An Open-Source Testbed for Next-Generation In-Vehicle Zonal Architectures

Jan de Voor*, Levent Çelik*, Johannes N. Weidner*, Mohammad Hamad§, Ameer Kashani†, Richard Brooks*, and Mert D. Pesé*
*Clemson University, {jdevoor, lcelik, jnweidn, rrb, mpese}@clemson.edu
§Technical University of Munich, mohammad.hamad@tum.de
†DENSO International America, Inc., ameer.kashani@na.denso.com

***Abstract*—Rising demand for scalable, efficient In-Vehicle Networks (IVNs) capable of operating with increasingly complex tasks and features has led to the emergence of the Zonal Architecture (ZA). As ZA and its enabling technologies mature, the ability to conduct comprehensive full system evaluations will become increasingly valuable in driving positive progress. In this paper, we present Zonal Architecture Testbed (ZAT), an open-source software and hardware platform designed to enable low-cost, effective, and scalable development and evaluation for the next generation of IVNs. The ZAT bridges the gap between software simulation and hardware prototyping by implementing a portable software suite designed for automotive-grade hardware and minimal-cost single-board computers alike, enabling users to easily implement custom applications for ZA systems. We demonstrate the adaptability of the ZAT by presenting three use cases spanning a range of research areas in ZA IVNs, including topology and security protocol analysis, dataset generation, and priority-based networking performance of vehicle applications.**

***Index Terms*—testbed, zonal architecture, automotive ethernet**

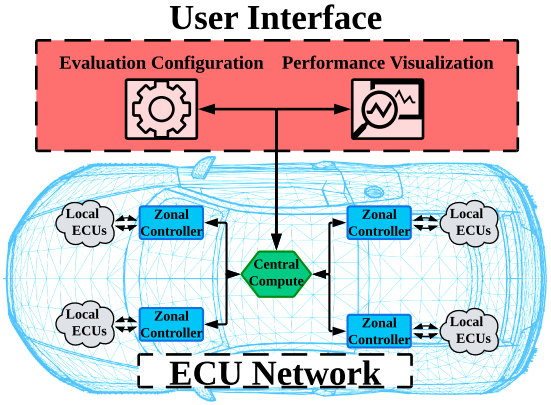


Fig. 1: The Zonal Architecture Testbed, designed to promote further research in zonal architecture by minimizing development costs and enabling adaptable networking evaluations without a full-scale vehicle platform.

I. Introduction

Recent advancements in vehicle functionality, driven by innovations in computational systems, have compelled Original Equipment Manufacturers (OEMs) to reevaluate traditional IVN architectures. Growing consumer demand for increasingly complex features [1], such as Advanced Driver Assistance Systems (ADASs) and external connectivity, requires substantial resources to deploy. The significant load placed on IVNs by these high-bandwidth feature sets has led to increased efforts across industry and academia for ZA to provide scalable solutions [2]. The advancements provided by ZA are further supported by enabling technologies such as Automotive Ethernet (AE), which provides the high-speed, deterministic communication backbone required to implement and validate safety-critical systems while addressing cybersecurity challenges. However, the shift to ZA and AE has exposed a growing gap between academic research and industrial needs.

Historically, academic efforts on IVN networking have focused on Controller Area Network (CAN), FlexRay, and Local Interconnect Network (LIN), legacy communication protocols which lack the bandwidth required for network-intensive feature sets in modern Electrical/Electronic (E/E) architectures [3]. Similarly, prior academic work on IVNs has predominantly explored centralized and domain architectures, with evaluations of ZA emerging only recently as industry interest has accelerated. In contrast to existing architectures, ZA reduces production costs by consolidating Electronic Control Unit (ECU) subnetworks into zones clustered by physical location [1]. However, ZA introduces new challenges, particularly in topological optimization for applications with strict real-time deadlines [4]. While the scarcity of academic work for ZA is explained by its novelty, the primary barrier lies in the prohibitive costs of hardware prototyping and the lengthy development cycles required to build functional systems.

In this paper, we introduce the Zonal Architecture Testbed (ZAT), a general-purpose evaluation platform designed for modularity and adaptability to address the challenges of ZA research by significantly lowering barriers to entry. We provide a software suite that runs across a range of hardware, enabling researchers to explore ZA without relying on proprietary solutions. Our platform provides a flexible approach for evaluating ZA network behavior, enabling researchers to dynamically generate traffic across a configurable topology while analyzing network responses via an integrated User Interface (UI). By

streamlining the development process, the ZAT reduces the time and cost typically associated with ZA implementations, allowing researchers to focus on advancing the state of the art.

We evaluate the feasibility of the ZAT using real automotive hardware across three core research areas: topology and security protocol overhead analysis, dataset generation, and the performance of vehicle applications operating in priority-based communication schemes. We further provide sample datasets generated by the ZAT for the evaluation, alongside the 3D-printed server-rack styled design files used for our deployment of the ZAT in our public GitHub repository [5].

Our contributions are threefold:

- To the best of our knowledge, we present the first hardware-based ZA testbed, complemented by an integrated UI for control and visualization [Section III];
- We implement a publicly available software stack using exclusively open-source components [Section IV];
- We evaluate the ZAT's viability through core research areas in topology analysis, dataset generation, and the performance of vehicle applications across a range of priority-based network processing. [Section V].

II. Background and Related Work

A. In-Vehicle Network Architectures

Traditional IVNs. Traditional IVN architectures evolved from purely mechanical systems to highly electronic networks consisting of up to hundreds of ECUs, many of which are low-cost and single-purpose, connected through a central gateway. While effective for their time, traditional architectures have become increasingly complex and costly to maintain as ECU count and inter-dependencies have grown [2], [6]–[8].

Zonal Architecture. The shift to ZA represents a more radical departure from traditional methods, organizing ECUs into zones based on their physical location within the vehicle. ZA networks consist of at minimum one a Central Computer (CC) and multiple zones, with each zone containing clusters of ECUs managed by powerful Zonal Controllers (ZCs). The CC serves as a management and high-performance computing system, coordinating communication between zones while handling complex computational tasks. The CC and the ZCs are interconnected via AE, enabling high-bandwidth communication across the network. Within zones, traditional communication protocols such as CAN and LIN may still be used, with ZCs facilitating protocol translation between the zones and AE as needed. This approach not only simplifies network cabling and reduces weight but also enhances scalability while supporting the high bandwidth demands of modern applications such as ADASs and autonomous driving [7], [8].

Automotive Ethernet. Traditional IVNs have relied on legacy communication protocols such as CAN [9], LIN [10], FlexRay [11], and Media Oriented Systems Transport (MOST) [12], each developed to meet specific communication needs within vehicles. AE, adapted from regular Ethernet, has been designed to meet the stringent demands of automotive applications [3], [13]. Unlike conventional Ethernet, which is designed for high throughput and general-purpose connectivity, AE provides Time-Sensitive Networking (TSN) [14] to ensure deterministic, low-latency communication required for safety-critical functions [15], while offering significantly higher bandwidth than legacy protocols.

TABLE I: Comparison of Automotive Testbed Platforms.

Design Principle	PASTA [19]	SANE [17]	VirtoBench [20]	**ZAT**
Adaptable	●	●	○	●
Generalizable	◒	●	○	●
Cyber-Physical	●	○	●	●
Focus on SDVs	○	●	○	●

○ No, ◒ Partially True, ● Yes

B. Automotive Testbed Platforms

The shift towards vehicle systems in which functionality is provided by software decoupled from fixed physical hardware has introduced the concept of Software Defined Vehicles (SDVs). This decoupling allows OEMs to reduce their reliance on specific hardware throughout the design process, leading towards a pursuit of generalized design environments for a much broader range of vehicles.

Software-in-the-Loop (SIL) development is one such manifestation of these environments, in which vehicle processes are simulated through software, accelerating development while reducing the costs associated with relying primarily on hardware for design and test procedures. This software-first philosophy is reflected in emergent solutions that digitally approximate vehicle hardware using digital twin systems capable of simulating various aspects of automotive platforms [16]. Wu *et Al.* [17] proposed one such framework for emulating automotive networks capable of scalable integration of vehicle functionality. However, a pure software approach is primarily effective in the early development phases rather than in the later validation stages, where measuring real-time interactions between the software and hardware is critical [18].

Toyama *et Al.* [19] introduced an open, adaptable testbed platform designed to provide adaptable methods to simulate vehicle networks with real automotive hardware, although the publicly available version is limited in its implementation for SDVs as it is designed for traditional CAN networks. Built with commercial off-the-shelf (COTS) ECUs, Yeo *et Al.* [20] developed a test platform focused on replicating the network conditions of a vehicle while driving. However, the testbed was specifically designed for security research, limiting functionality to manipulating the IVN via an FPGA-based communication board that operates the various CAN networks.

III. System Design

We designed ZAT to promote research in ZA through a testbed platform compatible with a variety of users and experimentation scenarios, and we centered our design philosophy on the four core principles described as follows.

Adaptable. Automotive-grade hardware differs extensively in computational power, networking capabilities, proprietary

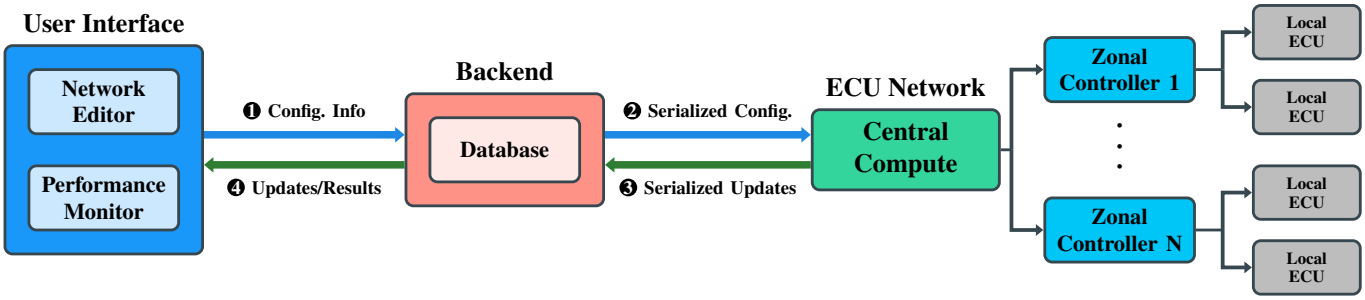


Fig. 2: ZAT Application Design

technologies, and price. The ZAT must therefore be compatible across a wide range of hardware configurations by carefully abstracting the underlying Hardware Abstraction Layers (HALs) specific to a given platform, such that each ECU in the ZAT can be supported by the same software suite.

Generalizable. Given the various open questions on the impacts of ZA and an IP-based IVN, the ZAT must be generalizable to novel research questions. The ZAT cannot provide functionality for a single domain, and the user interface must support the integration of user-defined test conditions.

Cyber-Physical. Simulating vehicle conditions in software is complex and leads to a confluence of approximations which, while suitable for certain applications, is not always reflective of real-world deployments due to hardware behavior that is difficult to realistically model in software [18]. Therefore, the ZAT must be capable of performing hardware evaluations to measure the characteristics of cyber-physical systems.

Focus on SDVs. The growing interest in SDVs designed with ZA and powered by emerging technologies has presented a variety of novel research questions with respect to the performance, reliability, and maintainability of this new generation of vehicles. As SDVs with ZA require a new IVN architecture and associated development philosophies, the ZAT needs to be designed to maximize the capabilities of automotive Ethernet, software-defined principles, and ZA network structures.

We compare our design principles against existing work, shown in Table I, and identify a gap for testbed platforms that can be used across the variety of research topics open to ZA SDVs. With these principles in mind, we describe ZAT by dividing the system design into two parts: the application design that underlies the ZAT's process flow, and the data flow design that ties the UI, backend, and ECU network together.

A. Application Design

We define the ZAT into three main components: a user interface, backend, and ECU network, as shown in Fig. 2. The UI serves to control the interaction between a user and the ZAT, enabling the topology, network behavior, and network characteristics of the ZAT to be tractable. The backend serves as the connection layer between the UI and the ZAT hardware, storing and communicating the configuration of an evaluation to the ZAT hardware, then similarly storing and reporting the results to the UI. The ECU network is defined as the hardware used for a given evaluation, in which a CC interacts with the backend to exchange evaluation information, then distributes and requests evaluation data to the ECUs defined for that evaluation. We fully define each component as follows.

User Interface. The user-facing aspect of ZAT interface offers test-engineering and data analysis functionality. It provides two main views: a network configuration interface and a performance monitoring dashboard. The network configuration interface enables users to visualize and manipulate network topologies, adding, modifying, and deleting ECUs and connections. Each connection can be configured with specific parameters, including IP addresses, security protocols, traffic-generation settings, and traffic-priority classes. The interface further provides functionality for managing multiple network configurations and initiating evaluations. The performance monitoring dashboard displays individual ECU performance data, such as CPU utilization, as well as connection data, including latency, bandwidth, and packet loss rates.

Backend. The backend serves as an intermediary between the UI and the hardware, acting as a translator to communicate a user's desired inputs and outputs for each evaluation. A database managed by the backend stores all persistent data, such as ECU connections and network parameters, trial results, hardware status logs, and user settings. The method of communication between the UI and the ECU network is defined to be easily modified, using standard open-source protocols and technologies to enable new evaluation techniques.

ECU Network. The hardware composing the ECU network is responsible for executing evaluations according to the user specification and reporting performance data to the backend. Each ZAT network must have one CC that serves as an intermediary between the backend and the rest of the ECU network. This CC must be capable of receiving and distributing evaluation requirements from the backend while also receiving performance data from the ECU network and sending it to the backend. In line with our focus on modularity and flexibility, we do not define requirements for the remaining hardware in the ECU network beyond the use of the ZAT software suite.

B. Data Flow Design

For any ZAT evaluation, data is transmitted across four stages. ❶ Upon the design of an evaluation, all appropriate configuration data is sent to the backend using HTTP requests,

including the ECUs involved, the connections between them, all traffic settings, and the timing for each evaluation. ❷ Once the backend receives the data for an evaluation, it first stores all persistent data to the database for future reference. Then, it serializes and transmits the evaluation settings using the format expected by the CC intermediary. ❸ When the CC receives the evaluation information, it will distribute the data to each ECU and begin the evaluation process. Throughout and after the evaluation, the CC will continuously send the backend performance data in the same serialized format in which it was received. ❹ The data received by the backend from the CC will be continuously stored in the database while simultaneously providing the updates to the user interface.

IV. Implementation

A. User Interface

We designed the ZAT for server rack-style deployment, combining a publicly available rack system [21] with custom-designed components, as shown in Fig. 3b. To allow for ease of operation using an integrated touchscreen display or a web interface, we developed the UI to run within Docker containers using Streamlit, an open-source Python framework focused on simplicity in creating web applications [22].

We developed two main UI pages, as shown in Fig. 3: one for configuring evaluation parameters and one for visualizing the performance of a given evaluation. Each page is designed to be modular, allowing a new parameter or visualization method to be introduced within the existing standard format once its respective HTTP endpoints are deployed. In the evaluation configuration page, shown in Fig. 3a, users can define an ECU's connections, set connection parameters, and specify traffic-generation characteristics. The performance monitor shown in Fig. 3c similarly displays latency, jitter, and throughput for each connection, based on the ECUs involved in the most recent evaluation.

B. Backend

We built the ZAT backend with *Flask*, which serves as the RESTful API layer. Endpoints use a database manager module that connects to a MariaDB instance running in a Docker container, encapsulating all database interactions. To facilitate communication with the CC, we use Scalable service-Oriented MiddlewarE over IP (SOME/IP), a service-based communication protocol with growing adoption for its ability to simplify communication among diverse components across vehicle applications [23]. Alongside Flask, the backend operates SOME/IP serialization and de-serialization with the Python library *someipy* [24] by spawning a new Python thread for the length of an evaluation to communicate directly with the CC, maintaining a consistent, standardized data format between the backend and the ZAT hardware.

C. ECU Network

The software suite powering the ZAT hardware is designed to minimize the development cost of applications for automotive platforms. We define a tiered abstraction structure in which low-level hardware-specific functionality is handled by a single set of increasingly high-level API calls across hardware platforms with portable open-source processing libraries.

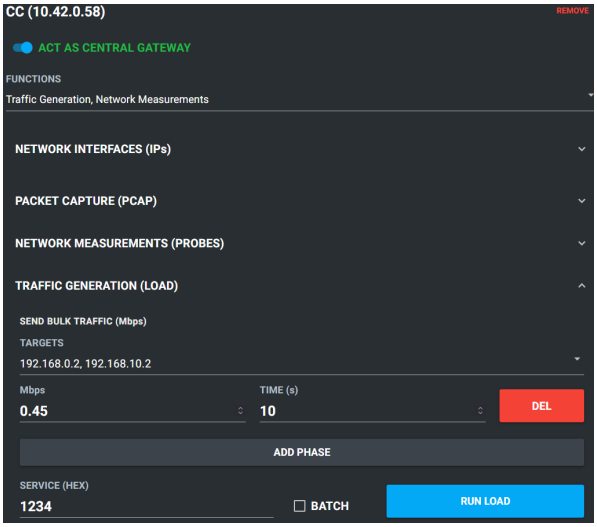
At the core of all hardware exists a set of HAL specific to its platform. The first abstraction tier for the ZAT software suite targets these HALs by handling Operating System (OS) and low-level networking calls. Specifically, we utilize a set of API calls that abstract OS calls to FreeRTOS, an open-source Real-Time OS that enables multitasking on a single-core processor, and Linux. We specifically defined OS abstraction to call FreeRTOS, Linux, or another integrated OS when built to do so using the same API calls, thereby ensuring applications retain the same functionality no matter the underlying OS or HAL. Further, we abstract networking calls using lwIP, an open-source lightweight networking stack specifically for resource-constrained devices, and the Linux socket networking layer. Both FreeRTOS and lwIP have extensive publicly available porting efforts across a wide range of hardware vendors, making them ideal to base the ZAT software on. Additionally, Linux has a vast user base with development efforts spanning several decades, making it an ideal alternative to FreeRTOS and lwIP for capable hardware in the ZAT.

Above the OS and networking layer, we define wolfSSL and wolfCrypt, open-source cryptographic libraries designed for embedded systems, to provide cryptographic functionality for the ZAT. While wolfSSL and wolfCrypt both depend on API calls defined in the OS and networking layers, the unique circumstances of automotive platforms require the ability to operate with various integrated Hardware Security Modules (HSMs). With wolfCrypt, we introduce the ability to define network security protocols, such as TLS with wolfSSL, that combine OS, networking, and cryptographic calls to secure data sent across the network by user applications operating at the final tier of the ZAT software suite.

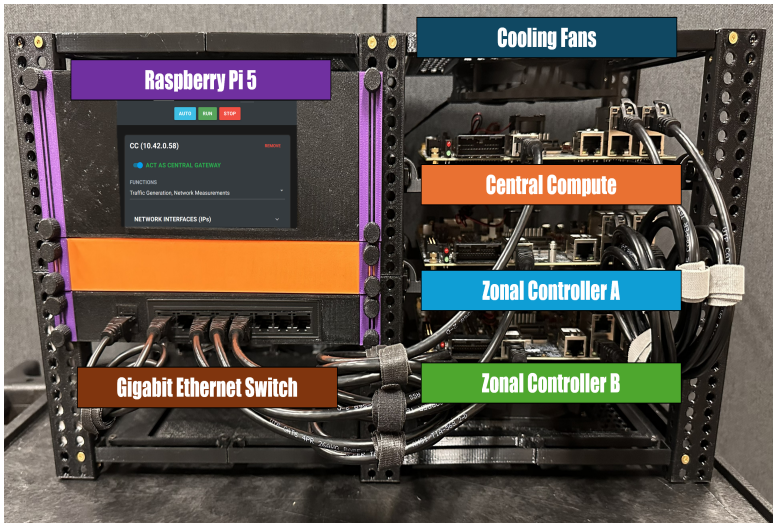
The user application space is designed to be user-friendly, allowing users to develop software without considering the low-level intricacies of each hardware platform or the method by which data is sent from their application to the networking layers and finally to another ECU in the ZAT. Within the user application space, each ECU runs a background SOME/IP task that receives and operates according to the evaluation configuration data specified by the user interface. Other background tasks handle networking-specific functionality, including security protocols, allowing user applications running in separate tasks to communicate with the networking layers without handling network data flows. As an evaluation runs, each ECU uses the same SOME/IP task to communicate performance metrics to the appropriate party, such as the backend for the CC or the CC for all other ECUs during evaluation.

V. Evaluation

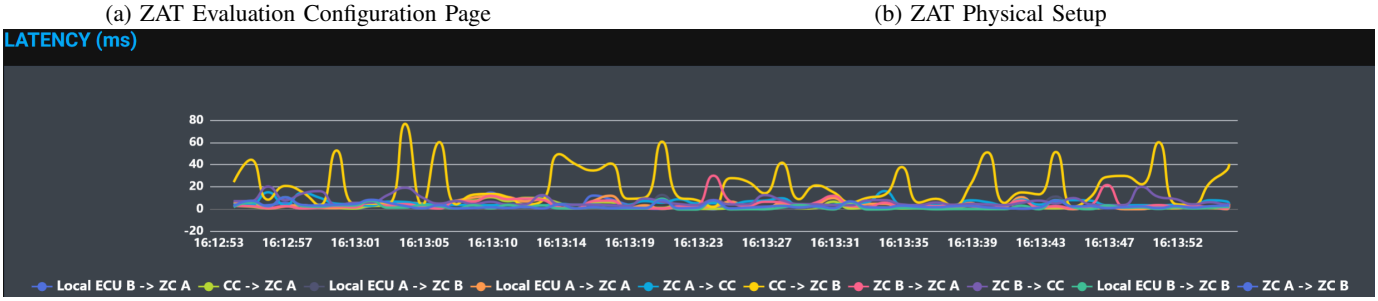
The emergence of ZA has raised several uncertainties for OEMs towards the adoption and deployment of a new networking architecture. The interaction between ECUs is fundamentally different from that in traditional IVNs, requiring the IVN to be re-examined in its entirety. While the



(a) ZAT Evaluation Configuration Page



(b) ZAT Physical Setup



(c) ZAT Performance Monitoring Page

Fig. 3: ZAT Setup consisting of one CC, two ZC, and one Raspberry Pi acting as two local ECUs. The ZCs and CC are connected via Gbps TSN links, while the ZCs and the CC are connected to the Raspberry Pi over a 1 Gbps switch.

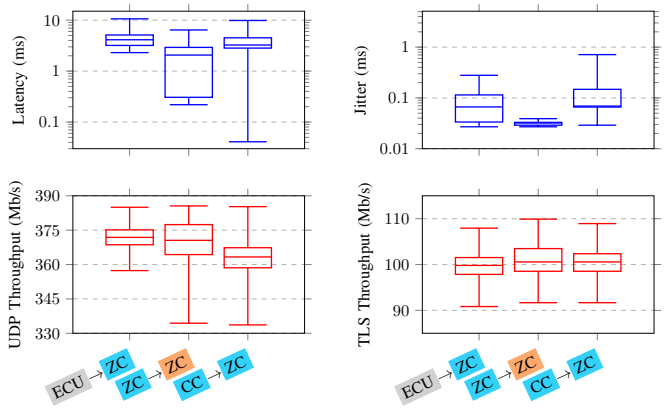


Fig. 4: Network and security performance between both local ECUs and their respective ZC, ZC A and ZC B, and the CC and ZCs across 50 iterations per experiment.

development of ZA continuously generates novel research questions, we demonstrate the feasibility of ZAT by evaluating its performance across three research areas: **(RA1)** network topology and security performance analysis, **(RA2)** dataset generation, and **(RA3)** time-sensitive application performance.

A. *Evaluation Setup*

For evaluation across all three research areas, the ZAT included the following: 3 NXP S32G-VNP-RDB3 reference design boards, one used as a central compute and two as zonal controllers, and one Raspberry Pi (RPi) 5 that doubles as the UI and two local ECUs, one for each zonal controller, in order to evaluate the ZAT's adaptability with consumer and automotive-grade hardware. The RPi is connected to all three NXP boards via an unmanaged Gigabit Ethernet switch, while each NXP board is connected to the others through an individual Gigabit TSN network interface. We define the two primary subnetworks for evaluation, shown in Fig. 3b: Zone A, consisting of one Zonal controller and one local ECU, and Zone B, similarly consisting of one Zonal controller and one local ECU. Both zones are connected to the CC, acting as the bridge between the user interface and the two zones. For consistent time synchronization, the RPi provides a reference clock to the CC, which it subsequently distributes to the rest of the network via gPTP.

B. *RA1: Network Topology & Security Performance Analysis*

Using the ZAT, we evaluated network performance within and across zones, measuring end-to-end latency, jitter, and packet loss across several potential network environments. We

measured the performance of local ECUs to their ZCs, ZC A to ZC B, and both ZCs to the CC across 50 experiments with senders transmitting 20,000 112-byte packets per second (PPS) over 60 seconds. We similarly performed another set of evaluations, sending the maximum allowable bandwidth of 1000 Mbps using UDP and non-hardware-accelerated TLS to assess the impact of security protocols when deployed within the IVN. We visualize the results from all experiments in Fig. 4. We show that latency within our network is stable and exhibits minimal jitter, even under heavy traffic loads. Further, while TLS had lower overall bandwidth than the UDP traffic load, performance was similarly consistent even without hardware-accelerated cryptography.

C. RA2: Dataset Generation

As with any networking technology, the availability of clean and repeatable datasets is crucial for understanding network behavior under specific conditions. Due to AE in ZA being an emerging technology under constant evolution, publicly available datasets remain limited. To promote further research that requires specific networking conditions, we evaluate the ZAT for configurable dataset generation. We define a tiered connection stream using UDP-based SOME/IP traffic, in which each local ECU sends data only to its ZC, the ZCs send data to each other and to the CC, and the CC sends data to the ZCs. The CC, ZCs, and local ECUs varied data transmission across 3 phases per dataset, 10 Mbps for 10 seconds, 250 Mbps for 20 seconds, and 100 Mbps for 15 seconds to simulate fluctuating network conditions. For completeness, all traffic on all interfaces were captured for each dataset, including PTP packets. We show the full size of each dataset in Table II. We compiled each dataset by capturing traffic on the ZCs and CC, and publicly released the *pcap* files in our repository [5].

D. RA3: Time-Sensitive Application Performance

A central focus of AE in ZA is the ability to control network timing to guarantee safety-critical and operational packets travel within specific bandwidth and latency requirements, according to their priority within the network. We demonstrate the capability of the ZAT to determine application performance across varying network priority classes as follows. In accordance with AE standards [25], we define seven priority classes ranging from 0, the highest priority, to 7, the lowest priority. While determining the network characteristics of a given priority is decided by OEMs, we simplify discussion by scaling available bandwidth to each priority class such that each class has a bandwidth of $10p$ Mb/s, where p denotes the priority class. We define an application in the CC that requests transmission of a 1 Gb collection of data to ZC B, analogous to ECU update images passed from a central compute connected to an OEM update service to a specific ECU. Further, we evaluate transmission times in realistic networking conditions by performing a Restbus simulation in which Audio Video Transport Protocol (AVTP) packets from an AE dataset are replayed with the priority defined in the dataset between the CC and ZC A [26]. Using the ZAT, we measured the time necessary to transmit the full 1 Gb file with all 8 priority classes across 100 runs, shown in Fig. 5

TABLE II: Total Size of ZAT Generated Datasets

	PCAP File Size (MB)			
Dataset	**CC**	**ZC A**	**ZC B**	**Total**
Full IVN	1009.55	1,225.30	1,271.67	3,506.52
CC and ZC	994.71	1104.33	1128.16	3,227.2
ZC Only	0	1062.98	995.92	2058.9

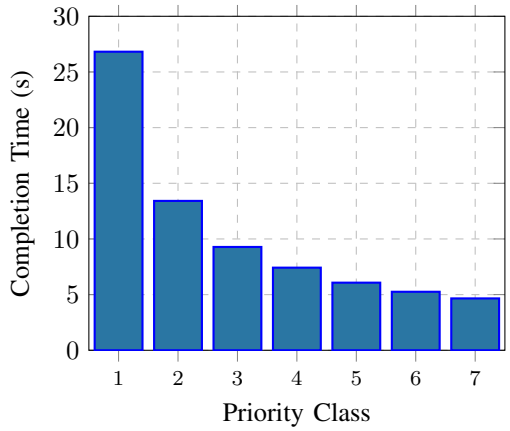


Fig. 5: Time to send 1 Gb with varying priority while the network is under Restbus simulation.

VI. Discussion

A. Comparison to Existing Work

We evaluate ZAT against three open-sourced automotive testbeds, PASTA [19], SANE [17], and VirtoBench [20], comparing their alignment to our four core principles.

Adaptable. PASTA and SANE focus on flexibility, with PASTA emphasizing hardware versatility and SANE focusing on software configurability. Vitrobench, however, relies on an FPGA board to interface between its ECUs and the host PC.

Generalizable. Despite PASTA's focus on security, its adaptability allows for a broader range of evaluation scenarios. Vitrobench, due to its FPGA communication board, is limited to IVN security. SANE can adopt new configurations of its simulated network, resulting in diverse evaluation capabilities.

Cyber-Physical. PASTA and Virtobench are deployed on hardware ECUs, enabling them to measure the properties of real physical systems. SANE, however, is constrained to software and relies on simulation accuracy for evaluation.

Focus on SDVs. SANE is specifically for E/E architectures and technologies, while PASTA and Virtobench primarily consider older CAN-based IVN architectures.

B. Future Work

Improved Functionality. As ZA progresses, so too will researchers' needs for tools that provide a comprehensive eval-

uation interface. We aim to increase ZAT's capability to fine-tune the variables in each experiment, including the traffic-generation method, individual-task control for each board during evaluation, and access to additional hardware properties and technologies to fully control experimental conditions.

User Experience Evaluation. To further evaluate the usability of ZAT, we plan to conduct qualitative user experience studies. By conducting semi-structured interviews with stakeholders using ZAT, combining a predetermined set of open-ended questions crafted to encourage discussion with the interviewer, we will gain a deeper understanding of user interactions with ZAT to recognize common themes and potential friction points. These studies will be crucial for strengthening the ZAT, and will dictate developmental focus.

VII. Conclusion

In this paper, we have presented ZAT, an open-source platform designed for research and development of ZAs IVNs. It offers a low-cost and flexible approach to testing topology and security protocols, generation of datasets, as well as studying software updates. Bridging the gap between software simulation and hardware prototyping, ZAT supports a wide range of hardware and provides a web interface for configuration and analysis. Its modular and adaptable system facilitates the integration of additional software, making it a valuable tool for advancing the next generation of IVNs.

Acknowledgments

This work is partially supported by a grant from DENSO International America, Inc., as well as partially based upon the work supported by the National Center for Transportation Cybersecurity and Resiliency (TraCR) (a U.S. Department of Transportation National University Transportation Center) headquartered at Clemson University, Clemson, South Carolina, USA. Any opinions, findings, conclusions, and recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of TraCR, and the U.S. Government assumes no liability for the contents or use thereof. This work is in part supported by Clemson University's Virtual Prototyping of Autonomy Enabled Ground Systems (VIPR-GS), under Cooperative Agreement W56HZV-21-2-0001 with the US Army DEVCOM Ground Vehicle Systems Center (GVSC). DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited. OPSEC #10573.

References

[1] O. Burkacky, F. Steiner, M. Kellner, J. Deichmann, and J. Werra, "Getting ready for next-generation e/e architecture with zonal compute," McKinsey & Company, Tech. Rep., 2023.

[2] 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.

[3] L. L. Bello, "The case for ethernet in automotive communications," *SIGBED Rev.*, vol. 8, no. 4, p. 7–15, Dec. 2011. [Online]. Available: https://doi.org/10.1145/2095256.2095257

[4] A. Frigerio, B. Vermeulen, and K. G. W. Goossens, "Automotive architecture topologies: Analysis for safety-critical autonomous vehicle applications," *IEEE Access*, vol. 9, pp. 62 837–62 846, 2021.

[5] TigerSec Laboratory, "Zonal architecture testbed github," https://github.com/JanPauldeVoor/Zonal-Architecture-Testbed, 2026.

[6] L. Mauser and S. Wagner, "Centralization potential of automotive e/e architectures," *Journal of Systems and Software*, vol. 219, p. 112220, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121224002644

[7] C. Park, C. Cui, and S. Park, "Analysis of e2e delay and wiring harness in in-vehicle network with zonal architecture," *Sensors*, vol. 24, no. 10, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/10/3248

[8] J. Lim, J. Lee, Y. S. Hong, and C. Kang, "A framework for designing zonal architectures for in-vehicle networks: Balancing communication load and wiring length," *IEEE Transactions on Vehicular Technology*, pp. 1–14, 2024.

[9] *Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical coding sublayer*, International Organization for Standardization Std. ISO 11 898-1:2024, 2024. [Online]. Available: https://www.iso.org/standard/86384.html

[10] *Road vehicles – Local Interconnect Network (LIN) – Part 1: General information and use case definition*, International Organization for Standardization Std. ISO 17 987-1:2016, 2016. [Online]. Available: https://www.iso.org/standard/61222.html

[11] *Road vehicles — FlexRay communications system – Part 1: General information and use case definition*, International Organization for Standardization Std. ISO 17 458-1:2013, 2013. [Online]. Available: https://www.iso.org/standard/59804.html

[12] *Road vehicles — Road vehicles — Media Oriented Systems Transport (MOST) – Part 1: General information and definitions*, International Organization for Standardization Std. ISO 21 806-1:2020, 2020. [Online]. Available: https://www.iso.org/standard/71833.html

[13] P. Hank, S. Müller, O. Vermesan, and J. Van Den Keybus, "Automotive ethernet: In-vehicle networking and smart mobility," in *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2013, pp. 1735–1739.

[14] T. Zhang, G. Wang, C. Xue, J. Wang, M. Nixon, and S. Han, "Time-sensitive networking (tsn) for industrial automation: Current advances and future directions," *ACM Comput. Surv.*, vol. 57, no. 2, Oct. 2024. [Online]. Available: https://doi.org/10.1145/3695248

[15] A. Kostrzewa, D. Stöhrmann, and R. Ernst, "Towards safety in automotive ethernet-based networks with dynamic workloads," in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, 2020, pp. 1–6.

[16] D. Piromalis and A. Kantaros, "Digital twins in the automotive industry: The road toward physical-digital convergence," *Applied System Innovation*, vol. 5, no. 4, p. 65, 2022.

[17] Y. Wu, H. Xie, and Y. Liao, "Sane: An integrated systematic framework for automotive in-vehicle network emulation," in *2024 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2024, pp. 164–171.

[18] G. Balan, P. Neninger, E. Ruiz Zúñiga, E. Serea, D.-D. Lucache, and A. Sălceanu, "A perspective on software-in-the-loop and hardware-in-the-loop within digital twin frameworks for automotive lighting systems," *Applied Sciences*, vol. 15, no. 15, p. 8445, 2025.

[19] T. Toyama, T. Yoshida, H. Oguma, and T. Matsumoto, "Pasta: Portable automotive security testbed with adaptability," *London, blackhat Europe*, 2018.

[20] A. K. T. Yeo, M. E. Garbelini, S. Chattopadhyay, and J. Zhou, "Vitrobench: manipulating in-vehicle networks and cots ecus on your bench: a comprehensive test platform for automotive cybersecurity research," *Vehicular Communications*, vol. 43, p. 100649, 2023.

[21] Natalie T, "Modular 10 inch server rack - reworked," https://www.printables.com/model/1090551-modular-10-inch-server-rack-reworked, 2025.

[22] Snowflake Inc., "Streamlit," https://www.streamlit.io, 2026.

[23] *SOME/IP Protocol Specification*, AUTOSAR Std. R22-11, 2022. [Online]. Available: https://www.autosar.org/fileadmin/standards/R22-11/FO/AUTOSAR_PRS_SOMEIPProtocol.pdf

[24] Christian Herzog, "someipy," https://github.com/chrizog/someipy, 2024.

[25] J. Migge, J. Villanueva, N. Navet, and M. Boyer, "Insights on the performance and configuration of avb and tsn in automotive ethernet networks," in *9th European Congress on Embedded Real Time Software and Systems (ERTS 2018)*, 2018.

[26] M. L. Han, B. I. Kwak, and H. K. Kim, "Tow-ids: Intrusion detection system based on three overlapped wavelets for automotive ethernet," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 411–422, 2023.