, 2) *frequency synchronization*, which focuses on aligning the oscillations of devices to maintain the same frequency, and 3) *phase synchronization*, which relates to aligning the phase angles of signals or waveforms. In certain protocols or processes, it is typical for the last two, i.e., *frequency* and *phase synchronization*, to be incorporated within the *time synchronization* mechanism. The sense of time reference to be shared can take two forms: *absolute* or *relative*. In the first case, clocks are aligned to a global time reference, such as UTC (Coordinated Universal Time) or GPS (Global Positioning System). In the second case, the alignment is based on specific events that may not be directly related to the true perception time, e.g., the time elapsed since the power-up of an ECU.

Achieving precise time synchronization entails calculating and compensating for the time offset between the reference master and the non-synchronized slave. This can be applied iteratively to multiple nodes that seek synchronization with the same reference. Compensation is essential, as transmitting the reference time from the master to the slave inherently consumes time, which, if left unaccounted for, could lead to the slave drifting out of synchronization by the time it receives it. Fundamentally, for synchronization with the master, the slave adjusts its timing as depicted in Equation 1:

$$\tau_{slave} \leftarrow \tau_{slave} - \vartheta + \gamma \tag{1}$$

where ϑ represents the time deviation between the master and slave clocks and γ the propagation delay in transmitting the reference time instance from the master to the slave. The left arrow operator $\leftarrow$, commonly used in computer science to denote an assignment, indicates that the right-side expression is evaluated and assigned to the left-side variable. Here, τ_{slave} on the right side represents the time base of the slave before synchronization, while τ_{slave} on the left side denotes the updated time after successful synchronization; Hence, the value does not change directly; rather, it is computed using the expression and then reassigned, effectively updating it.

By assigning $\tau_{offset} = \vartheta - \gamma$, Equation 2 can be expressed as follows:

$$\tau_{slave} \leftarrow \tau_{slave} - \tau_{offset} \tag{2}$$

Figure 1 demonstrates the process of time synchronization in NTP, PTP, and CAN, along with typical and worst-case synchronization errors based on the distance between the engaged nodes. Local values apply to nodes situated close together, such as those on a local Ethernet subnet, whereas global values apply to nodes communicating over longer distances. While these values provide a general idea of the differences between the protocols, the actual error depends significantly on various factors, including system implementation, network congestion, clock precision, etc. Next, we provide a concise overview of these protocols.

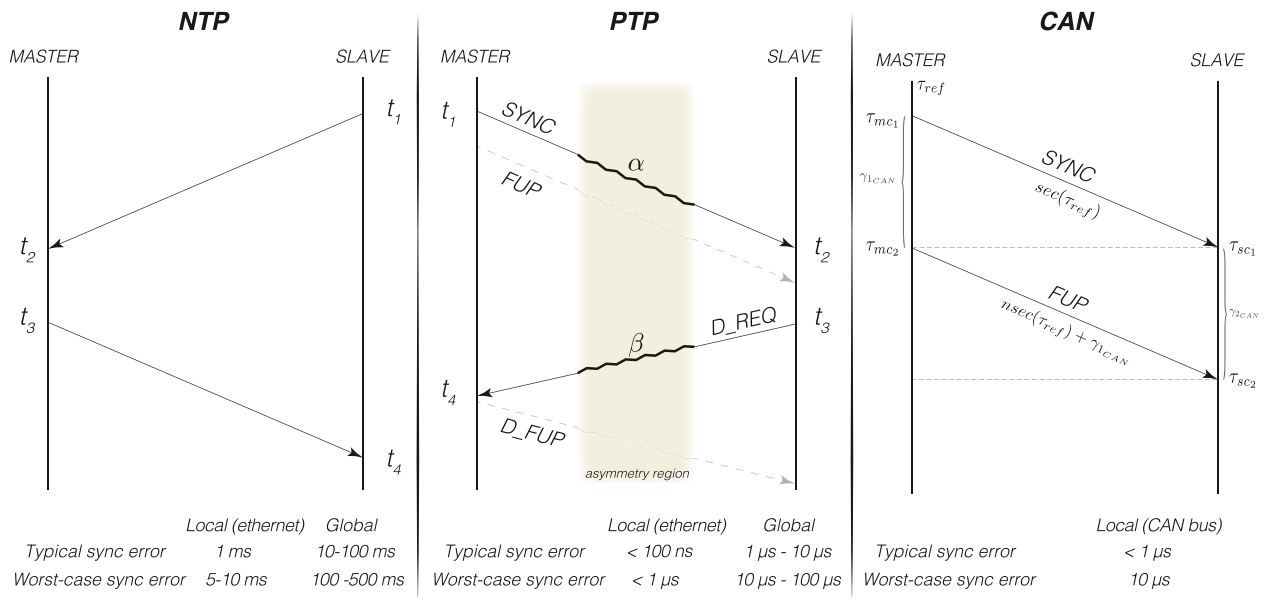

FIGURE 1
Time synchronization mechanisms and estimated synchronization errors in NTP, PTP, and CAN.

2.1 Network Time Protocol (NTP)

NTP Mills (1991) is one of the oldest protocols for time synchronization and operates on a hierarchical, stratum-based structure, where certain nodes act as time-servers at different levels or strata, based on their precision. The stratum level, ranging from 0 to 15, represents the proximity of a device to the reference clock. The protocol relies on a bidirectional handshake initiated by a slave to a master node. Through timestamps t_1 to t_4, two captured from each node, the slave adjusts its clock in line with the following Equations 3–5:

$$\vartheta_{NTP} = t_2 - t_1 \tag{3}$$

$$\gamma_{NTP} = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \tag{4}$$

$$\tau_{slave_{NTP}} \leftarrow \tau_{slave_{NTP}} - \underbrace{\frac{(t_2 - t_1) - (t_4 - t_3)}{2}}_{\tau_{offset_{NTP}}} \tag{5}$$

The delay γ_{NTP} is computed as the average of the two bilateral timestamp differences $(t_2 - t_1), (t_4 - t_3)$, assuming symmetrical stream links. This assumption is critical and often invalid in real-life scenarios due to potential queuing delays and various forms of asymmetry. Taking these factors into account, NTP is suitable in cases where precision on the millisecond scale is desirable.

2.2 Precision Time Protocol (PTP)

Introduced in 2002 under the IEEE 1588 standard IEEE (2002), PTP relies on a pairwise exchange for time synchronization. Offering significant advancements over NTP, PTP achieves a minimum accuracy of 1 microsecond for up to seven hops, making it valuable in various applications, such as Ethernet Audio Video Bridging (AVB) IEEE (2023), where audio and video synchronization are crucial. Its latest revision is the generalized Precision Time Protocol (gPTP), also known as IEEE (2020), which is one of the standards within the Time-Sensitive Networking (TSN) framework Fedullo et al. (2022). Notable improvements include the introduction of the peer delay mechanism, a transition to a layer two architecture from the original multicast IPv4 setup, and the implementation of Boundary Clocks (BC) and Transparent Clocks (TC). The latter enable adjusting time packets as they traverse the network, a useful feature in the context of automotive gateways. Additionally, the introduction of asymmetry correction within the standard frame structure allows compensation for various asymmetries along the synchronization links.

PTP utilizes a time synchronization process where the master initiates synchronization by sending *SYNC* messages to the slave at regular intervals. After sending *SYNC*, the master captures the timestamp and sends it back the nodes using a follow-up *FUP* message. To measure the delay of the link, they then exchange delay request *D_REQ* and delay response *D_FUP*. To account for the asymmetries introduced along the network links, the equations are adjusted as shown in Equations 6–8:

$$\vartheta_{PTP} = t_2 - t_1 \tag{6}$$

$$\gamma_{PTP} = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} - \frac{\alpha + \beta}{2} \tag{7}$$

$$\tau_{slave_{PTP}} \leftarrow \tau_{slave_{PTP}} - \underbrace{\frac{(t_2 - t_1) - (t_4 - t_3)}{2} - \frac{\alpha - \beta}{2}}_{\tau_{offset_{PTP}}} \tag{8}$$

where α and β, as shown in Figure 1, denote the cumulative asymmetrical delays in the upstream and downstream links, respectively. These delays commonly arise during packet routing, queuing, or protocol translation, e.g., in the case of an Ethernet to CAN gateway. The latter, which plays a central role in the proposed framework, will be further explored in the upcoming sections.

2.3 Controller area network (CAN)

The requirement for time synchronization among multiple nodes remains significant for automotive networking, particularly when safety takes precedence. Being no exception, CAN enables ECUs to synchronize with each other, ensuring accurate coordination and data exchange.

The time synchronization in CAN networks is described by the AUTOSAR (AUTomotive Open System ARchitecture) standard AUTOSAR (2021) and involves transferring the master's time reference to the slave and compensating for the propagation delay during this transmission. The key distinction here is that the synchronization handshake is unidirectional rather than bidirectional as in NTP and PTP. Upon receiving the master's reference, the slave node updates its internal clock based on Equation 9:

$$\tau_{slave_{CAN}} \leftarrow \tau_{ref} + \gamma_{CAN} \tag{9}$$

where τ_{ref} denotes the master's reference and γ_{CAN} the total delay until it is received by the slave.

A time reference (τ_{ref}) typically consists of 48 bits for seconds and 32 bits for nanoseconds. As a result, two CAN frames are required for transmission. In the synchronization process, the first frame *SYNC* carries the seconds' portion of the time reference $sec(\tau_{ref})$, while the second frame *FUP* carries the nanoseconds' portion $nsec(\tau_{ref})$.

The delay γ_{CAN} is calculated as the sum of the *SYNC* and *FUP* frame delays, denoted $\gamma_{1_{CAN}}$ and $\gamma_{2_{CAN}}$, respectively. To calculate $\gamma_{1_{CAN}}$ and $\gamma_{2_{CAN}}$ two 32-bit registers are utilized on both the master and slave sides. The master counter *mc* captures two stamps: one at the start of the *SYNC* frame τ_{mc_1} and one at the end τ_{mc_2} upon acknowledgment of receipt. Similarly, the slave counter *sc* records two stamps τ_{sc_1} and τ_{sc_2} upon receiving the *SYNC* and *FUP* messages, respectively. Finally, the slave adjusts its timing based on Equation 10:

$$\tau_{slave_{CAN}} \leftarrow \underbrace{sec(\tau_{ref}) + nsec(\tau_{ref})}_{\tau_{ref}} + \underbrace{\gamma_{1_{CAN}} + \gamma_{2_{CAN}}}_{\gamma_{CAN}} \tag{10}$$

where $\gamma_{1_{CAN}} = nsec_{\tau_{mc_2} - \tau_{mc_1}}$ and. $\gamma_{2_{CAN}} = nsec_{\tau_{sc_2} - \tau_{sc_1}}$

3 Related work

3.1 ECU development and automotive testing

At the outset of ECU development courses, hardware may not be readily accessible. Consequently, engineers frequently resort to virtual models and co-simulations to conduct initial testing and analysis. Expounding upon the inefficiency of real-world vehicle testing, Schmidt et al. (2015) propose a hardware-independent platform where virtual ECUs and their software can be seamlessly integrated, thereby advancing software maturity to higher levels. Echoing this perspective, Morishima et al. (2018); Phatak et al. (2016) propose the utilization of multi-virtual-ECU environments for validation and experimentation purposes. Despite the benefits of early virtual testing, physical testing remains critical later on. In bridging physical and virtual testing, Dengler et al. (2021) present a framework for testing virtual and physical ECUs together remotely. However, a limitation is that physical ECUs are still co-located and maintained by a central authority, leading to logistical challenges in case of malfunctions. With a focus on final real-vehicle testing, Johanson and Karlsson (2006) describe a framework where real-time testing data from is transmitted to a remote engineering lab. Despite networking limitations, the authors stress the importance of remote testing to meet future demands.

3.2 Intra-vehicular distribution and end-to-end analysis

Distribution of numerous functionalities and applications among various components within the local domain of a vehicle is an already established practice. For instance, the Anti-lock Braking System (ABS) requires the synergistic collaboration of wheel sensors, brake actuators, control units, and additional supporting components. Studying the temporal behavior of series of causal events, commonly referred to as *distributed transactions* or *data chains*, is a crucial part of *end-to-end analysis*, with a focus on understanding the interactions and timing traits between interconnected systems. This analysis is also pertinent in the context of AUTOSAR, an initiative to create an open and standardized software architecture for automotive systems, where software components are locally distributed across various nodes and communicate via a virtual function bus. Expanding on the end-to-end semantics, Feiertag et al. (2008) distinguish between two key semantics applicable to distributed register-based ECU communication: data age and button-to-action delay. These semantics are critical in different applications, ensuring timely and accurate responses within the system. Building on this foundation, Dürr et al. (2019) extend the analysis by considering mixed periodic and sporadic signals in a distributed architecture, operating under the pattern of locally time-triggered but globally asynchronous communication. Rajeev et al. (2012) introduce two additional semantics of delay, namely, the actuation and correlation constraints, and model the behavior of a locally distributed architecture using finite-state automata. Through the same lens of distributed real-time embedded systems, Mubeen et al. (2014) present a concept for communications-oriented development and conduct an end-to-end analysis of an adaptive cruise control system spanning locally distributed components. While the distribution of functions and components is a fundamental aspect of existing vehicular architectures and is well-supported by academic work, transitioning from a local to a global distribution domain—where ECUs are geographically dispersed rather than in close proximity—introduces new challenges. This transition, which involves overcoming issues such as latency, synchronization, and timeliness over long distances, remains largely unexplored.

3.3 Hardware as a (remote) service

The practicality of remotely accessing hardware components finds further applications. Domski (2022) introduces a remote laboratory, enabling students to access resources remotely through Raspberry Pi-based server architectures. Similarly,

Scherer (2022) presents a virtual laboratory environment for model-based development and Hardware-in-the-Loop testing, emphasizing that tasks executed online require only 10% additional time. While promising, the performance of both methods is constrained by real-time networking limitations, deteriorating as resources become more geographically dispersed.

3.4 Latency and timing

Considering the complexity of automotive distributed ECU architectures, Davare et al. (2007) describe a method to optimize periodic tasks and interactions. This involves asynchronous communication between non-blocking input and output buffers, along with hard bounds on latency and end-to-end delay, albeit, the latter being challenging in long-distance settings. In a parallel effort to minimize end-to-end delay in data transfer within smart cities, Chin et al. (2017) leverages software-defined networking and timestamp recording via NTP synchronization to monitor message flows over time. Timestamping proves to outperform traditional ping and packet probing methods. In general, minimizing latency is a complicated endeavor that, according to Briscoe et al. (2016), builds upon the following steps: identifying latency sources, devising techniques to minimize it, and finding the balance between deployment difficulty and benefit.

An effort to amalgamate these various aspects into a unified approach for remotely testing distributed automotive ECUs while addressing temporal constraints has, to the best of our knowledge, not yet been undertaken.

4 Hybrid framework for remote ecu testing

Expounding on our introductory definition of *framework* as a structured approach that encompasses the architecture of a distributed setup of components (e.g., ECUs, gateways, interconnection links), the communication protocols or entities (e.g., messages, data exchange formats, synchronization methods), and the methodological components (e.g., algorithms, processes), in this section we explore all three aspects in detail, with emphasis on the critical impediment of latency in the remote testing context. To facilitate the analysis, we introduce a mathematical model that describes the framework's nodes, encompassing the gateways, the ECUs, and the central OEM management, along with all the message interactions involved. Rajeev et al. (2012) employed a similar model to describe a distributed system, representing messages and ECUs as objects and resources respectively, characterized by tuples with seven elements: resource allocation, priority, initial offset, period, execution time, and input/output buffers. In our model, we abstract from the priority characteristics and communication patterns, focusing instead on the system's synchronicity and temporal behavior. To demonstrate the approach's applicability, we apply the mathematical model in a remote test scenario assessing the door-locking functionality.

The architecture of the framework's distributed components is depicted in Figure 2 and organized into three hierarchical levels. We use the term *level* to distinguish from the networking term *layer*, which is typically associated with the Open Systems Interconnection (OSI) model Kumar et al. (2014) and respective network protocol layers. At level 1, physical ECUs are housed locally by suppliers across various geographical locations. For this study, and to align with the previously discussed synchronization methods, our focus is exclusively on ECUs operating on CAN, which remains integral to the communication backbone of modern vehicles and underpins numerous critical features.

Level 2 consists of external gateways situated close to each ECU, one provided to each supplier accordingly. Fundamentally, gateways are devices that enable communication between heterogeneous networks Kim et al. (2015). In the proposed framework, they enable the interconnection between the low-level CAN bus and the Internet. Similar to the transparent clocks in TSN Waldhauser et al. (2020), the gateways modify and encapsulate CAN frames into higher-layer packets or segments, and *vice versa*. In addition to this transparency feature, we require that the gateways operate as independent CAN nodes and interact with their adjacent ECUs, establishing a dedicated local communication channel. This enables the independent reproduction of specific events without requiring coordination across all three levels simultaneously.

Level 3 involves test management, coordination, and result evaluation by the automotive OEM. The selection of involved ECUs depends on the specific functionality being tested and the hardware availability. To ensure data security and address intellectual property concerns during the exchange, communication between the level 3 central node and level 2 gateway nodes is established through VPN traffic channels.

Synchronization within the framework is achieved through PTP for communication between level 2 and level 3, and CAN mechanisms between level 1 and level 2, as discussed previously. A factor behind choosing PTP over NTP is its localized synchronization capability, as it does not depend on external systems like GPS, which are often inaccessible in most closed laboratory setups. Moreover, PTP is utilized in multiple automotive applications, and in conjunction with the TSN framework, it is projected to play a pivotal role in the advancement of Ethernet vehicular architectures Bandur et al. (2021); Walrand et al. (2021).

In addition to functional testing, not bound by stringent time constraints, the combination of asynchronous execution, alongside the synchronous timestamping strategy, enables the evaluation of more dynamic scenarios, in which the introduced latency in the links between levels 2 and 3 would otherwise lead to test case failures.

4.1 Framework's mathematical model

To explicate the methodological and algorithmic components of the framework as defined earlier, we employ the following mathematical model: the ordered couple $\langle i, j \rangle$: $i, j \in \{c, gw_1, \ldots, gw_n, e_1, \ldots, e_n\}$ denotes a bilateral interaction in the form of a message, which is initiated by node i to node j, where c is the central OEM node in level 3, $gw_1, \ldots, gw_n$ are the gateway nodes in level 2, and $e_1, \ldots, e_n$ are the ECU nodes in level 1. For instance, $\langle c, gw_1 \rangle$ represents a message sent from the central test-management node (*OEM*) to the first gateway node (*Gateway 1*), while $\langle gw_2, e_2 \rangle$ represents a message sent from the second gateway node (*Gateway 2*) to the second ECU node (*ECU 2*), as depicted in Figure 2.

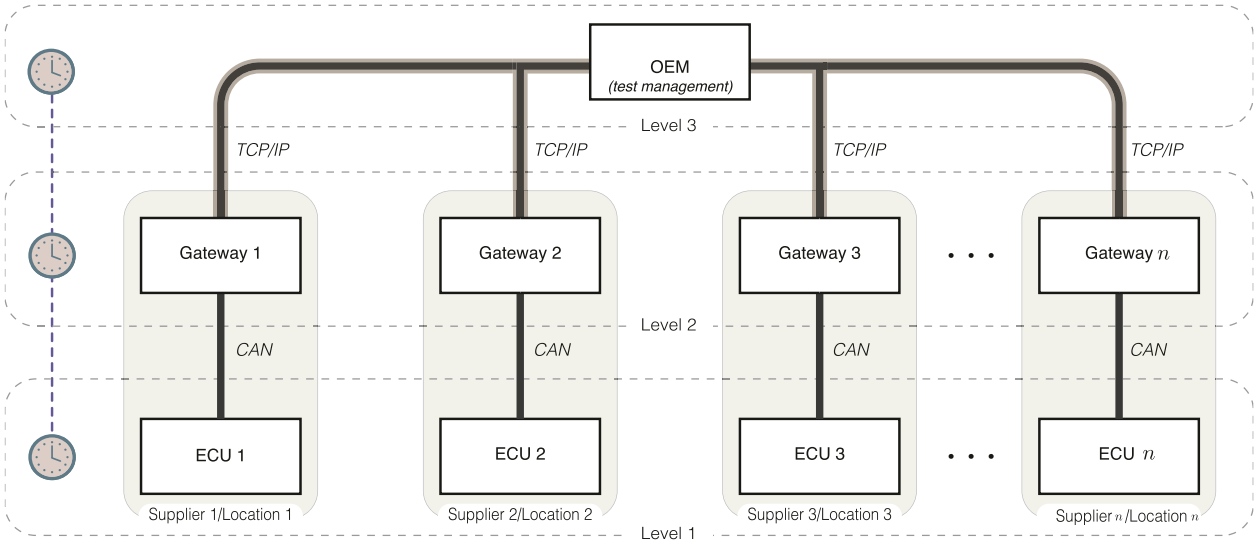

FIGURE 2
Hybrid framework for remote ECU testing: Synchronous timestamping and asynchronous processing enable latency compensation in the links between levels 2 and 3.

In keeping with networking nomenclature, $s\langle i,j\rangle \subset \langle i,j\rangle$: $i,j \in \{c, gw_1, \ldots, gw_n\}$ denotes a segment—a unit of data used at the Transport layer (Layer 4) of the OSI networking model—whereas $f\langle i,j\rangle \subset \langle i,j\rangle$: $i,j \in \{gw_1, \ldots, gw_n, e_1, \ldots, e_n\}$ denotes a CAN frame. For the sake of clarity, we use the term *segment* to refer to the encapsulated data exchanged between level 1 and level 2 nodes, and *frame* to refer to the CAN data exchanged between level 2 and level 3 nodes. The distinction lies in the involved OSI levels for each exchange.

Regarding the subset symbol, the notation $s\langle i,j\rangle \subset \langle i,j\rangle$ is not used in the conventional mathematical sense. Instead, it indicates that $s\langle i,j\rangle$ is a specific part of the overall communication event represented by $\langle i,j\rangle$. Therefore, $s\langle i,j\rangle$ (or $f\langle i,j\rangle$) should be understood as a distinct component within the broader interaction between nodes i and j. In the interest of simplicity, $sync\langle i,j\rangle \subset \langle i,j\rangle$: $i,j \in \{c, gw_1, \ldots, gw_n, e_1, \ldots, e_n\}$ represents the sum of all pairwise interactions required to establish synchronization between two nodes at adjacent levels. Synchronization between level 2 and level 3 nodes relies on PTP, while synchronization between level 1 and level 2 nodes relies on the CAN mechanism, as described in the previous section.

After successful synchronization between two nodes, they both share the same and consistent notion of time. Consequently, timestamping before a message is sent and after it is received enables the calculation of the respective duration, depicted as $|\langle i,j\rangle| = t\langle j\rangle - t\langle i\rangle$: $i,j \in \{c, gw_1, \ldots, gw_n, e_1, ..., e_n\}$, where $t\langle i\rangle$, $t\langle j\rangle$ are the timestamps taken by the sender and receiver node successively. For example, $|\langle c, gw_1\rangle|$ represents the time taken for a message to travel from the central node (*OEM*) to the first gateway node (*Gateway 1*). The link delay is calculated by subtracting the timestamp taken by the central node from the timestamp taken by the gateway node upon receiving the message.

To promote simplicity, we make the following *assumptions* for the model and the specific upcoming test scenario: 1) The distributed system comprising the ECU nodes, the gateways, and the testing authority (*OEM*) behaves in a manner that respects *serializability*; concurrent messages and events occur as if they were sequential, resulting in the same final effect. This holds true in scenarios where the most recent command determines the system state, making intermediate commands inconsequential. For example, in climate control systems, the final temperature setting command overrides any prior adjustments. 2) Time synchronization across all three levels occurs only once at the beginning of the test case, assuming that the total execution time of the test case is small enough for the different nodes to maintain synchrony without the clock skew increasing beyond the average event resolution. 3) The fidelity of physical clocks is sufficient to ensure proper event ordering without the need for logical clocks, e.g., Vector clocks Kleppmann (2017). This can be achieved with robust, low-drift hardware clocks, which maintain minimal clock skew over short durations. 4) Messages are received if sent (reliable distribution links), and nodes exhibit crash-stop behavior; in the event of a crash, they cease execution, resulting in test case failure. This is made possible by using protocols for guaranteed message delivery (e.g., TCP) and implementing fail-safe mechanisms (e.g., redundancy). Some of these aspects will be further discussed in the next section. More complex scenarios, where logical timing, causality, conflict resolution, and end-to-end timing constraints are emphasized, are part of our ongoing research and experimental activities and will be presented in future work.

4.2 Scenario: Remote central locking test

To exemplify the relevance of our proposed framework, we introduce a test scenario that evaluates the proper operation of the central door locking functionality. The scenario involves the central body control ECU, known as the body control module (BCM), and

two door ECUs provided by distinct suppliers at separate locations. The assessment focuses on the dynamic interaction among these ECUs in response to a vehicle locking command. Under normal circumstances, remote testing of such dynamic interactions would be challenging due to the time delays caused by the long-distance distribution links and the intricate coordination required between the ECUs to effectively exchange information. Accordingly, latency compensation becomes necessary.

The involved nodes are described as follows: c, e_{BCM}, e_{D1}, e_{D2}, gw_{BCM}, gw_{D1}, gw_{D2} denote the central OEM node, the BCM ECU, the two door ECUs, and their gateways, respectively. Following that, the sequential steps for evaluating the scenario, along with the exchanged messages, are described and depicted in Figure 3:

STEP 1: *Time Synchronization:* The central node initiates synchronization across all three layers, ensuring time alignment among all nodes. ▷ *Messages:* $sync\langle c, gw_{BCM}\rangle \Rightarrow sync\langle gw_{BCM}, e_{BCM}\rangle$, $sync\langle c, gw_{D1}\rangle \Rightarrow sync\langle gw_{D1}, e_{D1}\rangle$, $sync\langle c, gw_{D2}\rangle \Rightarrow sync\langle gw_{D2}, e_{D2}\rangle$.

STEP 2: *Locking Request to BCM:* The central node c at level 3 timestamps and triggers the Body Control Module (BCM) ECU with the input simulated signal for locking the vehicle. Upon receiving the encapsulated segment, the BCM gateway timestamps, unpacks, and injects the input frame into the CAN bus. ▷ *Messages:* $t_1\langle c\rangle$, $s\langle c, gw_{BCM}\rangle$, $t_1\langle gw_{BCM}\rangle$, $f\langle gw_{BCM}, e_{BCM}\rangle$.

STEP 3: *BCM Response:* The BCM gateway captures the response from the BCM ECU, generates a second timestamp, and encapsulates both the response frame and the two timestamps in the response segment. Upon arrival at the central node, the central node timestamps, unpacks the segment, and calculates the link delays using the four timestamps. ▷ *Messages:* $f\langle e_{BCM}, gw_{BCM}\rangle$, $t_2\langle gw_{BCM}\rangle$, $s\langle gw_{BCM}, c\rangle$, $t_2\langle c\rangle$.

STEP 4: *Door ECUs' Trigger:* The central node timestamps and triggers the door ECUs with the response received from the BCM ECU. Upon receiving the encapsulated segment, the gateways timestamp, unpack, and inject the frame into their respective CAN bus. ▷ *Messages:* $t_3\langle c\rangle$, $s\langle c, gw_{D1}\rangle$, $s\langle c, gw_{D2}\rangle$, $t_1\langle gw_{D1}\rangle$, $t_1\langle gw_{D2}\rangle$, $f\langle gw_{D1}, e_{D1}\rangle$, $f\langle gw_{D2}, e_{D2}\rangle$.

STEP 5: *Door ECUs' Response:* Each of the ECU gateways captures the response from the respective door ECU, generates a second timestamp, and encapsulates both the frame and the two timestamps in the response segment. Upon arrival at the central node, the central node timestamps, unpacks each of the two segments, and calculates the link delays. ▷ *Messages:* $f\langle e_{D1}, gw_{D1}\rangle$, $f\langle e_{D2}, gw_{D2}\rangle$, $t_2\langle gw_{D1}\rangle$, $t_2\langle gw_{D2}\rangle$, $s\langle gw_{D1}, c\rangle$, $s\langle gw_{D2}, c\rangle$, $t_4\langle c\rangle$, $t_5\langle c\rangle$.

STEP 6: *BCM Trigger:* The central node timestamps and triggers the BCM ECU with the door ECU responses. Upon receiving the encapsulated frame, the gateway timestamps, unpacks, and injects

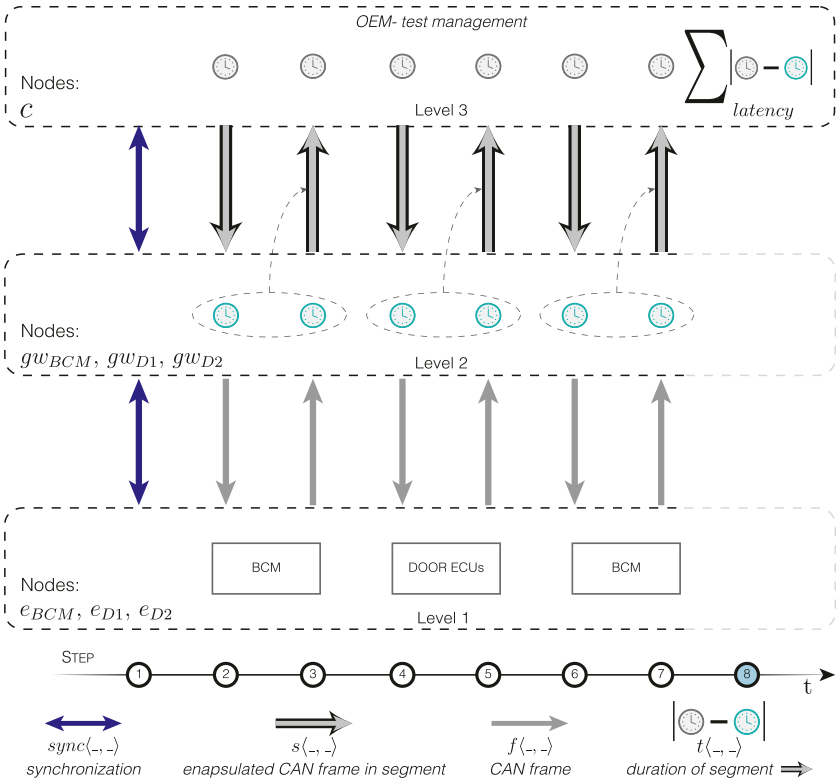

FIGURE 3
Central door locking: Stepwise procedure for remote evaluation. <_, _> denotes an interaction between any two adjacent nodes, involving synchronization, frame exchange, encapsulated frame exchange (segment), and timestamping.

the segments into the CAN bus. ▷ *Messages:* $t_6\langle c\rangle$, $s\langle c, gw_{BCM}\rangle$, $t_3\langle gw_{BCM}\rangle$, $f_{d1}\langle gw_{BCM}, e_{BCM}\rangle$, $f_{d2}\langle gw_{BCM}, e_{BCM}\rangle$.

STEP 7: *BCM Response:* The BCM gateway captures the response from the BCM ECU, generates a second timestamp, and encapsulates both the response frame and the two timestamps in the response segment. Upon arrival at the central node, the central node unpacks the segment, timestamps, and calculates the link delays using the four timestamps. ▷ *Messages:* $f\langle e_{BCM}, gw_{BCM}\rangle$, $t_4\langle gw_{BCM}\rangle$, $s\langle gw_{BCM,c}\rangle$, $t_6\langle c\rangle$.

STEP 8: *Latency Correction and Verification:* The central node timestamps, calculates the total latency delay, and evaluates the scenario's outcome based on two conditions: 1) the final BCM response matches the CAN frame indicating all doors are locked, and 2) the corrected execution time of the scenario falls within the predefined limit. ▷ *Messages:* $t_7\langle c\rangle$.

The corrected execution time of the scenario is calculated as the difference between the first and last timestamps recorded by the central node, excluding the total duration of all latencies introduced by the messages exchanged between levels 2 and 3:

$$t_{corr} = (t_7\langle c\rangle - t_1\langle c\rangle) - \sum |s\langle i, j\rangle|,$$

where. $\langle i, j\rangle \in \{\langle c, gw_{BCM}\rangle, \langle c, gw_{D1}\rangle, \langle c, gw_{D2}\rangle, \langle gw_{BCM}, c\rangle, \langle gw_{D1}, c\rangle, \langle gw_{D2}, c\rangle\}$

To clarify the latency compensation strategy, Figure 4 illustrates a comparison among three different testing setups, all performing the same test scenario: (1) a local testing setup, where all devices under test are co-located and interconnected via a local communication bus, typical of conventional testing methods; (2) a global testing setup, where devices are geographically distributed and interconnected through the Internet, but no latency compensation measures are employed; and (3) a global testing setup implementing the proposed hybrid synchronization approach. In each scenario, the test involves ensuring the timely arrival of a sequence of messages, m_1 to m_n, within a specified time interval, t_{REQ}, marked by a blue vertical line. The dotted lines, labeled t_{exec} and t_{test}, represent global time and testing time, respectively. In the local setup (1), the final message, m_n, arrives within the t_{REQ} interval, indicating a successful test. However, in setup (2), latency-induced delays cause the final message to arrive after t_{REQ}, leading to a test failure. As discussed earlier, this latency inherently limits the scope of remote testing to basic functional cases that do not require strict timing adherence. In setup (3), where the proposed hybrid approach is applied, messages are timestamped, allowing for *post hoc* reconciliation and correction of latency-induced discrepancies. This process decouples testing time from global time, enabling t_{REQ} to be shifted forward on the timeline. As a result, the test's success is evaluated relative to the decoupled t_{test} axis rather than the t_{exec} global time axis, which would have otherwise led to failure. This approach to latency compensation helps align the global testing setup more closely with the local setup.

5 Further considerations

This section discusses three aspects surrounding the proposed approach, including a consideration of potential impediments: 1) *distributed system traits* 2) *synchronization accuracy*, and 3) *performance requirements*.

5.1 Distributed system traits

The need to decouple ECU testing from very few, usually OEM-operated locations, where all components are traditionally co-

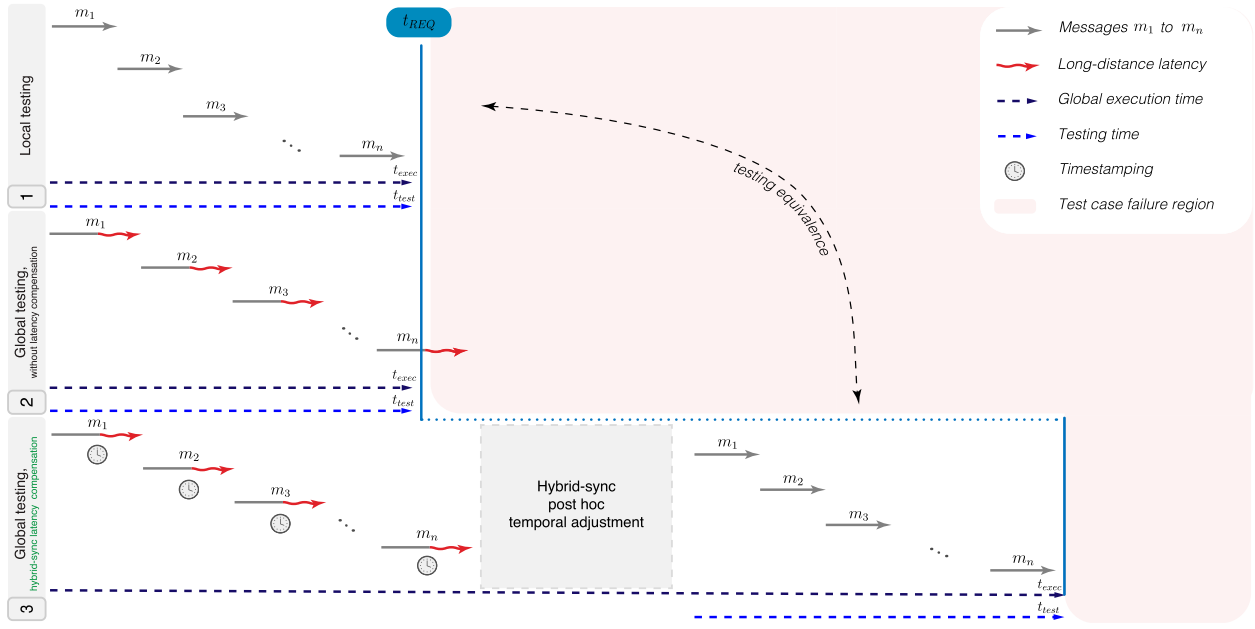

FIGURE 4
Local vs. global testing: the proposed hybrid approach compensates for latency-induced delays in long-distance interconnections, aligning results with local setups.

located, proposes the notion of an inherently distributed system with resources spanning across the globe. One challenge in the proposed architecture is replicating the locally broadcast functionality of CAN over the Internet, namely, at higher networking layers. Despite CAN not offering absolute guarantees of timely message arrival, all nodes must still actively engage in the exchange by receiving all messages and responding accordingly. This concept of *all or none* underscores the principle of *atomic commitment*, which denotes interactions or message multicasts occurring in non-overlapping epochs, within a certain *group view*—a perspective on the set of recipients the sender engaged with at the time of multicast Kleppmann (2017). For example, a group view may include the remote ECUs involved in a test case. While *atomicity* typically requires bounded timings and failure detectors, it is still feasible to apply it to asynchronous environments Kragl et al. (2018). As such, our approach, promotes iterative capturing and replaying of messages to all nodes, following a piece-wise execution model. Derived from the recovery concept in distributed systems, this approach is known as the combination of *checkpointing* and *message logging*, enabling nodes to progress from one state to another by replaying a set of messages from a previous state. This, along with coordination from a single leader node, leads to a globally-consistent *distributed checkpoint* van Steen and Tanenbaum (2023). Acting as the primary node, the OEM can gather the checkpoints, reassemble all the events comprising the test case, and compensate for latency through timestamp recording. Such techniques stem from the necessity for fault tolerance and recovery in distributed settings. However, our suggested approach can accommodate a less robust form of both: should a node fail, manual failover mechanisms can be used to restart the test case execution in a controlled manner.

5.2 Synchronization accuracy

Synchronization is a pivotal component in our proposed framework, as it enables time-tracking of interactions among nodes at different hierarchy levels. This, in turn, facilitates the asynchronous processing of messages and the correction of the overall latency at subsequent time points. As previously discussed in the background section, PTP and its revisions offer nanoseconds accuracy, ensuring effective coordination in numerous applications. However, synchronization handshakes largely assume symmetrical links and local proximity nodes, which are challenging to meet in remote testing environments due to the inherent delay and unpredictability of long-distance communication. As our approach relies on a hierarchical structure, failing to achieve precise synchronization between the central node and the gateways also impacts the accuracy of the lower level synchronization between the gateways and the ECUs.

The topic of asymmetry in PTP handshakes is discussed in a growing body of literature. Several approaches, such as individual delay link calculation Lee et al. (2012), buffered propagation delay Lv et al. (2010), fixed delay ratios Du et al. (2011), block burst transmission Lee (2008), and bias estimation Hajikhani et al. (2014), reported enhanced accuracy and performance. Nevertheless, none of the synchronization protocols, including PTP, intrinsically aligns with the long-distance interconnection requirements, whereas in their work, Tan and Wu (2021) argue that synchronization at long distances faces critical challenges, making syntonization the preferred choice. Notwithstanding, further research incentives should explore the potential of long-distance synchronization and its practical applicability within the context of remote testing.

5.3 Performance requirements

Integrating two heterogeneous networks while preserving their distinct characteristics demands substantial computational efforts and technical resources. To address the heterogeneity between a lower-level communication bus and higher-level exchanges over the Internet, our approach requires gateways to handle multiple tasks of varying processing complexity. From the perspective of time synchronization, the precision demands of PTP, especially in its later revisions, necessitate specialized hardware capable of achieving accuracy in the nanoseconds range. However, given the cost-sensitive nature of the automotive industry, existing hardware solutions, especially in the form of in-vehicle gateways, are constrained by hardware limitations. As a result, concurrent timestamping operations and message processing necessitate additional computational resources. Moreover, the reliability of the proposed framework can be impacted by inadequate software algorithms for timestamping, especially in the context of the CAN mechanism.

The increasing interest in advanced embedded systems and the emergence of high-performance hardware platforms, such as Advanced Field Programmable Gate Arrays (FPGAs), offer potential solutions for meeting stringent performance demands. As research and development efforts advance in the automotive domain, the prospect of more accessible gateways with enhanced capabilities that could align with the needs of remote testing becomes feasible.

6 Conclusion

Transitioning from a local testing setup with co-located components to a global testing framework with distributed resources is inherently entwined with networking limitations imposed by long-distance interconnections. Through the lens of the Internet of Things (IoT), where ECUs can be perceived as geographically dispersed IoT components and their remote testing as an IoT distributed process, we introduced a testing framework that enables the integration of physical ECUs over the Internet among automotive suppliers and OEMs.

In response to a limitation prevalent in a range of distributed IoT settings—latency—we presented a hybrid approach that leverages synchronous timestamping techniques alongside asynchronous event processing. By timestamping each event or message exchange, the total latency introduced in the Internet-distributed links is calculated, and the execution time of the test case is adjusted accordingly, echoing the characteristics of local testing environments. This proves advantageous in contexts where the focus shifts from real-time evaluation, which would otherwise be constrained by latency, to a deferred evaluation and *post hoc* reconciliation of system operations and outcomes.

To demonstrate the approach's applicability, we showcased a scenario of remote evaluation for the door-locking functionality,

supported by a mathematical model describing node interactions, flows, and messages. In addition, we discussed additional considerations and impediments surrounding the proposed distributed framework. Future endeavors will aim to formalize the notion of equivalence between local and global testing environments through communication calculus and to empirically validate the proposed framework via an end-to-end latency analysis.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding author.

Author contributions

ST: Conceptualization, Investigation, Methodology, Visualization, Writing–original draft, Writing–review and editing. PH: Writing–review and editing. MW: Writing–review and editing.

Funding

The author(s) declare that financial support was received for the research, authorship, and/or publication of this article. This publication is based on the research project (SofDCar) (19S21002), which is funded by the German Federal Ministry for Economic Affairs and Climate Action.

Acknowledgments

We extend our sincere gratitude to Marco Aiello for the valuable discussions and constructive suggestions.

Conflict of interest

Authors ST and PH were employed by Mercedes-Benz AG.

The remaining author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

AUTOSAR (2021). Specification of time synchronization over CAN. *Tech. Rep.* 674 (Release R21-11).

Bandur, V., Selim, G., Pantelic, V., and Lawford, M. (2021). Making the case for centralized automotive E/E architectures. *IEEE Trans. Veh. Technol.* 70, 1230–1245. doi:10.1109/TVT.2021.3054934

Briscoe, B., Brunstrom, A., Petlund, A., Hayes, D., Ros, D., Tsang, I.-J., et al. (2016). Reducing Internet latency: a survey of techniques and their merits. *IEEE Commun. Surv. Tutorials* 18, 2149–2196. doi:10.1109/comst.2014.2375213

Chin, T., Rahouti, M., and Xiong, K. (2017). "End-to-End delay minimization approaches using software-defined networking," in *Proceedings of the international Conference on Research in Adaptive and convergent systems (ACM)*. doi:10.1145/3129676.3129731

Ciravegna, L., and Pilkington, A. (2013). Outsourcing practices in automotive supply networks: an exploratory study of full service vehicle suppliers. *Int. J. Prod. Res.* 51, 2478–2490. doi:10.1080/00207543.2012.746797

Davare, A., Zhu, Q., Di Natale, M., Pinello, C., Kanajan, S., and Sangiovanni-Vincentelli, A. (2007). "Period optimization for hard real-time distributed automotive systems," in *2007 44th ACM-IEEE design automation conference*, 278–283.

Dengler, F., Eichler, B., and Dagli, A. (2021). "Acceleration of ECU-development by using connected mixed reality environments," in *21. Internationales stuttgarter symposium*. Editors M. Bargende and H.-C. Reuss (Wiesbaden: Wagner).

Domski, W. (2022). Remote laboratory offered as hardware-as-A-service infrastructure. *Electronics* 11, 1568. doi:10.3390/.electronics11101568

Du, Z., Lu, Y., and Ji, Y. (2011). An enhanced end-to-end transparent clock mechanism with a fixed delay ratio. *IEEE Commun. Lett.* 15, 872–874. doi:10.1109/LCOMM.2011.062911.110918

Dürr, M., Brüggen, G. V. D., Chen, K.-H., and Chen, J.-J. (2019). End-to-End timing analysis of sporadic cause-effect chains in distributed systems. *ACM Trans. Embed. Comput. Syst.* 18, 1–24. doi:10.1145/3358181

European Parliament Brown, D., Flickenschild, M., Mazzi, C., Gasparotti, A., Panagiotidou, Z., Dingemanse, J., et al. (2021). The future of the EU automotive sector. doi:10.2861/680215

Fedullo, T., Morato, A., Tramarin, F., Rovati, L., and Vitturi, S. (2022). A comprehensive review on time sensitive networks with A special focus on its applicability to industrial smart and distributed measurement systems. *Sensors* 22, 1638. doi:10.3390/s22041638

Feiertag, N., Richter, K., Nordlander, J., and Jönsson, J. (2008). A compositional framework for end-to-end path delay calculation of automotive systems under different path semantics

Hajikhani, M., Kunz, T., and Schwartz, H. (2014). "A recursive method for bias estimation in asymmetric packet-based networks," in *2014 IEEE international symposium on precision clock synchronization for measurement, control, and communication (ISPCS)*. doi:10.1109/ISPCS.2014.6948530

IEEE (2002). IEEE standard for a precision clock synchronization protocol for networked measurement and control systems. doi:10.1109/IEEESTD.2002.94144

IEEE (2020). IEEE standard for local and metropolitan area networks–timing and synchronization for time-sensitive. doi:10.1109/IEEESTD.2020.9121845

IEEE (2023). International Standard–Information technology – telecommunications and information exchange between systems – local and metropolitan area networks – Part 1BA: audio video bridging (AVB) systems. doi:10.1109/.IEEESTD.2023.10108543

Johanson, M., and Karlsson, L. (2006). A framework for distributed collaborative automotive testing

Kim, J. H., Seo, S.-H., Hai, N.-T., Cheon, B. M., Lee, Y. S., and Jeon, J. W. (2015). Gateway framework for in-vehicle networks based on CAN, FlexRay, and Ethernet. *IEEE Trans. Veh. Technol.* 64, 4472–4486. doi:10.1109/TVT.2014.2371470

Kleppmann, M. (2017). *Designing data-intensive applications* (O'reilly)

Kragl, B., Qadeer, S., and Henzinger, T. A. (2018). Synchronizing The Asynchronous (Schloss Dagstuhl – Leibniz-Zentrum für Informatik). doi:10.4230/LIPICS.CONCUR.2018.21

Kumar, S., Dalal, S., and Dixit, V. (2014). The OSI Model: overview on the seven layers of computer networks. *Int. J. Comput. Sci. Inf. Technol. Res.* 2, 461–466.

Lee, S. (2008). An enhanced IEEE 1588 time synchronization algorithm for asymmetric communication link using block burst transmission. *IEEE Commun. Lett.* 12, 687–689. doi:10.1109/LCOMM.2008.080824

Lee, S., Lee, S., and Hong, C. (2012). An accuracy enhanced IEEE 1588 synchronization protocol for dynamically changing and asymmetric wireless links. *IEEE Commun. Lett.* 16, 190–192. doi:10.1109/LCOMM.2011.092011.110582

Lévesque, M., and Tipper, D. (2016). A survey of clock synchronization over packet-switched networks. *IEEE Commun. Surv. and Tutorials* 18, 2926–2947. doi:10.1109/COMST.2016.2590438

Lv, S., Lu, Y., and Ji, Y. (2010). An enhanced IEEE 1588 time synchronization for asymmetric communication link in packet Transport network. *IEEE Commun. Lett.* 14, 764–766. doi:10.1109/LCOMM.2010.08.091601

Mills, D. (1991). Internet time synchronization: the network time protocol. *IEEE Trans. Commun.* 39, 1482–1493. doi:10.1109/26.103043

Morishima, K., Sugure, Y., and Miyazaki, Y. (2018). "Evaluation of parallel executions on multiple virtual ECU systems," in SAE technical paper series *(SAE international)* (Warrendale, Pennsylvania: ANNUAL). doi:10.4271/2018-01-0011

Mubeen, S., Mäki-Turja, J., and Sjödin, M. (2014). Communications-oriented development of component-based vehicular distributed real-time embedded systems, 60, 207, 220. doi:10.1016/j.sysarc.2013.10.008

Phatak, S. S., Chen, H., Xiao, Y., Wang, C., McCune, D., Schliecker, S., et al. (2016). "Virtual multi-ECU high fidelity automotive system simulation," in SAE technical paper series *(SAE international)* (Warrendale, Pennsylvania: ANNUAL). doi:10.4271/2016-01-0013

Rajeev, A. C., Mohalik, S., and Ramesh, S. (2012). "Verifying timing synchronization constraints in distributed embedded architectures," in *2012 design, automation and test in europe conference and exhibition*, 200–205. doi:10.1109/DATE.2012.6176463

Scherer, B. (2022). "Remote or virtual laboratory for HIL (Hardware-In-the-Loop) testing education," in *2022 23rd international carpathian control conference (ICCC)*, 167–170. doi:10.1109/ICCC54292.2022.9805976

Schmidt, S., Henning, J., Wambera, T., and Kronimus, S. (2015). Early PC-based validation of ecu software using virtual test driving. *ATZelektronik Worldw.* 10, 14–17. doi:10.1007/s38314-015-0508-y

Sobotka, J., and Novák, J. (2013). "Automation of automotive integration testing process," in *2013 IEEE 7th international conference on intelligent data acquisition and advanced computing systems (IDAACS)*, 01, 349–352. doi:10.1109/IDAACS.2013.6662704

Tan, W., and Wu, B. (2021). Long-distance deterministic transmission among TSN networks: converging CQF and DIP. *Comput. Res. Repos. (CoRR).* doi:10.48550/arXiv.2111.03246

Tziampazis, S., Kopp, O., and Weyrich, M. (2023). "Distributed integration of electronic control units for automotive OEMs: challenges, vision, and research directions," in *2023 IEEE 20th international conference on software architecture companion (ICSA).* doi:10.1109/ICSA-C57050.2023.00068

van Steen, M., and Tanenbaum, A. (2023). *Distributed systems*. 4th ed. (distributed-systems.net).

Waldhauser, S., Jaeger, B., and Helm, M. (2020). Time synchronization in time-sensitive networking. *Network* 51, 51–56. doi:10.2313/NET-2020-04-1_10

Walrand, J., Turner, M., and Myers, R. (2021). An architecture for in-vehicle networks. *IEEE Trans. Veh. Technol.* 70, 6335–6342. doi:10.1109/TVT.2021.3082464