A Comparison of End-to-End Architectures for Connected Vehicles

Sidi Lu*, Nejib Ammar†, Akila Ganlath†, Haoxin Wang† and Weisong Shi*
*Department of Computer Science, Wayne State University, Detroit, MI 48202, USA
†Toyota Motor North America R&D, InfoTech Labs, Mountain View, CA 94043, USA
{lu.sidi, weisong}@wayne.edu, {nejib.ammar, akila.ganlath, haoxin.wang}@toyota.com

***Abstract*—With an ever-increasing number of vehicle services, connected vehicles (CVs) are becoming more software-dependent than ever, which is bringing a tremendous burden on the infrastructure. However, how to efficiently support such a tremendous amount of services is unknown. This paper takes the first step to examine different vehicle architectures. We first illustrate the evolution of automotive software and computing system. Next, we introduce the concept of software-defined vehicle, classify essential CV applications, and list their performance requirements such as communication mode, critical latency, data rate per vehicle, and communication range. Then, we choose over-the-air (OTA) update as our case study, followed by the introduction of four types of CV architectures. Then, we discuss advantages and drawbacks of each type of architecture. Finally, we conclude this article by presenting an edge-based architecture for CVs, named EdgeArC.**

I. Connected Vehicles: From Present to Future

Recently, with the wide deployment of communication mechanisms (*e.g.,* DSRC, V2X, 5G) and the vast improvements in computing technologies (*e.g.,* sensors, deep learning, and high-performance computing), there has been an acceleration in the research and development (R&D) efforts to bring the idea of connected vehicles (CVs) to fruition, and a plethora of intelligent applications are enabled in CVs such as remote real-time diagnostics, adaptive cruise control, and fuel efficiency optimization. These software-based services fuel the CV market. For example, the global CV market is projected to reach $225 billion by 2027 with a compound annual growth rate (CAGR) of 17% [1]. Besides, Automotive Edge Computing Consortium (AECC) predicts that every new vehicle will be connected by 2025, which will result in 50% of national vehicles (with a total number up to 400 million) on the road with connected features [2].

The Evolution of Automotive Software: The past decade has seen the transformation in consumer electronics from the early days of hardware differentiation to nowadays software differentiation, where consumers are more and more looking into new applications and services for improved user experience. The automotive industry has sensed the urgency of enhancing their software capabilities. Today's vehicles can already have up to 150 million lines of software code, distributed among more than 100 electronic control units (ECUs). The complexity and quantity of software for future CV will continue to increase exponentially, pushing the value of software in a future CV far exceeds that of the traditional vehicles made of mainly mechanical hardware with self-contained electronic devices such as ECUs.

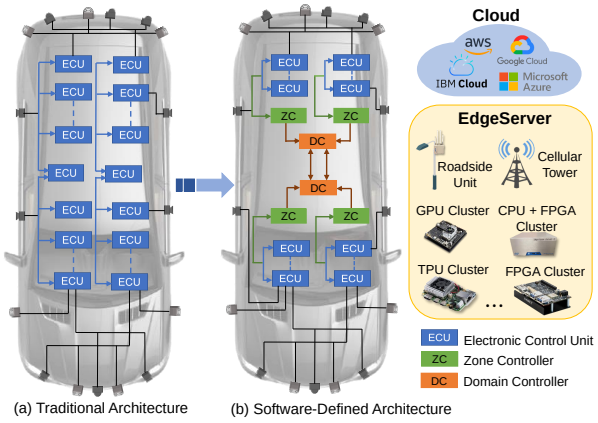

Fig. 1: The paradigm shifts in automotive computing system architecture.

The Evolution of Automotive Computing System: Moreover, the forthcoming evolution in CV also requires a major re-design of the underlying supporting automotive computing system architecture. Currently, most production vehicles are equipped with various small and fixed-function ECUs, which are usually produced by different suppliers. ECUs are connected through a Controller Area Network (CAN bus) so that ECUs can communicate with each other without a host computer. The various automotive functions are distributed across multiple ECUs located throughout a vehicle as shown in Fig. 1 (a). Given the limited resources in an ECU's micro-controller, it is almost impossible to deploy diverse computation-intensive applications with-

out incurring significant efforts in redesigning vehicle's computing system architecture.

Therefore, some leading automotive companies such as Tesla and APTIV [3] have adopt a type of sensible computing system architecture as shown in Fig. 1 (b). The idea behind this in-vehicle high performance computer architecture is to keep most existing ECU designs as is to not only better meet the software-based functional safety (FuSA) and real-time constraints, but also ease the transition from traditional vehicle architectures to the future of software-defined architecture. ECUs are partitioned into multiple functionally related zones controlled by their zone controllers (ZCs). ZCs are further interconnected with each other and partitioned into a set of domains controlled by domain controllers (DCs). Both ZCs and DCs are modern microprocessors capable of running embedded Linux operating systems. This system architecture not only greatly simplifies vehicles' system interconnection, but also makes the deployment of software functionalities to both ZCs and DCs possible. Through wireless connections, this architecture can be easily complemented by edge servers and clouds for collaboration.

The Need of End-to-End Architecture: Although the automotive software and computing system architecture are transforming vehicle capabilities, they also making automotive development trapped in a maze of complexity. This is mainly because that the development of automotive-software and computing system modules frequently occurs in isolation, and these changes are happening so rapidly that automotive OEMs and other industry stakeholders are now struggling to keep pace. In addition, integrating and upgrading the features that consumers increasingly expect for CVs also incurs enormous costs to automotive companies.

In this context, a suitable end-to-end architecture that integrates independent software elements into a comprehensive platform can improve functionality and decrease complexity, whereby the customer is central and the vehicle is integrated as optimally as possible into daily life and the needs of the customer. However, today's CV architectures are not optimized to handle the ever-increasing volumes of CV data, nor can they reliably and efficiently support time-sensitive CV services on a large scale. Therefore, choosing a suitable end-to-end CV architecture to support the efficient communication and collaboration between vehicle, edge server, and cloud is necessary to fill this gap.

The rest of the paper is organized as follows. We first elaborate on the big picture of the software-defined vehicle and four types of vehicle applications in Sec. II. Then, we summarize performance requirements of vehicle applications in Sec. III. Sec. IV describes our selected use case, *i.e.,* over-the-air (OTA) update, and Sec. V demonstrates four typical architectures. After comparing four architectures in Sec. VI, we present an edge-based architecture in Sec. VII and conclude the whole paper in Sec. VIII.

II. Software-Defined Vehicle & Applications

A. A Forthcoming Evolution in Automotive Industry

In recognition of the growing complexity and importance of software, as well as the forthcoming evolution in automotive computing system architecture, thought-leaders from automotive industry have begun working toward software-defined vehicles (SDV) [4]. The idea behind SDV is to emphasize a software-centric view of customers' experience with the vehicle, which can be personalized and updated with newer versions of software and newer services to be deployed throughout the life-cycle of the vehicle. And software updates and new service additions no long require customers visiting a dealership, but can be done through an over-the-air (OTA) update technology whenever as needed, which has already proven to improve consumer satisfaction [5].

However, current automotive hardware architectures are not well suited for the automotive industry's existing waterfall software development practices. The main reason is that the existing computing devices in CV are still constrained by limited computing power and inefficient communication protocols. To address such issues, the automotive industry is looking into the successful cloud-native technologies to help with such a SDV transformation. One recent example is the Scalable Open Architecture for Embedded Edge (SOAFEE) consortium led by Arm's [6].

The key concept of the cloud-native design principle is Service Oriented Architecture (SOA) where applications are decomposed as a set of self-contained functional services (sometimes also called **micro-services**) that can be deployed and orchestrated onto computing devices at different locations. These micro-services are typically deployed as either containers or virtual machines (VMs), depending on the nature of the application requirements. A properly designed cloud-native vehicle architecture, can consolidate a modern vehicle's various resources (such as system hardware, software, cloud services) into an integrated and centrally managed platform. Vehicle services can be run as different micro-services inside either a container or virtual machine, enabling scalable management and smooth migration of micro-services across different computing devices, in the vehicle, on the edge, or across clouds.

B. Application Landscape

Inspired by previous studies [7], we divide CV applications into four categories according to their themes as

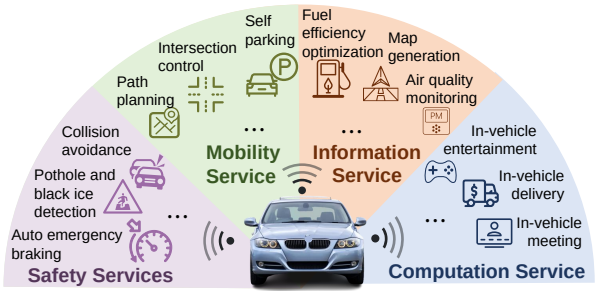

Fig. 2: Four types of CV applications.

shown in Fig. 2, related to safety, mobility, information, and computation:

i) **Safety application** mitigate the risk of hazards by issuing warnings to vehicle operators or to directly controlling a single vehicle's actuators. These applications usually address the critical need and/or call for hard real-time data processing and vehicle response.

ii) **Motion application** includes individual motion services and group motion services. Individual motion services intelligently provide (soft) real-time motion advisories for a single vehicle based on vehicle's real-time locations, destinations, and dynamic driving environment. Group motion application uses vehicle sensors and external data to influence or control the behavior of vehicles in aggregate.

iii) **Information application** aim to enhance users' comfort and ability to perform other tasks while driving or allow viewing vehicle parameters remotely. These applications usually tolerate transmission delays (soft real-time or non time-critical) and errors.

iv) **Computation application** reflects the efficient on-board computation capability of CVs, which can help computation-constrained connected devices/things to finished computation-intensive tasks even when CV is in charging or parking mode.

III. Application Requirements

The end-to-end CV architecture are derived by studying the needs of the CV applications and use cases [7]–[9]. In this section, we summarize application and their system performance requirements based on the requirements classification given in [8], [9].

Specifically, [9], [10] summarize the system performance requirements of vehicle applications, including the related communication mode, minimum transmission frequency, and critical latency, as shown in Table I. The coverage distance associated with these applications varies from 0 m to full communication range, depending on the use case.

In addition, considering the cooperation between vehicle, edge server, and cloud, [8], [11] list the performance requirement of different V2X applications, as shown in Table II, including end-to-end latency, reliability, data rate per vehicle (kb/s), and communication range. Here, V2P refers to vehicle-to-pedestrian. V2N stands for vehicle-to-network, where the vehicle connects to an entity within the network, *e.g.,* a back-end server or a traffic information system. The communication range is qualitatively defined as long for >500 meters, medium from 200 to 500 meters, and short for < 200 meters.

IV. Over-The-Air Update

A. Why OTA is important?

The number of automotive recalls and costs linked to software failures has risen exponentially in the U.S. — in 2020, a total of 83.2 million vehicles have undergone recalls, of which 75 million (90%) vehicles have been affected by software related problems. This reveals that more than a quarter of vehicles on the roads nationally have been recalled due to the software problem at least once a year (with an estimated 286.9 million registered vehicles in the U.S. [14]). Since the average cost of an auto recall over the last 10 years was about $500 per vehicle [15], it can be estimated that the average software-related recall cost nationally is around **$38 billion** in 2020. With the increasing number of vehicles, the software-related safety threat and cost is greater than ever.

All these software-related recalls could have been avoided if there were OTA software updates. OTA updates not only provide assistance for patching against security holes, but also support patching against automotive glitches in software that can cause malfunction in cars. OTA updates also provide enormous advantages in keeping in-vehicle software systems up-todate and maintaining consumer satisfaction.

B. Comparative Study of OTA Update Characteristics

ABI Research [16] predicts that the number of OTA supported vehicles will be around 203 million by 2022 with a CAGR of 58.15%. In the meanwhile, many established automotive companies are already on the path of providing OTA udpate services for CVs.

Halder *et al.* [5] presented a comparative study of OTA update characteristics for car companies, including Tesla, BMW and Mercedes Benz. In addition, they also present a comparative analysis of in-vehicle features that support OTA updates in automobile companies. Based on [5] and latest news [5], [12], [13], we summarize useful information in Table III and Table IV. It can be seen that Tesla came first with the ability to update various vehicle services, including maps, . Toyota currently offer OTA map update functionality and provide update for new audio multimedia platform, and it aims to develop

TABLE I: CV application and the related performance requirements [9], [10].

Use case	Communication mode	Minimum transmission frequency	Critical latency
Intersection collision warning	Periodic message broadcasting	10 Hz	100 ms
Lane change assistance	Co-operation awareness between vehicles	10 Hz	100 ms
Overtaking vehicle warning	Broadcast of overtaking state	10 Hz	100 ms
Head on collision warning	Broadcasting messages	10 Hz	100 ms
Co-operative forward collision warning	Co-operation awareness between vehicles associated to unicast	10 Hz	100 ms
Emergency vehicle waning	Periodic permanent message broadcasting	10 Hz	100 ms
Co-operative merging assistance	Co-operation awareness between vehicles associated to unicast	10 Hz	100 ms
Collision risk waning	Time limited periodic messages on event	10 Hz	100 ms
Regulatory contextual speed limit notification	Periodic, permanent broadcasting messages	1-10 Hz depending on technology	Not relevant
Green light optimal speed advisory	Periodic, permanent broadcasting messages	10 Hz	100 ms
Electronic toll collection	Intemet velicle and unicast full duplex session	1 Hz	200 ms
Co-operative adaptive cruise control	Cooperation awareness	2 Hz (some systems require 25 Hz)	100 ms
Co-operative vehicle highway automatic system (platoon)	Cooperation awareness	2 Hz	100 ms
Intersection collision wanning	Periodic message broadcasting	10 Hz	100 ms
Lane change assistance	Co-operation awareness between vehicles	10 Hz	100 ms
Overtaking vehicle warning	Broadcast of overtaking state	10 Hz	100 ms
Head on collision waming	Broadcasting messages	10 Hz	100 ms
Co-operative forward collision waming	Co-operation awareness between vehicles associated to unicast	10 Hz	100 ms
Emergency vehicle warning	Periodic permanent message broadcasting	10 Hz	100 ms
Co-operative merging assistance	Co-operation awareness between vehicles associated to unicast	10 Hz	100 ms
Collision risk warning	Time limited periodic messages on event	10 Hz	100 ms
Regulatory contextual speed limit notification	Periodic, permanent broadcasting of messages	1-10 Hz depending on technology	Not relevant
Green light optimal speed advisory	Periodic, permanent broadcasting of messages	10 Hz	100 ms
Electronic toll collection	Internet vehicle and unicast full duplex session	1 Hz	200 ms
Co-operative adaptive cruise control	Cooperation awareness	2 Hz (some systems require 25 Hz)	100 ms
Co-operative vehiclehighway automatic system (platoon)	Cooperation awareness	2 Hz	100 ms

TABLE II: Performance requirements of different V2X applications [8], [11].

Use Case Type	V2X Mode	End-to-End Latency	Reliability	Data Rate per Vehicle (kb/s)	Communication Range
Cooperative awareness	V2V/V2I	100 ms	90–95%	5–96	Short to medium
Cooperative sensing	V2V/V2I	3 ms	>95%	5–25,000	Short
Cooperative maneuvers	V2V/V2I	<3–100 ms	>99%	10–5,000	Short to medium
Vulnerable road user (VRU)	V2P	100 ms	95%	5–10	Short
Traffic efficiency	V2N/V2I	>1 s	<90%	10–2,000	Long
Teleoperated driving	V2N	5–20 ms	>99%	>25,000	Long

its own new operating system by 2025, named Arene, which allows updating new features over the air [13]. With these new promising advancement, future Toyota vehicles will be able to stand toe-to-toe with Tesla in terms of OTA software updates [12].

V. Four Types of Architectures

In this section, we list and discuss four types of architectures to support the collaboration between vehicles, edge servers, and clouds, including cnetralized architecture, decentralized architecture, publish/subscribe architecture, and broadcast architecture. A roadside unit (RSU) and cellular tower could both be treated as the edge server.

A. Centralized Architecture

Centralized architecture [17], also known as client-server architecture, refers to a type of architecture in which a set of client vehicles request and receive services from a centralized server/cloud. The centralized server/cloud waits for requests from client vehicles and then responds to those requests through a standardized interface, while the client vehicle does not need to know the specifications and details of the centralized server/-cloud. This computing model is particularly effective when the client vehicle and the centralized server/cloud each perform different routine services.

Mender [18] is an open source end-to-end OTA software update manager with a centralized architecture. Particularly, Fig. 3 shows an example of the centralized

TABLE III: Comparative study of OTA update characteristics [5], [12].

Car Companies	S/W Update Triggered by Whom	Update Notification	Driving Possibility during Update Process
Tesla	Tesla	Sent through an embedded ATT 3G data connection or a Wi-Fi router for Model S cars	No
BMW	BMW	Customer receives notification through Connected Drive system present in the car	No
Mercedes Benz	Costumer	Update notification sent through an embedded Verizon 3G data connection for C and S class cars	No
Audi	Information N/A	Update notification sent through an embedded T-Mobile 3G data connection for its A3, A4, A5, Q2, Q5 and Q7 cars	No
General Motors	Information N/A	GM vehicles are equipped with the OnStar 4G LTE-WiFi connectivity	No
Toyota	Toyota	Currently uses a standard LTE connection to update but plans to use Starlink satellite internet	No

TABLE IV: Comparative study of the in-vehicle features that support OTA updates [5], [12], [13].

Car Companies	Maps and navigation	Infotainment	Power Management Options	Location based Air Suspension Settings	Forward Collision Warning	Traffic aware Navigation	Blind Spot Warning	Auto Emergency Braking	Dashcam Feature
Tesla	✓	✓	✓	✓	✓	✓	✓	✓	✓
BMW	✓	✗	✗	✗	✗	✗	✗	✗	✗
Mercedes Benz	✗	✓	✗	✗	✗	✗	✗	✗	✗
Audi	✓	✗	✗	✗	✗	✗	✗	✗	✗
General Motors	✗	✓	✗	✗	✗	✗	✗	✗	✗
Toyota	✓	✓	✗	✗	✗	✗	✗	✗	✗

architecture. Taking the urban information sharing as an example, a client vehicle can run a related application to continuously collect urban information (such as road boundary information and lane detection results), while a centralized server/cloud can run another application program to perform assigned computation tasks for its corresponding terminals. Client vehicles can simultaneously pull or push information from/to the centralized server/cloud, and at the same time, client vehicles can perform other tasks.

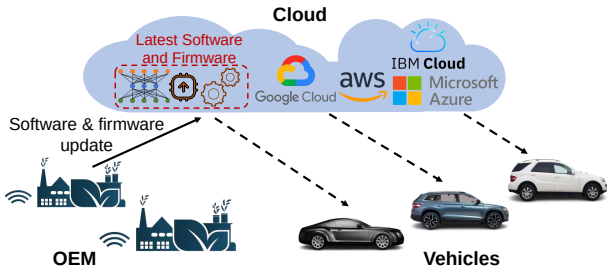

Fig. 3: An example of the centralized architecture.

B. Decentralized Architecture

Decentralized architecture, *i.e.,* peer-to-peer (P2P) architecture, is another solution for the communication and collaboration between vehicles, edge servers, and clouds, whereby two vehicles interact directly with each other without the participation of a third party (as shown in Fig. 4). A decentralized architecture distributes computing tasks among vehicles, with all vehicles contributing and consuming resources within the vehicular network without the need for a centralized server.

This architecture is suitable for the applications between vehicles, such as vehicle sharing and P2P vehicle rental. For example, HireGo [19] is a decentralized P2P private vehicle hire application, which is steered by the blockchain technology. Recently, P2P vehicle rental, *i.e.,* leasing idle vehicles to other people in need of driving, is becoming an increasingly popular way to earn additional income. The existing centralized applications suffer from high transaction fees and centralized ownership, which makes it easy to be targeted by hackers. In addition, as to P2P vehicle rental, another disadvantage of a centralized system is the monopoly of valuable user data. For example, a user may have built a good rental reputation over the years, and the user's review history is managed by a client server and may be revoked at any time. Decentralized architectures based on blockchain smart contracts show the potential to overcome these shortcomings, because it can return the control to end users and also eliminate the need of high-priced third parties.

As another example, the Mobility Open Blockchain Initiative (MOBI) [20] has initiated a global standard for a decentralized vehicle charging network, which in-

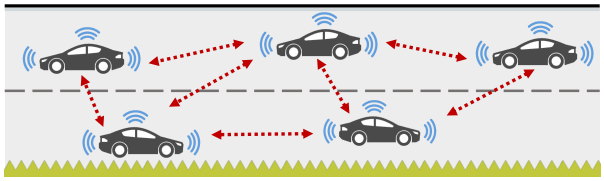

Fig. 4: An example of the decentralized architecture.

tegrates blockchain-related technologies. Led by General Motors and Honda, this standard targets to create a user-centric energy community for decentralized vehicles. Specifically, it includes two key use cases: (1) vehicle-to-grid integration, which allows electric vehicles to share electricity with the grid, and (2) P2P applications, which enables the power sharing between electric vehicles.

C. Publish/Subscribe Architecture

Publish/subscribe architecture is widely used in serverless application scenarios. In a publish/subscribe architecture (as shown in Fig. 5), the message sender is called publisher, and the message receiver is called subscriber. Publishers can send the message asynchronously to different subscribers without knowing who the recipient is. The publisher only needs to send the message to the message queue, and the subscriber can take out the messages they are interested in, *i.e.,* any message related to a topic will be received by all of the subscribers to the topic. Publish/subscribe architecture can be used to enable event-driven applications, with the objective of improving the application scalability and performance.

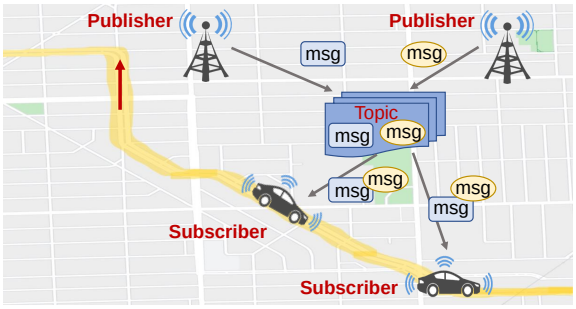

Fig. 5: An example of the publish/subscribe architecture.

More specifically, take the traffic warnings as an example [21], we can consider vehicles as mobile sensors collecting traffic information, accidents, etc. Vehicles report it collected information to the connected edge server based on vehicle-to-vehicle (V2V) communication. Edge server receive, store, and combine the information pushed from different vehicles in diverse locations, and then, generate and spread traffic warnings for reaching vehicles in the affected area.

In the publish/subscribe architecture, the vehicle only receives notifications related to a certain topic of the subscription. For example, a vehicle may be subscribed to receive only traffic warnings that may affect its route to its destination. The related notification continues to propagate within the affected zone to notify new reaching vehicles. Vehicles that receive such warnings will automatically recalculate suitable routes to avoid affected areas. It is worth noting here that this scenario can be easily expanded to support numerous applications, such as accident warnings, road works, free parking spots, fuel prices, advertisements, etc.

D. Broadcast Architecture

Due to the concentration of traffic on urban roads, vehicles often approach each other at intersections. In this case, the broadcast architecture (as shown in Fig. 6) is suitable for transmitting real-time traffic information at the intersection. For example, real-time safety-related traffic information can be broadcast to any vehicle in the affected area, whether or not they are interested in the information. Then, each vehicle that receives broadcast messages will rebroadcast them to other approaching vehicles, *i.e.,* perform successive broadcasts.

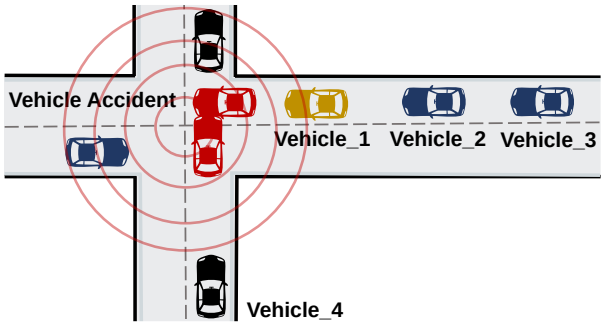

Fig. 6: An example of the broadcast architecture.

Figure 6 present an example of the broadcast architecture showing a vehicle collision accident near at an intersection. Suppose vehicle_1 generate the collision accident message, which will be broadcast to surrounding vehicles. If this message is only forwarded to one vehicle (vehicle_2) and cannot be delivered to the direction of vehicle_4, then vehicle_4 and the vehicles behind it will arrive at the accident point without being warned about the collision. This shows the significance of successive broadcasts for collision messages.

Since the collision messages is safety-related and contains urgent information, it should be broadcast in a real-time manner. However, the successive broadcasts will inevitably increase the network payload and therefore leads to excessive latency (known as the broadcast storm problem). Therefore, broadcasting urgent information to all vehicles still still has a long way to go.

VI. Architecture Comparison and Discussion

In this section, we compare four architectures that are discussed in Sec. V and summarize their related advantages, disadvantages, suitable vehicle applications, and the role of edges in terms of OTA update (as shown in Table V).

TABLE V: A summary of architecture comparison.

Architectures	Advantages	Drawbacks	Suitable Applications	The Role of Edges in OTA
Client-server Architecture	Consistency, efficiency, and affordability • Less IT management time • Fewer admins • Easy to track/collect data	• Safety and Resilience: A single software failure can influence various vehicles • Scalability: limited • Bandwidth: handicap	OTA update, (HD) map generation, fuel efficiency optimization	• Notify vehicles which software should be updated • Provide updates to vehicles • Provide its location info and its WiFi range to cloud • Enough memory capacity
Decentralized (P2P) Architecture	• Cybersecurity: prevent a DDoS attack • Efficient data management • Scalability	Only P2P architecture cannot satisfy all the real-world application requirements (vehicle is moving)	Vehicle sharing, vehicle rental, vehicle-to-grid integration, P2P applications	Vehicle itself is an edge computing platform • Communication • Computation
Publish/Subscribe Architecture	• Asynchronous: little risk of performance degradation • Scalability and flexibility: easy to add or removing subscribers	• Testing can be a challenge • An unexpected surge in message emission • Requires a well-defined policy	OTA, accident warnings, road works, free parking spots, fuel prices, advertisements	• Provide updates to various vehicles if needed • Have the memory capacity to hold both the old and new software image from cloud
Broadcast Architecture	• low cost of network deployment, • High communication efficiency, • Low server traffic load	• Inability to provide personalized services • Lack of pertinence • High network bandwidth	traffic live information in road intersections	Broadcast information to vehicles (push-based)

Centralized Architecture: The main advantages of a centralized architecture are its high efficiency, consistency, and affordability. Specifically, a central server controlling the entire OTA network can reduce the number of administrators and IT management time. Additionally, all data on a centralized network needs to go through the central server, making it easy to track, collect, and analyze data across the network.

Nevertheless, a centralized architecture does have its drawbacks. For example, a single point of software failure on the central server can be a risk factor for a group of vehicles: if the central server fails, individual client vehicles connected to it will not be able to process driver requests. In addition, they also offer limited scalability. Since a single central server is responsible for the processing of all applications, the way to scale up the network is to add more processing capability, storage, and bandwidth to the server, which may not be a cost-effective method in the long run. Moreover, lack of bandwidth may also become a hindrance. If there are fluctuating periods of OTA activities, a single server can quickly become a bottleneck, because the processing power of the central server may struggle to keep up with the sudden burst of concurrent requests.

Decentralized Architecture: Compared with centralized architecture, decentralized architecture has the following main advantages: (1) First, it allows vehicles to use blockchain technology to securely share information and conduct transactions without personal information. (2) Second, it is able to manage a large number of vehicles without exponentially increasing the cost of edge servers. (3) Besides, vehicle operators can use P2P software in OTA updates to more effectively reduce cybersecurity risks. In a centralized architecture, most cybersecurity attacks are possible. Because to paralyze vehicle fleets, all attackers do is focus their attacks on centralized edge servers/clouds, which puts businesses at higher risk. However, these attack methods are often not effective against decentralized networks. (4) Additionally, the decentralized architecture enables more efficient real-time data management, which is critical for vehicle services. This is because, in a decentralized architecture, real-time data transfer occurs between nodes and does not need to go through a server. (5) Moreover, with millions of CVs in the near future, one of the biggest advantages of a decentralized architecture is the ability to manage a large number of vehicles without multiplying the cost. For example, Streembit [22] claims that their P2P network design can grow to 1.28 billion devices in seconds, without additional cost, and they point out can address the scalability issues of connected vehicles.

However, only decentralized architecture cannot satisfy all the real-world application requirements. For example, it is challenging to support efficient and reliable OTA update, as vehicles are moving at different speed, leading to unstable connectivity.

Publish/Subscribe Architecture: A key advantage of the publish/subscribe architecture is that it supports asynchronous applications, so there is little risk of performance degradation because of processes getting caught in long-running data exchanges. Also, removing and addting subscribers to a defined topic is a matter of configuration and does not require complex programming. Therefore, a publish/subscribe architecture can provide great scalability and flexibility.

As for the downsides of a publish/subscribe architecture, first, testing can be a challenge. Because the interaction is asynchronous, testing is not a matter of making a request and then analyzing the result. Instead, a message must be sent into the architecture, and observe the behavior of processes to figure out when and how it processes the message. Also, when the number of processes for a given topic keeps increasing over time, the testing process can become more difficult to manage. Besides, unexpected surges of message transmission may lead to network congestion. Furthermore, it requires an

explicit message format and message transmission policy; otherwise, the whole process may become confusing and error-prone.

Broadcast Architecture: In a broadcast architecture, a sender push data, software, or firmware to a predefined vehicle network without a specific target, and vehicles are forced to accept messages from the sender, so it has advantages and disadvantages. In terms of advantages, first, network equipment and maintenance are simple, and network deployment costs are low.Second, the information can be transmitted to all vehicles on a network at once, leading to an efficient data communication. In addition, the traffic load on the server is low since the server does not need to send data to each vehicle client individually.

As to the disadvantages, first, it cannot provide personalized services in a timely manner according to the specific requirements of vehicle services. Besides, it lacks pertinence: vehicles are forced to receive data whether or not they actually need to receive data. In addition, although the server traffic load is low, it broadcast all information to all vehicles, which requires the high bandwidth of the network.

VII. EdgeArC: Edge-based Architecture for Connected Vehicles

The upgrading of automotive end-to-end architecture is mainly reflected in three aspects: hardware architecture, software architecture, and communication architecture. As described in Fig. 1 (b), hardware architecture is changing from distributed to domain control/centralized development. Software architecture is developing from high coupling of software and hardware to hierarchical decoupling (containerized), as discussed in Sec. II. In terms of communication architectures supporting the collaboration between vehicles, edge devices, and cloud, each type of the aforementioned architecture (*e.g.,* centralized architecture, decentralized architecture, publish/-subscribe architecture, and broadcast architecture) has its suitable application scenario (as shown in Table V).

In this section, we propose and illustrate an edge-based architecture for CVs, called EdgeArC (as depicted in Fig. 7), which enables the vehicle to collaborate with surrounding vehicles, offloading workloads to edge servers and connected devices/things (denoted as XEdge) and remote cloud.

Specifically, Fig. 7 presents the three-tier paradigm of EdgeArC. In EdgeArC, vehicles are software-defined where the consolidation of functional blocks within the vehicle can be enabled, and the performance, safety, and comfort of CVs can be continuously improved through OTA updates. In vehicle, a hypervisor is deployed on top of the vehicle computing unit, which can provide well

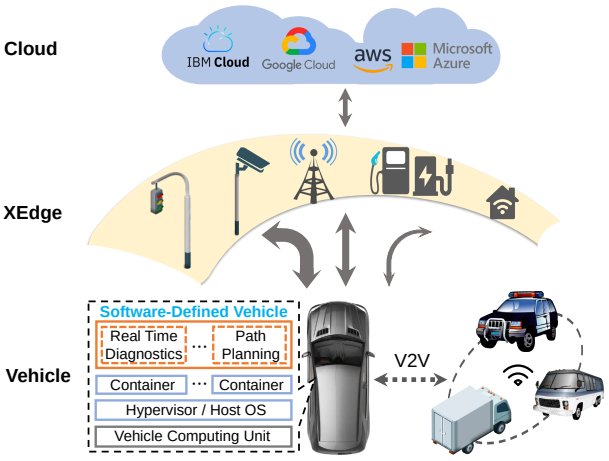

Fig. 7: An illustration of EdgeArC.

isolated virtualized environments for diverse software. Each ECU codebase can run almost unmodified in its own virtual machine and the resulting server platform may run a mix of real-time operating system (RTOS) for safety-critical and hard real-time applications (such as collision avoidance and real-time diagnostics) and rich operating system (OS) for the soft real-time services (*e.g.,* map generation) and non time-critical services (*e.g.,* infotainment).

Besides, thanks to the rapid development of the Vehicle-to-Vehicle (V2V) technologies, vehicle services could be migrated to collaborative vehicles that have idle computation resources. Here, the decentralized architecture is an appropriate communication architecture to support the data transmission between vehicle fleets. More importantly, vehicle data and services could also be offloaded to XEdge (*e.g.,* roadside units, cellular towers, gas stations, charging piles, and the computation devices that are installed in a connected home) and receive corresponding analysis results, where the publish/subscribe and broadcast architecture are suitable communication architectures. In the XEdge layer of Fig. 7, an arc that decreases in size from left to right indicates a decreasing number of these XEdge devices that may be encountered by the host vehicle. In the meanwhile, XEdge devices may also upload data or computation-intensive services to cloud for further analysis and then receive related results, and the centralized architecture is suggested for XEdge devices to send and receive data through cellular or satellite communication, etc.

VIII. Conclusion

In this article, to determine the appropriate end-to-end CV architecture designing strategies, we first illustrate the evolution of automotive software and computing system. Next, we introduce the concept of software-defined vehicle, classify essential CV applications, and list their

performance requirements such as communication mode, critical latency, data rate per vehicle, and communication range. Then, we choose over-the-air (OTA) update as our case study, followed by the introduction of four types of CV architectures. Then, we discuss advantages and drawbacks of each type of architecture. Finally, we conclude this article by presenting an edge-based architecture for CVs, called EdgeArC.

REFERENCES

[1] A. M. Research, "Connected Car Market Size, Share, Growth & Trends Analysis Report by Technology, Connectivity Solution, Service, End-Use, And Segment Forecasts, 2020-2027," https://reports.valuates.com/market-reports/ALLI-Manu-3Z1/connected-car, 2019.

[2] AECC, "Distributed computing in an aecc system (online)," ht tps://aecc.org/resources/publications/, August 2021.

[3] L. Bauer, "Smart vehicle architecture: A sustainable approach to building the next generation of vehicles," in *APTIV White Paper*, June 2020.

[4] "Software-defined vehicles," https://www.arm.com/solutions/automotive/software-defined-vehicles, accessed: 2021-12-17.

[5] S. Halder, A. Ghosal, and M. Conti, "Secure over-the-air software updates in connected vehicles: A survey," *Computer Networks*, vol. 178, p. 107343, 2020.

[6] "Soafee: Scalable open architecture for embedded edge," https://soafee.io, accessed: 2021-12-17.

[7] N. Williams and M. Barth, "A qualitative analysis of vehicle positioning requirements for connected vehicle applications," *IEEE Intelligent Transportation Systems Magazine*, vol. 13, no. 1, pp. 225–242, 2020.

[8] M. Boban, A. Kousaridas, K. Manolakis, J. Eichinger, and W. Xu, "Connected roads of the future: Use cases, requirements, and design considerations for vehicle-to-everything communications," *IEEE vehicular technology magazine*, vol. 13, no. 3, pp. 110–123, 2018.

[9] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil, "Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions," *IEEE communications surveys & tutorials*, vol. 13, no. 4, pp. 584–616, 2011.

[10] T. ETSI, "Intelligent transport systems (its); vehicular communications; basic set of applications; definitions," *Tech. Rep. ETSI TR 102 6382009*, 2009.

[11] ——, "Intelligent transport systems; vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service, std," *ETSI EN Std*, vol. 302, pp. 637–2, 2014.

[12] TechWalls, "Toyota's new move to take on Tesla in advanced auto software," https://www.techwalls.com/toyota-new-move-take-on-tesla-advanced-auto-software/, October 2021.

[13] Aroged, "Toyota to release its own Arene operating system for cars by 2025," https://www.aroged.com/2022/01/04/toyota-to-release-its-own-arene-operating-system-for-cars-by-2025/, January 2022.

[14] H. Laguna, "Recap of 2020 recalls reveals impact of pandemic on compliance and the continuing threat (online)," https://www.recallmasters.com/sor/, May 2020.

[15] C. Isidore and P. Valdes-dapena, "Hyundai's recals 82,000 electric cars is one of the most expensive in history (online)," https://www.kktv.com/2021/02/26/hyundais-recals-82000-electric-cars-is-one-of-the-most-expensive-in-history/, Feb 2021.

[16] ABIresearch, "ABI Research Anticipates Accelerated Adoption of Automotive Software Over-the-Air Updates with Nearly 180 Million New SOTA-Enabled Cars Shipping Between 2016 and 2022," https://www.abiresearch.com/press/abi-research-anticipates-accelerated-adoption-auto/, 2016.

[17] C. Olaverri-Monreal, "Autonomous vehicles and smart mobility related technologies," *Infocommunications Journal*, vol. 8, no. 2, pp. 17–24, 2016.

[18] Lakshan, "Importance of Secure and Robust OTA Updates for Embedded Linux Systems," https://www.seeedstudio.com/blog/2021/11/26/importance-of-secure-and-robust-ota-updates-for-embedded-linux-systems/, 2021.

[19] A. Scott-Briggs, "HireGo – Peer to Peer Car Hire and Car Sharing Platform," https://techbullion.com/hirego-peer-to-peer-car-hire-car-sharing-platform/, 2018.

[20] T. Gresham, "Honda, GM-led group develops global blockchain standard for EV grid integration," https://www.utilitydive.com/news/honda-gm-led-group-develops-global-blockchain-standard-for-ev-grid-integra/587263/, 2020.

[21] I. Leontiadis, "Publish/subscribe notification middleware for vehicular networks," in *Proceedings of the 4th on Middleware doctoral symposium*, 2007, pp. 1–6.

[22] Streembit, "Autonomous vehicle management," 2020. [Online]. Available: https://zovolt.com/wp-content/uploads/zovolt_autonomous_vehicles.pdf