Received 5 August 2022, accepted 11 August 2022, date of publication 17 August 2022, date of current version 6 September 2022.
Digital Object Identifier 10.1109/ACCESS.2022.3199356

RESEARCH ARTICLE

Elastic Gateway Functional Safety Architecture and Deployment: A Case Study

ABDOUL AZIZ KANE[1], ANGELA GONZALEZ MARIÑO[1], FRANCESC FONS[1], (Senior Member, IEEE), SANDRO NUEESCH[1], PIOTR SERWA[2], AND MICHAEL SCHOETZ[2]
[1]Munich Research Center, Huawei Technologies Düsseldorf GmbH, 80992 Munich, Germany
[2]Exida.com GmbH, 83730 Fischbachau, Germany
Corresponding authors: Abdoul Aziz Kane (abdoul.aziz.kane@huawei.com) and Angela Gonzalez Mariño (angela.gonzalez.marino@huawei.com)

ABSTRACT The automotive industry has started its transformation towards **Software-Defined Vehicles**. This transformation is driven by the rise of the number of vehicle features, the high complexity of these features and their constraining availability requirements which affect all the players (Original Equipment Manufacturers, Tier1 and Tier2) of the sector. In the context of this transformation, our target, from **functional safety** point of view, is to, inter alia, provide an easy-to-use and safety-compliant execution and development flow and simplify the development and argumentation for safety by providing a) a pre-certified execution environment with **safety design patterns** and best-in-class safety measures and b) processes and tooling to minimize the system integrator's effort. Therefore, in this work we propose a top-down approach where we first define a **New Generation In-Vehicle Network, NGIVN,** capable of fulfilling the performance (e.g. high bandwidth, low end-2-end delay), safety-related availability (e.g. Autonomous Driving / Advanced Driver-Assistance Systems (**AD/ADAS) up to SAE level 5**) and **safety requirements** of modern vehicles. Also, we illustrate the advantages of this approach by deriving the functional and safety attributes of an **Automotive Gateway SoC**, named Elastic Gateway and destined to be part of the NGIVN. Through the deployment of the Elastic Gateway functional safety concept we demonstrate the **flexibility** provided by our approach with regards to the design of elements of the NGIVN.

INDEX TERMS AD/ADAS, automotive gateway SoC, fail-operational, functional safety, requirements, software defined networking, software defined vehicles.

I. INTRODUCTION

The new trends in the automotive industry (connected vehicles, autonomous driving (AD)) are making vehicles evolve faster than ever from a technology perspective [1], [2]. Traditionally, vehicles were fully mechanical products, where electronics were introduced as additions for specific functionalities. Basically, for each new sensor or actuator with electronic control integrated in the car (lights, windows, infotainment system, etc.) an electronic control unit (ECU) would be introduced [3], [4]. While being a simple approach at the beginning, with the introduction of many more sensors and actuators, this became difficult to handle [5], [6].

The associate editor coordinating the review of this manuscript and approving it for publication was Liang-Bi Chen.

In today's vehicles, the In-Vehicle Network (IVN) is organized in a logical distribution, where sensors and actuators are organized in functional groups or domains, i.e. the body/comfort domain, the cockpit/infotainment domain, the powertrain/chassis domain, etc. Each of these domains is managed by a domain controller that handles the exchange of information between the sensors and actuators of the domain, and also to other domain controllers through a central gateway. This is known as domain-based architecture [7]. Each domain has sensors and actuators all across the vehicle (e.g. lights in front and rear), which means that cables need to be laid out from front to back of the vehicle for each domain.

The vehicle software (SW) is also organized following this logical distribution, mapping the software functionalities to the related hardware that is present in the vehicle [8].

However, we are seeing that this is changing due to the increase in functionalities and their complexity, which usually involve different sensors and actuators from mixed domains, requiring a paradigm shift. The trend we are seeing today is to move towards Software-Defined Vehicles, where everything can be easily configured by the Original Equipment Manufacturer (OEM) from software perspective and without strict hardware (HW) dependencies [9], [10]. At the same time, this new concept is impacting on the Electric/Electronic (E/E) architecture transitioning from a domain-based approach (i.e., grouping of functions according to logical criteria) to a zonal approach (i.e., grouping of those functions according to physical/spatial criteria).

In the zonal approach, domain controllers are replaced now by new ones named zonal controllers that embed cross-domain functions in order to give service to all the functions allocated in a given physical zone of the vehicle [8], [11]. This means that the functions previously performed by each domain controller are now remapped across the zonal controllers interconnected through the IVN. Also, each zonal controller hosts now functions corresponding to different domains.

With this, we see that the current domain-based IVN imposes some limitations in terms of configuration capabilities and the deployment of new functionalities from a software perspective. Furthermore, it brings a very high wire harness cost and limits the scalability of the network [12].

Therefore, a new generation of In-Vehicle Networks (IVNs), characterized by a multitude of data sources, including Long-Range Radar, LIDAR, Cameras, Short-/Medium Range Radar, and Ultrasound, easily amounting to more than 100 sensors per car, is required for Software-Defined Vehicles [13].

The New Generation IVN, or NGIVN, shall interconnect sensors, in-vehicle processing units, actuators, and external communication peers, i.e. the cloud and other vehicles. Data must flow between the sensors and the other nodes at least partly with real-time guarantees. Also, the NGIVN shall support different types of sensors using different communication protocols, generating data with different rates and characteristics, offering sensor data output at different fidelity and raw sensor data as well as preprocessing to different degrees [14], [15].

In addition to these functional features, when considering new use cases such as autonomous driving, safety and reliability aspects of the network become of utmost importance [16], [17]. Today, the safety goals of item functions of the vehicle are typically fail-safe, i.e. when a failure is detected, the function enters into its safe state (e.g. low beams are turned on as countermeasure to some failure, covering thus the worst case scenario) [18], [16]. Other safety-goals are not fail-safe, but the availability is only needed at actuation side (e.g. Electric Power Assisted Steering performs a ramp-down of steering assist in case of loss of communication with torque sensor) [19]. However, going towards autonomous driving this is not enough. The new requirement for Autonomous Driving / Advanced Driver-Assistance Systems (AD/ADAS)) is that the vehicle must be fault-tolerant, i.e. it must continue operation even in the presence of failure [20], [21]. Thus, the NGIVN shall also migrate from being fail-safe to being fail-operational. It shall detect faults in any of its components, recover from it by either correcting it or by redirecting the traffic, and notify the rest of the vehicle if necessary. All of which shall be done without reducing the availability of the vehicle's features and without endangering the safety of the traffic participants [18], [22].

Ensuring compliance with safety regulations and standards such as ISO26262 [18], or SOTIF [23] is an arduous task, which usually needs to be re-done when changes are made to a subsystem of the vehicle. That is, today, the way to prove safety compliance is to follow a bottom-up approach, certifying first each component, and later its interactions with the rest of the components of the network [24]. On the other side, the vehicle development process is top-down. An OEM defines a top-level item architecture and successively chooses subsystems, components and HW elements. Given the increase in complexity of the NGIVN compared to traditional IVNs, this bottom-up approach is non-scalable and limits the flexibility to deploy new safety mechanisms in the vehicle [25].

In this context, our work is focused on providing an easy-to-use and safety-compliant design and development flow to simplify the development and argumentation for safety in modern vehicles. We propose to change the current bottom-up approach for a top-down approach, able to provide a pre-certified execution environment with safety design patterns and best-in-class safety measures that allow for minimizing the system integrator's effort. The main contributions of this work are listed below:

- **C1. Top-down approach to develop different safety concepts for NGIVN devices**: Being the main contribution of this work, we define the strategy and development flow that permits to go from top to bottom of the NGIVN ensuring that safety aspects are met by design at all system levels. We also define a methodology that allows for simplifying the integration of new safety mechanisms into the network with minimum engineering effort and guaranteeing safety compliance.
- **C2. Functional Safety concept NGIVN**: We define the NGIVN, identify the components of which it is composed and the requirements to be met by each of them from safety perspective. We also identify and define the state-of-the-art and beyond state-of-the-art technologies or safety mechanisms that constitute the backbone of the safety concept of the different elements of the NGIVN. With this, we are able to define the functional safety concept of the NGIVN and, from this, to derive the functional requirements of the different elements that compose the NGIVN.
- **C3. Demonstration of the flexibility and scalability of the top-down approach through a case study**: We

show how the top-down approach allows to develop new safety concepts for automotive devices in a flexible and scalable manner. For this, we present a case study focusing on the development of the functional safety concept for a novel Automotive Gatewaying System on Chip (SoC) specifically designed to meet the requirements of NGIVNs. We also provide background information on this SoC referred as "elastic Gateway (eGW)". We define the failure modes of the NGIVN and the safety mechanisms that allow to overcome them. Finally, we deploy the safety mechanisms in the eGW ensuring that the safety requirements are met, and therefore guaranteeing that the safety goals of the NGIVN are met as well.

The remainder of this paper is structured as follows: We start in section II by providing a comprehensive overview of the state of the art regarding safety mechanisms deployment in automotive networking devices, including Automotive Gateway Controllers. We show that the current methods and implementation are neither flexible enough nor optimized to meet the stringent requirements of the Autonomous Driving trend. In Section III we discuss new trends in automotive FuSa solutions and the identified gap in the state of the art as the motivation behind this work. In section IV we present the NGIVN, analyzing first the components integrated in it and their requirements, and deriving then the functional safety concept for the NGIVN. In section V we introduce the software defined strategy proposed in our solution. In section VI we present a case study of application of this top-down approach to develop the safety concept of an automotive network SoC. For this, we present our new generation automotive gateway controller and its associated safety concept which is derived from the NGIVN safety concept. In Section VII we detail the steps required in our top-down approach for the deployment of this safety concept. Then, in Section VIII we elaborate on the flexibility of our approach and its advantages with regard to defining the safety concept of various gateway devices simultaneously. Finally, we conclude the work in Section IX.

II. STATE OF THE ART ON AUTOMOTIVE FUNCTIONAL SAFETY

A. FUNCTIONAL SAFETY PROCESS IN THE CONTEXT OF AUTOMOTIVE INDUSTRY TODAY

The automotive industry is a huge and complex environment where many different actors are involved. From Original Equipment Manufacturers (OEMs) on the top of the supply chain, to Tier-1 and Tier-2 players which provide vehicle subsystems, or components of these vehicle subsystems. In order to guarantee a successful product development, the whole ecosystem needs to be aligned, and the interfaces between components and subsystems need to be clearly defined. For this purpose, standardization plays a key role in driving direction for all the industry players. Regarding safety, the current standard that defines the process and requirements needed across the whole product lifecycle (from a safety perspective) is ISO 26262. Under this framework, Tier-2 companies like semiconductor vendors design the gateway chipset that enables internal communication within the vehicle. Each of them equips the different products with functional and safety features aligned with the needs of their customers, the Tier 1 and OEMs. However, it is important to highlight here the fact that all these different devices developed by automotive semiconductor vendors (e.g. Infineon, NXP, ST, Renesas) are usually delivered by all those Tier-2 as a Safety Element out of Context (SEooC) solution, i.e. they are delivered to the Tier-1/OEM responsible for its integration in its particular business case solution as a component equipped with safety mechanisms without considering the context of their application in a particular vehicle. Therefore, the responsibility in terms of reaching a safety-compliant solution relies on the automotive player who integrates such device in the context where it is targeted. The Tier-1/OEM will configure and program all the safety features provided in the device in order to reach the safety compliance with the Automotive Safety Integrity Level (ASIL) required by its solution.

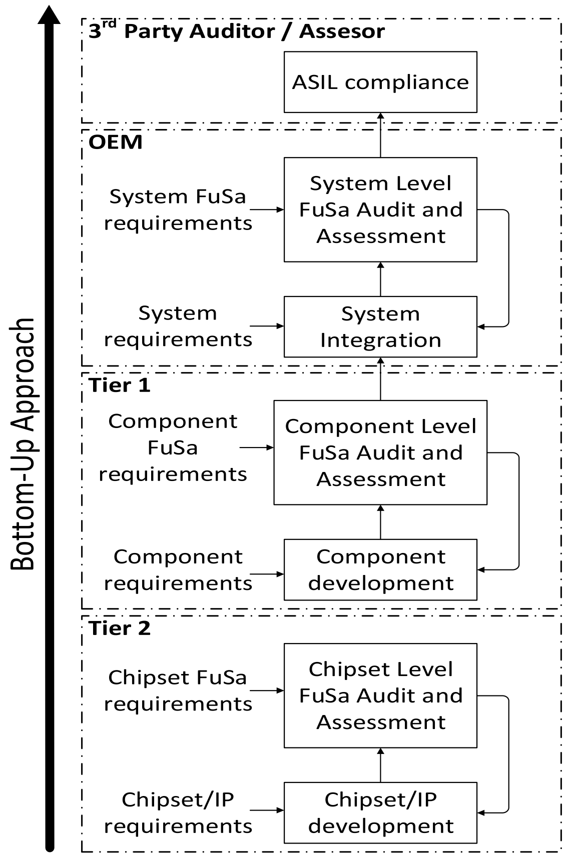
3rd Party Auditor / Assesor
ASIL compliance
OEM
System FuSa requirements
System Level FuSa Audit and Assessment
System requirements
System Integration
Tier 1
Component FuSa requirements
Component Level FuSa Audit and Assessment
Component requirements
Component development
Tier 2
Chipset FuSa requirements
Chipset Level FuSa Audit and Assessment
Chipset/IP requirements
Chipset/IP development
Bottom-Up Approach

FIGURE 1. State of the art bottom-up approach for FuSa integration within system development flow.

As we can see, the typical approach followed towards reaching the required FuSa compliance is a bottom-up approach. This starts at the chipset level, where Tier 2 players try to provide the best safety mechanisms according to their expertise, by including isolated safety capabilities which then need to be validated at both component and system level by

Tier 1 and OEM, respectively. This approach is depicted in Fig. 1. In the figure we identify three levels of abstraction for the safety analysis of the IVN, typically corresponding to different actors in each of the levels.

However, these levels can be merged through vertical integration, as depicted in Fig. 2, simplifying the interfaces between the different stages. In case of one actor taking charge of the full system solution, this could be better optimized and iterations at the three levels could be more agile. Nevertheless, in this second figure, we can still see how the three safety analysis levels are present: Chipset or SoC level, GW or ECU at component level and system or IVN level. Even when done by the same actor, nowadays the approach is the same, going bottom-up, keeping the three safety analysis levels isolated, and usually taken charge by completely different teams.

B. SAFETY MECHANISMS EMBEDDED IN IVN, ZONAL GW CONTROLLER AND NETWORK SoC

In order to build a fail-safe, fault-tolerant or fail-operational system, it is extremely important to design robust and reliable system architectures able to withstand and accomplish such target. That is, adopting the ‘‘safety by design’’ or ‘‘design for safety’’ principle since the very beginning of the product conceptualization is the right and only way to efficiently architect and develop safety compliant solutions.

In this section, we collect all those technical aspects which are part of the state-of-the-art today and break them down into specific strategies and tactics that derive in safety mechanisms portable to the design of safe in-vehicle networks, embedded in electronic control units (ECU) and taking into account the diverse types of failures that can occur in a network – from some transport delay in safety critical frames to link failures in an ECU or even an entire node (ECU) failure, among others. Next, we classify the networking safety mechanisms according to different technical criteria and elaborate about them.

1) SYSTEM REDUNDANCY AND DIVERSITY

Redundancy, understood as a second path available in case that a primary path is defective, is one of the strategies for reliability in cyber physical systems. This second path, moreover, is convenient to be implemented in a diverse way compared to the first path in order to not be affected by the same type of fault than the primary path.

Like this, diversity refers to implementing the redundant solution in a different way or with a different technology than the primary solution to avoid, by design, a single and common source of failure for both primary and redundant solution. Some strategies that exploit the redundancy and diversity principles applicable into in-vehicle networks are described next.

a: FRAME REPLICATION AND ELIMINATION FOR RELIABILITY (FRER)

Given a network topology based on different nodes interconnected, the ‘‘P802.1CB – Frame Replication and Elimination for Reliability’’ standard [26] specifies procedures, managed objects and protocols for bridges and end stations that provide:

- Identification and replication of frames, for redundant transmission.
- Identification of duplicate frames.
- Elimination of duplicate frames.

To achieve that, it is necessary to create and eliminate duplicate frames, and this can be done in both end stations and relay nodes (e.g., bridges or routers).

b: NETWORK CODING

Network coding was originally proposed for improving throughput of multicast communications [27]. The key idea is to allow intermediate nodes not only perform forwarding operations but also to create new packets by combining two or more packets from the incoming flows. Throughput enhancement is just one of the possible applications of network coding. Robustness, security and storage are other examples. In wireless and cellular networks, network coding is typically applied to improve throughput and robustness in scenarios with relays, when base station transmissions are relayed at least once before they can reach end users.

c: DUAL MODULAR REDUNDANCY (DMR)

Dual Modular Redundancy (DMR) is a technique that uses a functionally identical secondary system to back up the primary system. It consists of logic duplication and a comparator. The secondary system is additional like a spare solution and does not monitor the main system. As example, lock step execution is an extended use case of DMR [28]. Lockstep systems are fault-tolerant systems that run the same set of operations at the same time, in parallel. The redundancy (i.e., duplication) allows error detection: the output from lockstep operations can be compared to determine if there has been a fault.

d: TRIPLE MODULAR REDUNDANCY (TMR)

Triple modular redundancy (TMR) is an established technique for improving hardware fault tolerance. It is a process that uses a form of triple redundancy to control faults.

A typical example of use is in single event upset (SEU) mitigation. If a single indication of a critical operation resulted in an error, the entire mission would result in a disaster. However, a TMR design has three identical instances of hardware with a voting hardware at the output. If some SEU affects one of the hardware instances, the voting logic notes the majority output and this operation masks malfunctioning hardware by withstanding to it, therefore avoiding a system shutdown due to a single error and providing both safety and availability to critical missions.

e: PARTITIONING AND DECOMPOSITION

Partitioning (typically software-based) can be used for fault containment to avoid cascading failures between

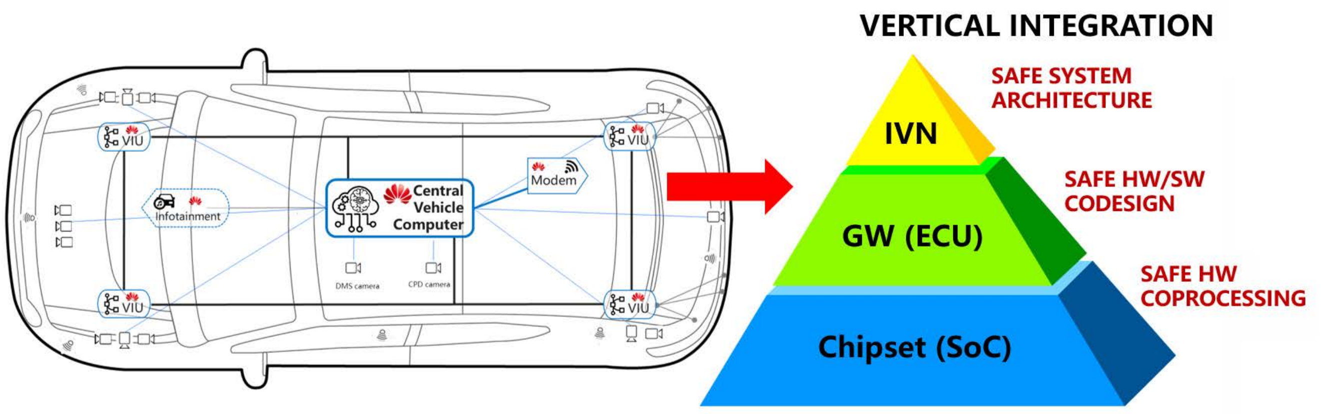

FIGURE 2. In-vehicle network functional safety analysis decomposed in three levels of abstraction: vehicle/system level (IVN E/E architecture), ECU/component level (HW/SW codesign) and chipset/SoC level. (HW network accelerators and peripherals design.)

software elements. It is a technique for providing isolation between software components to contain and/or isolate faults. This method can be implemented in hardware, software and a combination of hardware and software [29], [30].

Decomposition is another strategy that can contribute to effective deployment of safety solutions. Applications and network communications are assigned Automotive Safety Integrity Levels (ASILs) based on the ISO 26262 standard for functional safety in automotive systems. ISO 26262 outlines, for each ASIL, requirements on coverage of random hardware errors and systematic errors. Specific to systematic errors, the ISO 26262 standard defines ASIL decomposition as a vehicle to decompose functions into independent components, each with a lower safety requirement than that of the original function. Since the cost of a component is increasing with its ASIL, decomposition can lower the total cost while still meeting the original safety requirements [31].

f: NETWORK TOPOLOGY
Different topologies can be exploited: ring, double ring, mesh, star, etc. Each of them provides different advantages/disadvantages regarding safety in general, and redundancy in particular. For example, a ring topology provides at least two alternate paths for each node by construction, while a star topology provides only one.

g: T-SHAPE INTERFACING
One strategy to achieve redundancy at node level is to provide every node with at least three ports to different nodes in the network in order to guarantee redundant paths.

2) TIME SYNCHRONIZATION
In time deterministic distributed systems, it is necessary that all the processing nodes that compose the system share a common notion of time or global time. To meet this goal, a synchronization protocol which aligns the time of each node to a global reference clock is required.

a: GENERALIZED PRECISION TIME PROTOCOL
The generalized Precision Time Protocol or gPTP standard (IEEE 802.1AS [32]) was developed by the AVB Task Group and released by the IEEE in 2011. The synchronization of networked devices afforded by gPTP provides a common time base needed for professional-quality clocking of audio and video as well as time-sensitive streams, giving place to distributed timing: gPTP nodes periodically exchange packets with embedded timing information with the end result that all gPTP nodes are synchronized to the gPTP grandmaster. The gPTP grandmaster is a single device that is either automatically selected with gPTP's Best Master Clock Algorithm or manually pre-assigned when so desired. Either way, the end result is the same – precise synchronization (+/- 500ns) of all gPTP nodes. This level of synchronization is the underpinning that enables protocols built on gPTP to subsequently synchronize media clocks of talkers and listeners with uncompromised professional quality.

b: IEEE802.1AS
IEEE 802.1AS-2020 is the IEEE Standard for Local and Metropolitan Area Networks – Timing and Synchronization for Time-Sensitive Applications. Protocols, procedures, and managed objects for the transport of timing over local area networks are defined in this standard. It includes the transport of synchronized time, the selection of the timing source (i.e., best master), and the indication of the occurrence and magnitude of timing impairments (i.e., phase and frequency discontinuities).

3) FREEDOM FROM INTERFERENCE (FFI)
ISO 26262 defines freedom from interference as the absence of cascading failures between two or more elements that could lead to the violation of a safety requirement, where ''element'' is a system or part of a system including components, hardware, software, hardware parts, and software units, and ''cascading failure'' is a failure of an element of an item causing another element or elements of the same item to fail.

That said, ISO 26262 specifies three types of interference: timing and execution, memory, and exchange of information.

- Interference in the time domain occurs when execution of a safety-relevant software element is blocked due to a fault in another software element. This type of problem can be detected and handled by handshake mechanisms like a watchdog.
- Interferences in the memory space domain occurs when a software element accesses or modifies code or data belonging to another software element. This type of interference is related to corruption of memory content and device configuration data. Such interferences are typically detected at run-time by a hardware component called the Memory Protection Unit (MPU).
- Interferences due to exchange of information are sender and receiver related and are caused by errors such as: repetition of information, loss of information, delay of information, insertion of information, blocking a communication channel, etc. These types of problems are detected by handshake mechanisms provided with data integrity checks.

a: SAFETY ISLAND

In order to isolate the subsystem that deploys the safety-relevant functionality from the rest of the system (non-safety-related) to avoid thus any kind of interference, the functional safety subsystem is architected as a building block that guarantees the freedom from interference from any other external source, independently of being this solution implemented in hardware or software.

4) RELIABILITY

a: STATIC ROUTING

Static routing and dynamic routing are two methods used to determine how to send a packet toward its destination. Static routes are configured in advance of any network communication. Dynamic routing, on the other hand, requires routers to exchange information with other routers to learn about paths through the network. Static and dynamic routing are used where appropriate, and some networks use both.

In reduced and non-changing networks like in a vehicle, the use of static routing, or non-adaptive routing, contributes to reduce complexity, especially in terms of orchestration. Static routing uses small routing tables normally with only one entry for each destination. It also requires less computation time than dynamic routing because each route is preconfigured. Because static routes are preconfigured, administrators must precompute upfront or manually reconfigure routes to adapt to changes in the network when they occur. In the case of automotive, it could be possible to reconfigure static routing tables as response to certain types of failures.

b: SOURCE ROUTING

In alignment with Static Routing, source routing comprises a viable approach to the reduction of forwarding state [33], [34]. This state reduction can yield significant switch Ternary Content Addressable Memory (TCAM) savings in the Match&Action stage of the networking device, allowing for cheaper switching hardware. In principle, source routing encodes the path into each packet header, enabling switches to forward packets using a minimal number of (nearly static) flow independent forwarding entries. In particular, the path is encoded as a set of labels which correspond (or may even match) to the sequence of switch ports that each packet needs to traverse. As such, there is no need for the switches to maintain L2 or L3 forwarding entries to all destinations within the network, saving a significant amount of TCAM space.

c: ACTIVE QUEUE MANAGEMENT (AQM)

AQM schemes are strategies of interest in the field of congestion control mechanisms [35]. They exploit the idea that an incoming packet can be dropped by the networking device even if there is some available buffering space in it as a countermeasure to save space for safety critical frames to come.

d: SMART QUEUEING

Another more conservative option instead of deciding to drop a frame is to try to accommodate the ingress frame in the appropriate space of the internal buffers of the device when there is free space yet in order to avoid thus the retransmission of frames by the sender, fact that would increase the traffic, for instance in the case of dropped messages managed by TCP/IP.

5) TIME DETERMINISM

Many applications deployed in a distributed cyber physical system demand a bounded end-to-end delay or latency as response, in alignment with some safety requirement. Several mechanisms are developed and extensively used nowadays to accomplish this type of time deterministic demands.

a: TRAFFIC SHAPERS

Traffic shaping is an adopted strategy in order to fit the traffic of the network into time deterministic requirements targeting specific end-to-end bounded delays. Many types of traffic shapers are nowadays in place in many verticals and industries, also considered in the automotive field: Asynchronous Traffic Shaper (ATS) [36], Time Aware Shaper (TAS) [37], Credit Base Shaper (CBS) [38], Stream Reservation Protocol (SRP) [39], Frame Preemption [40], etc.

b: WATCHDOG

A watchdog timer is a specialized timer module that helps a microprocessor or cyber physical system in general to recover from SW malfunctions. When a watchdog timer expires, i.e., it reaches the end of its counting period, it resets the entire processor system. In order to prevent this, the microprocessor that is monitored by the watchdog must perform some type of specific action that refreshes the watchdog by restoring the timeout interval. Thus, a watchdog timer can be configured such that it will reach the end of its counting interval only if a processor failure has occurred, and by forcing a system reset, the watchdog timer helps the processor to escape from the failure mode and restart normal operation. There are two types of watchdog mechanism: normal and windowed. A normal watchdog causes a reset if the counter expires (i.e., not refreshed in time). A windowed watchdog is more stringent in terms of accuracy of the microprocessor-watchdog handshake and also causes a reset if the refresh occurs out of the configured window, either too soon or too late of the upper and lower time thresholds defined.

6) SELF-MONITORING

Fast failure detection is critical in many FuSa applications. To achieve this goal, it is necessary to add monitoring strategies able to detect failures and trigger the required safety reaction.

a: BUILT-IN SELF-TEST (BIST)

The term Built-In Self-Test (BIST) is used to describe the on-chip hardware mechanisms that can be used to detect latent faults within an electronic device, e.g. SoC or microcontroller unit (MCU) [41]. The BIST allows the device to conduct periodic self-tests to identify faults. The results of these self-tests can then be used by the device to handle the faults and ensure that the system remains in a safe state. These self-tests are usually performed in startup time but also triggered periodically at run time. Some examples are memory BIST, both flash and RAM.

b: DATA SCRUBBING

Data scrubbing is a technique used to reconfigure or reprogram a device, for instance a RAM-based FPGA. It can be used periodically to avoid the accumulation of RAM errors without the need to find them, but just eliminating them by rewriting the right configuration, simplifying thus the detection and corrective process. The same strategy can be applied to the configuration of RAM-based registers in a microprocessor. Numerous approaches can be taken with respect to scrubbing, from simply reprogramming the full device to partially reconfiguring it. The simplest method of scrubbing is to completely reprogram the device at some periodic rate taking into account the effect of having an error present for a given specific time (i.e., the selected period) before being fixed and the consequences of it on the full system.

c: DATA POLLING AND EVENT-DRIVEN PROCESSING

Other classical ways of self-monitoring are data polling (i.e., periodic reading of inputs) or event-driven conditions derived in hardware interrupts or software callback functions when certain conditions are met.

d: DATA INTEGRITY CHECK

In networking, every transmitted frame is provided with a field in the trailer to check the integrity of the data and ensure that the frame is consistently created and transported from end to end. This mechanism is intended to detect errors in case of frame inconsistency or data corruption due to some noisy environment affecting the data transmission. Some examples of algorithms used to check the data integrity are: Parity bit (PB), checksum (CS) or cyclic redundancy check (CRC).

e: ALIVE COUNTER

A liveness strategy commonly used in networking is the use of an Alive Counter built-in in the header of frames in order for the receiver/listener to check that the sender/talker is alive by increasing that counter in each transmitted frame. The fact of incrementing the counter at every transmission is a proof that the sender/talker remains in operation (i.e., its software program flow is not lost).

7) FAIL SAFE AND FAIL OPERATIONAL RESPONSE

Guarantee-of-service demands self-healing strategies when failures occurs. There exist several strategies applicable to in-vehicle networking, as detailed next.

a: FAST FAILOVER

In-vehicle applications which are deployed and affected by networks in autonomous and automated vehicles demand a fail-operational behavior. The network components must not only detect failures, but also reduce their effects in terms of safety countermeasures. An in-vehicle network must safely deal with transient and permanent failures that are critical for passengers and the environment. For autonomous systems, safe operation even in the event of a failure (''fail-operational'') is of great importance. A simple stop of the system (''fail-safe'') is not possible in most cases and they demand a fast failover solution [42].

b: FAST RE-ROUTE (FRR)

Most modern networks support different kinds of fast-reroute (FRR) mechanisms which leverage pre-computed alternative paths at any node towards any destination. When a node locally detects a failed link or port, it can autonomously remove the corresponding entries from the forwarding table and continue using the remaining next hops for forwarding packets: a fast local reaction [43]. In FRR solutions implemented on Software Defined Network (SDN) compliant devices, the control plane is hence just responsible for pre-computing the failover paths and when a failure occurs the data plane utilizes this additional state to forward packets. This reaction needs to be aligned with the fault tolerant time interval (FTTI) or fault reaction time interval (FRTI) determined for each safety goal.

c: LOAD BALANCING

Load balancing is one of the reliability strategies of distributed architectures such as cloud services [44] or automotive IVN E/E architectures [45]. Existing load balancing algorithms are mainly classified into two categories: static and dynamic. Static load balancing algorithms allocate tasks with a fixed probability or order, regardless of the current node states, such as the round-robin algorithm.

The static algorithms are simple, but they only work properly in a system with a low variation of load and similar processing capacity in each node. Dynamic load balancing algorithms use real-time node load and health states to determine task distribution, such as the consistent hashing algorithm, the least connection method, and the least response time method. The dynamic algorithms improve node resource utilization and avoid overload, but they usually lead to poor throughput and long response latency due to the high algorithm complexity.

d: RUN-TIME RECONFIGURATION

Another responsive strategy is the run-time reconfiguration of the system as response to failures by reconfiguring on the fly, either in SW or in HW, certain subsystem while the system keeps in operation.

8) OTHER RELATED STANDARDS

Some automotive related standards incorporate also safety and security approaches. Some of them are discussed next.

a: AUTOSAR

AUTOSAR (AUTomotive Open System ARchitecture) is a worldwide development partnership of vehicle manufacturers, suppliers, service providers and companies from the automotive electronics, semiconductor and software industry. The primary goal of the AUTOSAR partnership is the standardization of a common methodology, basic system functions and functional interfaces. This enables development partners to integrate, exchange, re-use and transfer functions within a vehicle network and substantially improves their efficiency of development. Having this goal in mind, AUTOSAR pushes the paradigm shift from an ECU-based to a function-based system design attempt in automotive software development and enables the management of the ever-growing software and E/E complexity with respect to technology and economics. Many of the standardized aspects included in AUTOSAR deal with safety and security measures, also for the in-vehicle network [46], [47].

b: DDS

Data Distribution Service (DDS) is a middleware protocol and API standard for data-centric connectivity from the Object Management Group (OMG). It integrates the components of a system together, providing low-latency data connectivity, extreme reliability, and a scalable architecture that business and mission-critical applications need [48]. In a distributed system, middleware is the software layer that lies between the operating system and applications. It enables the various components of a system to more easily communicate and share data. Like this, most middleware works by sending information between applications and systems. DDS is uniquely data centric and data centricity ensures that all messages include the contextual information an application needs to understand the data it receives. The essence of data centricity is that DDS knows what data it stores and controls how to share that data. Furthermore, DDS uses a global data space that lets applications share information with full control of reliability and timing. That is, DDS middleware takes full responsibility for both the distribution of data (from producers to consumers and from publishers to subscribers), as well as the management of data (such as maintaining non-volatile data for late-joining applications).

TABLE 1. Examples of FuSa related functions inside a vehicle [27], [26].

Subsystem	Possible hazard	ASIL
Engine management	Unwanted acceleration	C-D
Head Lights	Both side failure	B
Radar Cruise Control	Inadvertent braking	B
Electric Power Steering	Incorrect self-steering	D
Vision ADAS	Incorrect sensor feedback	B
Active Suspension	Suspension oscillations	B-C
Antilock braking	Unintended full power braking	D
Brake lights	Both side failure	B
Rear View camera	No valid sensor data	B
Rear Lights	Both side failure	A
Instrument Cluster	Loss of critical data	B
Airbag	Inadvertent deployment	D

C. STATE-OF-THE-ART FUNCTIONAL SAFETY DEPLOYMENT: SOFTWARE-CENTRIC APPROACH

Most of the state-of-the-art automotive FuSa solutions applied in current vehicles are clearly SW-centric. Although the ISO 26262 standard covers both HW and SW development cycles, a high percentage of the FuSa related contributions in terms of effective functionalities adopted by the automotive electronics industry and synthesized in real vehicles are software-based. That is, both the FuSa concept of the solution and the integration and manufacturing of the ECU that embeds such solution are sustained through the implementation of safety mechanisms directly in SW.

Some example of the most relevant FuSa related functions currently integrated in a vehicle, according to [49], are summarized in Table 1. In this context, concerning HW contributions to the aforementioned SW-centric FuSa solution, this is typically an automotive MCU or SoC device equipped with particular ISO26262 features. For instance, CPU cores able to run in lockstep as well as many other powerful features in terms of built-in monitoring and diagnostic strategies (e.g. watchdog, self-tests, etc.). As a reference design, we could showcase a SW-centric solution dominated by AUTOSAR Classic on a typical MCU/SoC, for instance ARM Cortex R5, running code. Some examples are Infineon AURIX [50], NXP S32G [51], or ST SPC5x [52].

In the next-generation E/E architectures, the functional scope of the GW controller is being extended from only networking functions to now the combination of both networking and power distribution functions per zone. That means that this zonal controller takes major responsibilities in terms of safety countermeasures affecting to the reliability of power supply and its distribution across each zone.

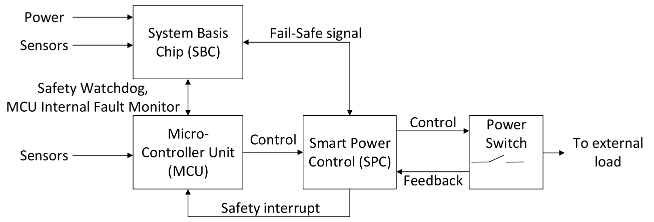

FIGURE 3. Safety architecture for electric power steering function as part of a zonal GW controller decomposed in MCU+SBC+SPC.

In this direction, a typical safety architecture addressed for these cases is shown in Fig. 3, applied to the specific use case of the Electric Power Steering. This example shows a safety concept distributed in HW through three key components: (i) the system MCU, (ii) a second safety-related element devoted to ''Power Management and Safety Monitoring'' functions and (iii) a Smart Power Controller (SPC) or Power Switch device responsible for the control of the power supply to an external load. The second element could be either a dedicated safety MCU or a System Basis Chip (SBC) –dedicated chip equipped with safety measures like watchdog for supervision of the system MCU and power monitoring system. The third element, SPC, is also equipped with safety mechanisms like self-diagnostic and monitoring, able to switch off by its own the load in case of detecting abnormal conditions like over temperature or overcurrent (e.g. electric shortcut on the load). These three elements together compose the HW architecture of the safety concept. Then, the safety solution able to achieve the specific targeted ASIL level depends on the SW strategy implemented through these three components (i.e. program code running on the MCU including the configuration of both smart devices, SBC and SPC). Many of the SW strategies considered in the uses cases above are standardized in AUTOSAR [46]. As deduced from this architecture, the safety goals of the EPS are accomplished by the concurrent deployment of some of the safety mechanisms detailed previously in section II.B, like watchdog, BIST, data polling, data integrity check or fast failover, among others. Another comprehensive example of deployment of safety process is the one elaborated by AUTOSAR for front light management in [53].

III. NEW TRENDS IN FUNCTIONAL SAFETY DEPLOYMENT: MORE HW-CENTRIC FUSA SOLUTIONS

As it is stated in [47]: ''Functional safety is a system characteristic which is taken into account from the beginning, as it may influence system design decisions. Therefore, AUTOSAR specifications include requirements related to functional safety. Aspects such as complexity of the system design can be relevant for the achievement of functional safety in the automotive field. Software is one parameter that can influence complexity on system level.

New techniques and concepts for software development can be used in order to minimize complexity and therefore can ease the achievement of functional safety. AUTOSAR supports the development of safety-related systems by offering safety measures and mechanisms. However, AUTOSAR is not a complete safe solution. The use of AUTOSAR does not imply ISO26262 compliance. It is still possible to build unsafe systems using the AUTOSAR safety measures and mechanisms.''

The authors of this case study, as researchers in the automotive electronics domain, fully agree with the previous AUTOSAR statement. In fact, we corroborate that in many cases a HW-centric solution is safer and more trustable than a SW-centric solution, especially in terms of aspects like time determinism (i.e. clock cycles) and accuracy/control. In complex systems like automotive, where many inherent SW disturbance factors or sources coexist in the solution (like multicore processing, arbitration or shared resources - e.g. memory -, multithreading, OS and hypervisors, large set of different and heterogeneous interrupt sources, physical restrictions, etc.), the fact of making an estimation of when a certain piece of code will be executed to guarantee certain behavior in the vehicle is complicated ''by construction''. All these aspects have even major relevance when they need to coexist with critical safety functions or stringent quality-of-service (QoS) requirements.

In this particular context, i.e. automotive use cases that HW/SW developers –from any OEM/Tier-1/Tier-2 in the automotive electronics industry– deal with on a daily basis, there is another different trend –with less followers today but that we expect will grow in the near future– consisting in shifting part of the HW/SW FuSa related solution directly to HW. Like this, HW-centric solutions often bring more consistent approaches in terms of scientific rigor, simplicity in proof, accuracy and time determinism than SW-centric solutions. In other words, SW solutions are more unpredictable and provoke more uncertainties whereas in HW solutions some of these effects can be skipped and solved by design.

A. THE CASE FOR HW-CENTRIC FUSA SOLUTIONS IN AUTOMOTIVE

There are several works discussing the advantages of HW solutions for automotive systems such as [54] and [55], and more specifically about safety and security HW based solutions as in [56] and [57]. The New Era of Mobility materialized through the visionary Autonomous-Connected-Electric-Shared (ACES) vehicle is well-known by its motto ''Software-Defined Vehicle'' or SDV in short. However, authors sustain the view that the SDV concept is not possible without new emerging ASIC/SoC devices with dedicated networking hardware coprocessors inside, able to embed at least part of the most stringent and high-demanding automotive functional features, requirements and standards which are seamlessly related to high performance, time determinism, functional safety and cyber security. In other words, software alone is not enough. New dedicated hardware coprocessors are needed to support the ACES vehicle conception.

The authors strongly believe the solution shall come out from the right balance of HW/SW codesign, giving rise to novel Network/System on Chip (NoC/SoC) devices equipped with new HW-centric peripherals or coprocessors which integrate advanced safety mechanisms able to contribute to the monitoring, self-diagnostic, self-test and control of all those safety-related and fault-tolerant subsystems of the vehicle. The presence of these new HW-based coprocessors aims at helping system architects and software developers to notoriously reduce the complexity of the safety solutions by trying to displace safety-related functionality implemented in SW towards deterministic and reliable HW-centric mechanisms.

In this regard, especially in the area of in-vehicle networking and gatewaying, we start to see how new MCU/SoC devices emerge in the market equipped with dedicated networking engines intended for offloading the system/host CPU from those more stringent tasks which are now performed directly by dedicated HW coprocessors. Some examples already available or announced in the automotive electronics industry about those more HW-centric networking solutions are: NXP S32G Vehicle Network Processor with Packet Forwarding Engine (PFE) [51] or Infineon Aurix TC4x provided with Data Routing Engine (DRE)) [50].

B. IDENTIFIED CHALLENGE: FUNCTIONAL SAFETY DEPLOYMENT IN SOFTWARE DEFINED VEHICLES

Based on the previous state of the art and future trends analysis, we identify a need for new strategies to deploy functional safety mechanisms in software defined vehicles. As stated before, the different components of the vehicle are delivered to OEMs as Safety Elements out of Context (SEooC) which they need to integrate in their designs. This integration can be a very arduous task when done bottom-up since changes to low level elements might imply a new complete re-assessment of the whole product from a safety point of view. On the opposite side, the software defined vehicle philosophy is trying to escape from this, providing a software oriented platform, fully portable regardless of the HW below (HW agnostic).

What we see, is that a bridge between these two flows is missing. To the best of our knowledge, there is no methodology available in the state of the art that allows for integrating different FuSa mechanisms in a Software Defined Vehicle with the right level of flexibility, scalability and simplicity. Such a methodology is a key factor in order to enable a new generation of safety compliant SDVs. From our perspective, this can be achieved with the right combination of HW and SW features in low level devices that enable a reasonable level of abstraction over the low level functionalities and safety mechanisms.

On top of this, we propose a paradigm shift, moving from a bottom-up approach to a top-down approach, and show how it simplifies the deployment of new safety concepts for different HW and SW platforms.

Our proposal is to start the safety concept analysis at network level. We start by defining the NGIVN, its components, signal groups and their safety goals. Then, we map the safety goals to the different signal groups for each component. By doing so, we are able to provide a safety-compliant framework which guarantees the fulfillment of the safety goals, i.e. ASIL compliance is guaranteed by design. In our case study, we provide a definition of a possible NGIVN, its components and their related signal groups. Then we focus on one element of the NGIVN: the zonal GW controller. We derive the safety requirements of the gateway based on the safety goals and the definition of the NGIVN. The last step is to identify the possible failure modes of the gateway and to allocate the required safety mechanisms that allow to fulfill the safety goals. For this analysis we use our elastic gateway (eGW) concept, which is a new generation GW architecture specifically tailored for the needs of automotive networking devices. Through our case study we show possible failure scenarios that could happen in the NGIVN and select suitable safety mechanisms to overcome them. The selected safety mechanisms shall guarantee the fulfillment of the safety goals. Afterwards we allocate this safety mechanisms to the eGW, ensuring thus full compliance with the functional safety concept described at NGIVN level. An overview of the proposed top-down approach is depicted in Fig. 4.

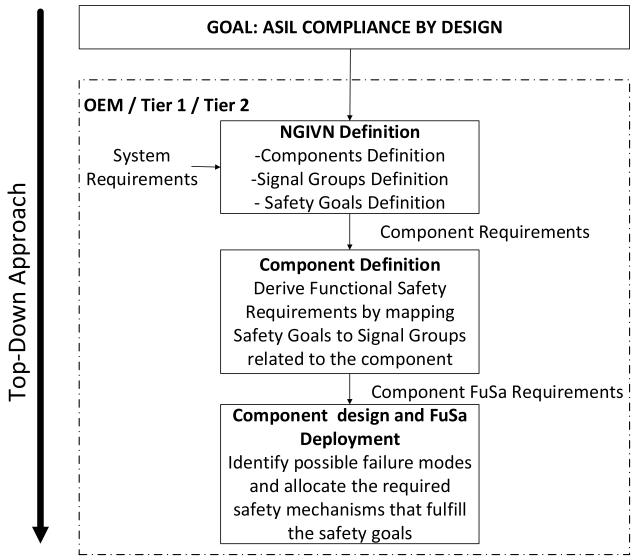
Top-Down Approach
GOAL: ASIL COMPLIANCE BY DESIGN
OEM / Tier 1 / Tier 2
System Requirements
NGIVN Definition
-Components Definition
-Signal Groups Definition
- Safety Goals Definition
Component Requirements
Component Definition
Derive Functional Safety Requirements by mapping Safety Goals to Signal Groups related to the component
Component FuSa Requirements
Component design and FuSa Deployment
Identify possible failure modes and allocate the required safety mechanisms that fulfill the safety goals

FIGURE 4. FuSa top-down approach proposed.

Finally, we highlight the innovative contribution of our approach which is its flexibility and elasticity when integrating new safety mechanisms in the NGIVN in a safety compliant manner. Indeed, our process enables the specification and design of a large variety of eGW safety concepts, each answering to specific customer needs simply by adjusting the safety goals allocated to the NGIVN, and, as a consequence, the safety and availability capabilities of the eGW. This implies the design of a family of gateway devices, where each device is unique with regards to safety features and targeted application with minimal additional effort.

IV. NEW GENERATION IN-VEHICLE NETWORK

The New Generation In-Vehicle Network, NGIVN, is a system-level Safety Element out of Context (SEooC) as described in ISO 26262 Part 1. It means it is not an item or item function but a system (made of multiple SW and HW elements) used by various item functions of OEMs.

The main goal of the NGIVN is to connect elements of the automated driving (AD) system [58]. In terms of functionality, the NGIVN is a fault-tolerant network (called hereafter fabric) of interconnected new generation automotive gateways (called hereafter VIUs for Vehicle Intranet Unit or Vehicle Interface Unit) connecting various vehicle ECUs, networked sensors, and high-performance CPUs, as depicted in Fig. 5. The main advantage of the NGIVN compared to classic gateways (that only perform gateway functions) is the support to OEMs in fulfilling safety-related availability requirements by supplying the following functions:

1. Fault-tolerant networking
2. Load balancing – in case of failure of High-Performance CPUs (HPC), logical migration of functionality of faulty HPC on the NGIVN.

The NGIVN does not directly access any sensors or actuators. It is within the responsibility of the network peers.

The following list summarizes the main functions of NGIVN:

1. Read-only gateway (no change of safety-related data)
2. Modifying gateway (a gateway that reads safety-related data and processes/changes it)
3. Network redundancy (multiple IVNs with redundant connections, with multiple transmission routes)
4. Network monitoring and node monitoring (detecting issues like dead link, failure of a node, a failure of a given TCP connection, or similar)
5. Network reconfiguration (changing the routing depending on failures, in case of detected errors)
6. Redundant transmission over multiple links/paths
7. Load balancing/function migration: migration (from logical perspective) of function from High-performance CPU onto NGIVN (in case of failure of High-performance CPU).

Note that from a logical perspective, to realize the migration, it might be necessary to have, e.g., a hot standby software running in NGIVN.

In the next section we can find the assumed stakeholder requirements related to the main functions of the NGIVN.

A. NGIVN STAKEHOLDER REQUIREMENTS

This section defines top-level requirements to be fulfilled by the NGIVN focusing only on the core IVN functions.

[STR_IVN_0001] Gatewaying of In-Vehicle Data

The NGIVN shall transmit (gateway) defined data by network communication between the in-vehicle communicating peers:

1. Sensor ECUs
2. Actuator ECUs
3. HPCs
4. Smart/networked vehicle sensors (e.g., Ethernet cameras)
5. Tools/devices connected over On Board Diagnostics interface

The transferred data can be:

1. Application payload, data from sensor ECU to actuator ECU, e.g., transfer of gear shifter position from gearbox ECU to engine ECU
2. Software updates/image files
3. Logging/audit data

[STR_IVN_0002] Gatewaying of V2V and V2C Data

The NGIVN shall transmit (gateway) defined data by network communication between the communicating peers from/to the Internet, meaning:

1. Other vehicles (vehicle to vehicle (V2V))
2. Other systems on the Internet (vehicle to cloud (V2C) or vehicle to infrastructure (V2I))
3. ECUs (network nodes) within the vehicle.

The transferred data can be:

1. Application payload, e.g., GPS position, the position of other vehicles, driving situation, GPS maps (e.g., current tile), traffic information
2. Software updates/image files
3. Logging/audit data

[STR_IVN_0003] Taking over of the Workload from HPCs

The NGIVN shall take over the workload of HPCs in defined conditions by executing appropriate Adaptive applications.

For instance, in the case where an HPC was to perform a reset due to a detected error, the NGIVN shall execute a backup function. The backup function is either hot standby or cold standby, depending on its real-time properties.

Taking over the workload can be considered a form of load balancing, where the load (workload) is logically transferred from an HPC to a VIU.

Note: By adaptive application, it is meant a C/C++ application using POSIX and C++ interfaces and which does not use any "classic" input/output peripheral access (e.g., PWM, Digital Input Output or Analog to Digital Converter). However, it can use specific real-time network or hardware features, e.g., Time Sensitive Networking (TSN) shapers [59].

An adaptive application can be an AUTOSAR application, ROS application, or similar.

[STR_IVN_0004] Hosting of Adaptive Applications

The NGIVN shall host defined Adaptive Applications on VIUs and provide specific hardware features to support them. For instance, the conversion of a PWM signal, received from a wheel speed sensor over Ethernet, into a speed information in [m/s].

Adaptive Applications may have any cybersecurity level and any safety integrity level, including QM. Applications need to comply with various limitations, e.g., they need SW interfaces provided by VIUs, meaning there will be no peripheral access (just network input and outputs and inputs/outputs from/to other applications).

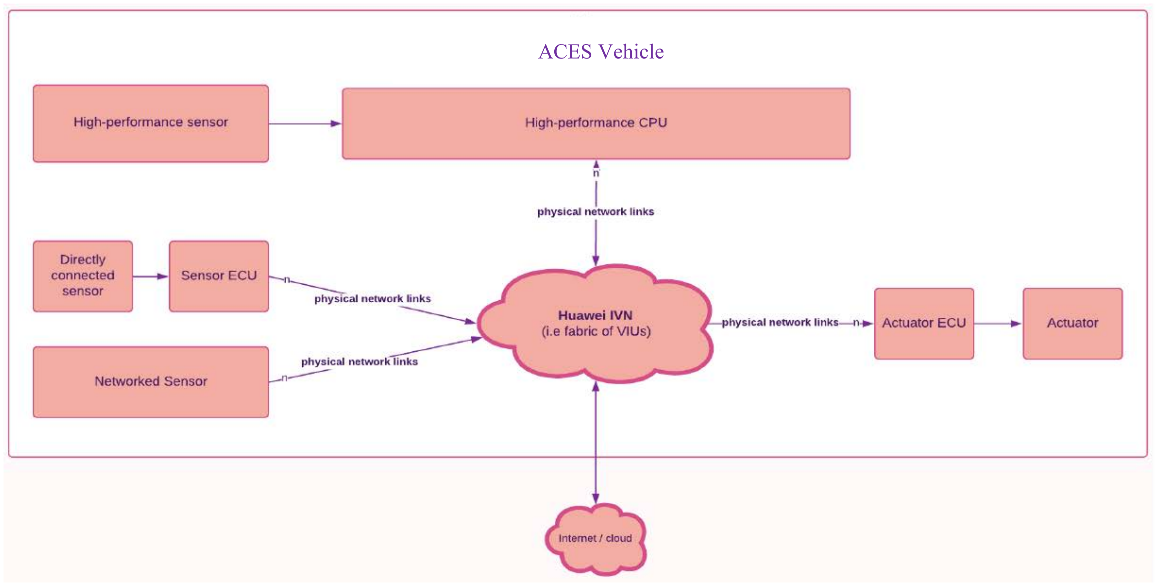

FIGURE 5. NGIVN context in vehicle architecture.

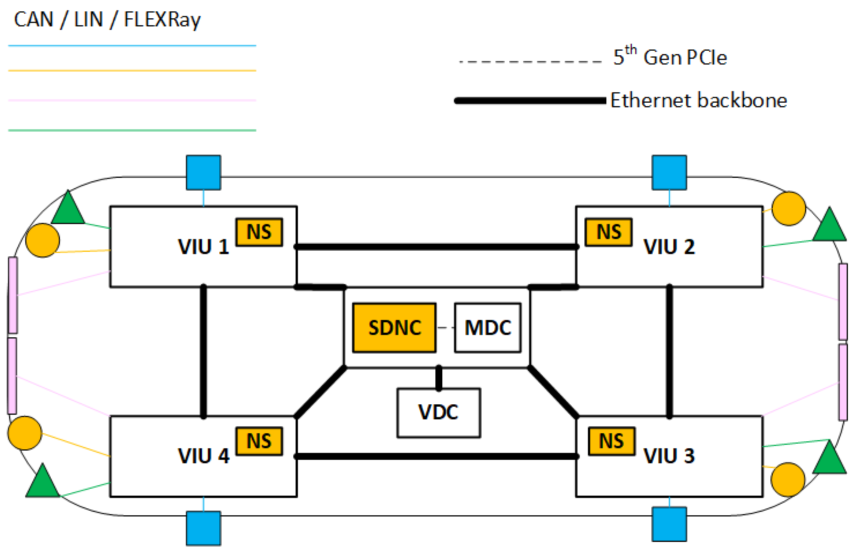

FIGURE 6. NGIVN deployment in ACES vehicle.

The architecture, topology of NGIVN, number of VIUs is analyzed and evaluated in the next section. In particular, a token-ring architecture is addressed.

B. NGIVN ARCHITECTURE OVERVIEW

In this section we describe the NGIVN architecture proposed in this work. It is an Ethernet-based zonal network architecture, where sensors and actuators are connected to zonal gateway controllers via different communication protocols, while the zonal controllers and main computing units (High Performance Computers, HPCs) are interconnected through an Ethernet backbone. Figure 6 shows an overview of the layout of such network in the vehicle. The zonal controllers correspond with the Vehicle Interface Units (VIUs) in the figure, and the High Performance Computers are the three nodes in the middle: 1 Vehicle Dynamics Controller, VDC, 1 Mobile Data Center, MDC, and 1 Software-Defined Network Controller, SDNC. Each VIU contains a Node Supervisor (NS), which is in charge of monitoring the status of the node and exchanging information with the SDNC. This is a key element of the network enabling the software defined paradigm for safety related features, as explained later in Section V.

As introduced before, in order to keep up with the demands of ACES and Software-Defined Vehicles, new strategies and architectures need to be deployed in the area of in-vehicle communication. One of these strategies is Software Defined Networking, which is a necessary step in order to enable Software Defined Vehicles. The management of the configuration in Software Defined Networks, and also in vehicular networks, is a hot topic under discussion in today's community. In [60] an overview of how SDN technology can be interesting for the automotive industry and some use cases are exposed. In [61] and [62] authors explore the benefits of using SDN in IVNs. In [63] authors propose the use of SDN in automotive CAN-based networks while [64] proposes an SDN architecture for Ethernet-based automotive networks.

From standardization point of view, IEEE802.1Qcc [65] defines means to perform the configuration of Time Sensitive Networks. One of the options defined in the standard that seems to make more sense for automotive networks is the Centralized Network / Distributed User Model. This model defines a Central Network Configurator that is in charge of configuring and monitoring the network. Authors in [66] and [67] explore the integration of TSN with SDN and propose different strategies for this combination.

Regarding safety and security, an SDN controller for safety applications is proposed in [68] and in [69] a fault tolerant dynamic scheduling for TSN networks is introduced. In [70] authors propose an SDN-based strategy for fast fail over routes in the data plane.

Following this approach, one of the HPCs integrated in the NGIVN is an SDN Controller (SDNC) which is in charge of the configuration and status monitoring of the network. This provides the required flexibility in terms of software defined network configuration. This also enables compliance of the NGIVN with TSN standards. As introduced before, standardization is of high importance in automotive due to the complexity of the environment and the amount of stakeholders involved.

Additionally, the software defined approach also enables the deployment of a Service oriented Architecture (SoA). In a SoA applications are composed of services which can be deployed in one processing unit or another. This means that the NGIVN allows to shift the processing of one VIU to another VIU, or even from one of the HPCs to a VIU in some cases. A distributed SoA enables new functionalities like sharing services or applications between the vehicle and user smartphones as explored in [71], or the interaction with the infrastructure for autonomous driving, as proposed in [72]. In terms of safety, this software defined approach is key in bringing new strategies to enable fault-tolerant behavior in the NGIVN. For instance, in case of a node fail, another node could take over the processing tasks allowing to continue system operation regardless of the failure.

The detailed architecture of the NGIVN proposed in this work, including sensors Signal Groups and application services hosted in the different HPCs, is represented in Fig. 7.

As seen before, this network is composed of 4 VIUs interconnected in a ring topology, 1 Vehicle Dynamics Controller, VDC, 1 Mobile Data Center, MDC, and 1 Software-Defined Network Controller, SDNC. The VDC is in charge of Electric Power Steering (EPS) and Automatic Emergency Breaking (AEB), while MDC is in charge of Adaptive Cruise Control (ACC). All VIUs have a direct connection to the High-Performance Computer Hub, HPC Hub, composed of the SDNC and the MDC. The HPC Hub is, in turn, connected to the VDC. The VIUs are also connected to sensors and actuators modelled here as signal groups and to other ECUs such as the TCU or the Infotainment Unit. The VIUs act as zonal gateways but also as switches by forwarding the data coming from the different signal groups to the relevant High-Performance Computer, HPC. They are also capable of taking over pre-specified functions of the HPCs in case of a failure of an HPC. The next section defines all the network components present in Fig. 5, and the path taken by the frames belonging to the various signal groups.

C. NGIVN COMPONENTS DESCRIPTION

Table 2 describes the SoCs/ECU introduced in Fig. 7, and Table 3 describes the signal groups depicted also in Fig. 7.

D. NGIVN SAFETY GOALS

This section describes a representative set of Safety Goals defined in the context of the NGIVN. To derive this safety goals, we define Item functions according to 3-5.5.1 in [18] and perform a Hazard analysis and risk assessment (HARA) report according to 3-6.5.1 in [18]. The safety goals presented here are created out of the HARA and allocated to the items containing the NGIVN. These safety goals will then be derived into functional safety requirements as we will see more in detail later on.

1) NGIVN SAFETY GOALS CLASSES

The NGIVN is application-agnostic / application-independent.

It can connect any network nodes that realize any vehicle item functions, focusing on SAE [58] Level 0 to Level 5 functions.

These item functions result in different functional safety requirements that the NGIVN needs to fulfil. Therefore, the item functions are described here and grouped/clustered by classes.

- **Class 1: Item Functions with no Safety Goals**

These functions have no safety goals, no Fault Tolerant Time Interval (FTTIs), no ASIL. The NGIVN can perform, if needed, load balancing for these functions. There are no safety goals for these functions, just QM requirements. Table 4 shows some examples of item functions with no safety goals.

- **Class 2: Item Functions with only Fail-Safe Safety Goals**

In this class of the item functions, there is a clearly defined safe state set by the actuator HW or actuator ECU in case of a system malfunction. It can be de-energize / power off (e.g. flasher off) or energize / power on (e.g. brake light). The NGIVN can switch to a defined fail-safe state for these functions. Table 5 shows some examples.

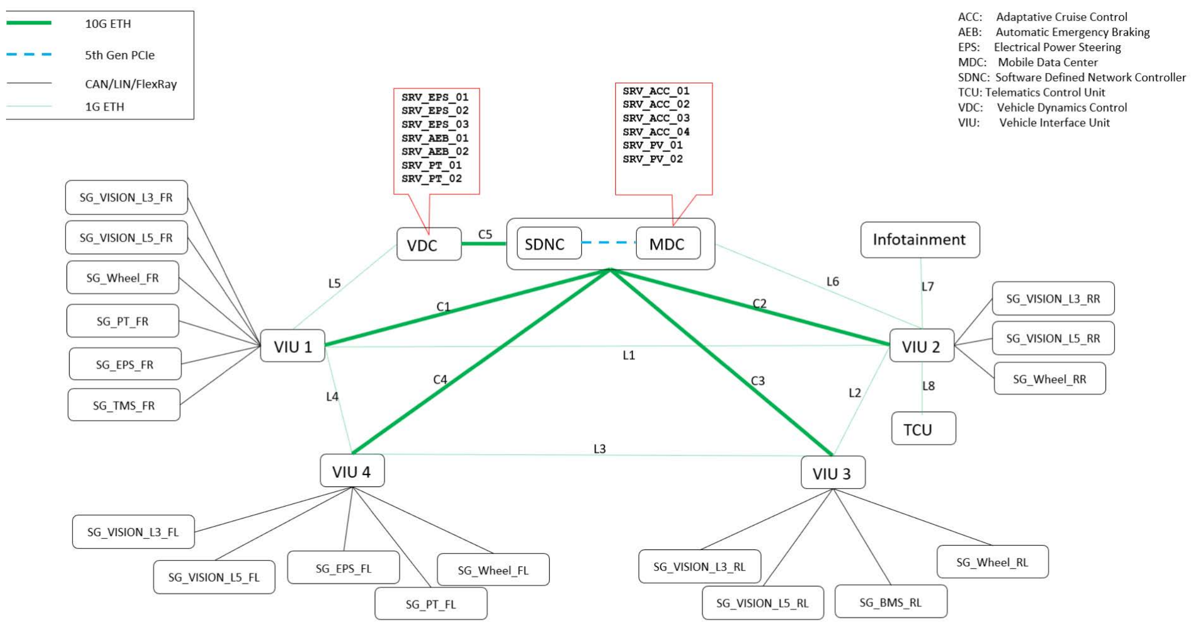

FIGURE 7. NGIVN architecture.

TABLE 2. ECUs specification table.

SoC/ECU	Description	Primary Application	Secondary Application
VIU1	SoC responsible for forwarding the frames of front right signal groups to the rest of the network. Also responsible for VDC backup in combination with VIU4	Frame forwarding and TMS control	VDC back for EPS and PT applications
VIU2	SoC responsible for forwarding the frames of rear right signal groups to the rest of the network. Also responsible for forwarding frames to/from TCU and infotainment system	Frame forwarding	Communication with TCU and Infotainment
VIU3	SoC responsible for forwarding the frames of rear left signal groups to the rest of the network. Also responsible for control of the BMS	Frame forwarding	BMS control
VIU4	SoC responsible for forwarding the frames of front left signal groups to the rest of the network. Also responsible for VDC backup in combination with VIU1	Frame forwarding	VDC back for EPS and PT applications
VDC	SoC responsible for the control of vehicle dynamics. It hosts and is the primary path for the execution of AEB, PT and EPS strategies	EPS, AEB, PT	-
MDC	SoC responsible for the execution of AD application such as ACC and PV. The same SoC hosts both the MDC and the SDNC	ACC, PV	-
SDNC	SoC responsible for the control of the network configuration and the execution of the fail-operational network reconfiguration mechanism. The same SoC hosts both the MDC and the SDNC	Fail-operational network reconfiguration	-
TCU	ECU responsible for the vehicle communication with the cloud (V2X) and with other vehicles (V2V)	V2X	V2V
Infotainment	ECU responsible for the execution and display of infotainment applications	Infotainment	

- **Class 3: Item Functions with Safety Goals Requiring Availability, with Availability Addressed at Actuator Side**

In this class of the item functions, we can find safety-related availability requirements allocated to the item function, but achieved at the actuator ECU side.

In other words, the rest of the systems realizing the function are fail-safe. NGIVN can switch to a defined fail-safe state for these functions.

Table 6 provides some examples.

- **Class 4: Item functions with Safety Goals Requiring Availability, with Availability Addressed at Multiple Systems of the Item**

In view of ISO 26262 [18], in this class of item functions, safety-related availability requirements are allocated to the item function, and they impact several systems realizing this item function.

Also, in view of ISO 4804 [73], in this class of the item functions, there are functions of fail-operational items and functions of fail-degraded items. Fail-operation items must provide the required level of performance even in presence of failures, while fail-degraded items are allowed to reduce or degrade performance to some extent for the sake of safety.

Here, the NGIVN cannot switch to a defined fail-safe state for these functions, as a defined fail-safe state of the item functions does not exist. For example, the NGIVN cannot just stop sending messages or send an error message, as the

TABLE 3. Signal groups specification.

Sensor Group	Description	Sensor Type	BW [Mbps]	Target	Path
SG_ Wheel_ FR	Front right wheel sensors (speed, pressure, etc.) and actuators signals	Analog / Digital IO	64	VDC	L5
SG_PT_ FR	Front right powertrain sensors (speed, pressure, etc.) and actuators signals	Analog / Digital IO	64	VDC	L5
SG_EPS_FR	Front right EPS sensors (angle, , torque, pressure, etc.) and actuators signals	Analog / Digital IO	64	VDC	L5
SG_TMS_FR	Front right TMS sensors (external/cabin temperature, pressure, etc.) and actuators signals	Analog / Digital IO	64	VDC	L5
SG_ VISION_ L3_FR	Front right below L3 vision sensors (lidar, radar, etc.) signals	Radar Lidar	128	MDC	C1
SG_ VISION_ L5_FR	Front right L3 or above vision sensors (camera, hd maps, etc.) signals	Camera	196	MDC	C1
SG_ Wheel_ RR	Rear right wheel sensors (speed, pressure, etc.) and actuators signals	Analog / Digital IO	64	VDC	C2, SDNC, C5
SG_ VISION_ L3_RR	Rear right below L3 vision sensors (lidar, radar, etc.) signals	Radar Lidar	128	MDC	L6
SG_ VISION_ L5_RR	Rear right L3 or above vision sensors (camera, hd maps, etc.) signals	Camera	196	MDC	L6
SG_ Wheel_ RL	Rear left wheel sensors (speed, pressure, etc.) and actuators signals	Analog / Digital IO	64	VDC	C3, SDNC, C5
SG_BMS_RL	Rear left BMS sensors (temperature, charge level, etc.) and actuators signals	Analog / Digital IO	64	VDC	C3, SDNC, C5
SG_ VISION_ L3_RL	Rear left below L3 vision sensors (lidar, radar, etc.) signals	Radar Lidar	128	MDC	C3
SG_ VISION_ L5_RL	Rear left L3 or above vision sensors (camera, hd maps, etc.) signals	Camera	196	MDC	C3
SG_ Wheel_ FL	Front left wheel sensors (speed, pressure, etc.) and actuators signals	Analog / Digital IO	64	VDC	L4, VIU1, L5
SG_PT_ FL	Front left powertrain sensors (speed, pressure, etc.) and actuators signals	Analog / Digital IO	64	VDC	L4, VIU1, L5
SG_EPS_FL	Front left EPS sensors (angle, , torque, pressure, etc.) and actuators signals	Analog / Digital IO	64	VDC	L4, VIU1, L5
SG_ VISION_ L3_FL	Front left below L3 vision sensors (lidar, radar, etc.) signals	Radar Lidar	128	MDC	C4
SG_VISION_ L5_FL	Front left L3 or above vision sensors (camera, hd maps, etc.) signals	Camera	196	MDC	C4

TABLE 4. Examples of item functions with no safety goals (class 1).

Item Function	(Safety) goal	FTTI	ASIL	Safe state
Multimedia	SG11: A loss of multimedia function shall be avoided	-	QM	-
Radio	SG12: An unintended increase of audio volume shall be avoided	-	QM	-
Interior light	SG13: An unintended activation of interior lights shall be avoided	-	QM	-
Exterior light	SG14: An unintended deactivation of position lights shall be avoided	-	QM	-
Airbag	SG15: The related airbag shall be inflated in case of a crash	-	QM	-
Horn	SG16: The horn shall sound in case of driver request	-	QM	-
Engine control	SG17: A loss of traction torque shall be avoided	-	QM	-
Gearbox	SG18: A loss of torque transmission shall be avoided	-	QM	-
Anti-theft system	SG19: Missing theft alarm in case of theft/stealing activities shall be avoided	-	QM	-

TABLE 5. Examples of item functions with only fail-safe safety goals (class 2).

Item Function	(Safety) goal	FTTI	ASIL	Safe state
Window lifter	SG21: An unintended movement of window lifter shall be avoided	100 ms	A	Stop movement OR operate like intended
Engine control in manual or automated driving mode	SG22: An unintended acceleration shall be avoided	500 ms	B	No propulsion torque OR operate like intended
Steering in manual driving mode	SG23: An unintended or wrong steering torque shall be avoided	20 ms	D	No steering assist OR operate like intended
Steering in manual driving mode	SG24: A blocked steering shall be avoided	100 ms	D	At max. mechanical steering force OR operate like intended
Airbag	SG25: An airbag shall not be inflated in the absence of crash	~5ms for side airbags and ~20 ms for front	D	-

receiver ECUs require valid input data. Table 7 shows some concrete safety goal examples all defined considering the top-level goal "collision avoidance".

V. SDN CONTROLLER AND NODE SUPERVISOR

In this section we introduce the micro-architecture of the main component of the SDNC: the node supervisor (NS). The NS, located within each VIU (Fig. 6), is in charge of monitoring the status of the VIU and exchanging information with the SDNC.

A. NODE SUPERVISOR ARCHITECTURE OVERVIEW

The high level architecture of the node supervisor is depicted in Fig. 8. As seen in the figure, the NS communicates periodically with the SDNC allowing both the SDNC and the VIU to report their own status, and to receive information about the other components. Internally, the NS hosts a Finite State Machine (FSM) that takes as input the information from the SDNC and from an internal error detection unit and, based on this, decides the operation mode of the node and determines thus the internal configuration of the node.

TABLE 6. Examples of item functions with safety goals requiring availability, with availability addressed at actuator side (class 3).

Item Function	(Safety) goal	FTTI	ASIL	Safe state
Windshield wiper	SG31: A loss of wiping when required shall be avoided	800 ms	A	Operate as intended or keep on wiping
Brake light	SG32: A loss of brake light shall be avoided	300 ms	B	Operate as intended or turn on the brake light
Low beam	SG33: A loss of low beam shall be avoided	300 ms	B	Operate as intended or turn on low beam
Steering in manual driving mode	SG34: A sudden loss of steering assist shall be avoided	1 s	B	Operate as intended or torque ramp down

TABLE 7. Examples of item functions with safety goals requiring availability, with availability addressed at multiple systems of the item (class 4).

Item Function	(Safety) goal	FTTI	ASIL	Safe state
Steering in automated driving mode	SG41: A loss of steering torque shall be avoided	100 ms	D	Provide function or emergency operation
Engine control in manual or automated driving mode	SG42: An unintended steering torque shall be avoided	20 ms	D	Provide function or emergency operation
Braking/vehicle deceleration in automated driving mode	SG43: An unintended braking torque shall be avoided	150 ms	D	Provide function or emergency operation
Braking/vehicle deceleration in automated driving mode	SG44: A too low vehicle deceleration shall be avoided	various	D	Provide function or emergency operation
Acceleration in automated driving mode	SG45: A loss of propulsion shall be avoided	n/a	QM	n/a

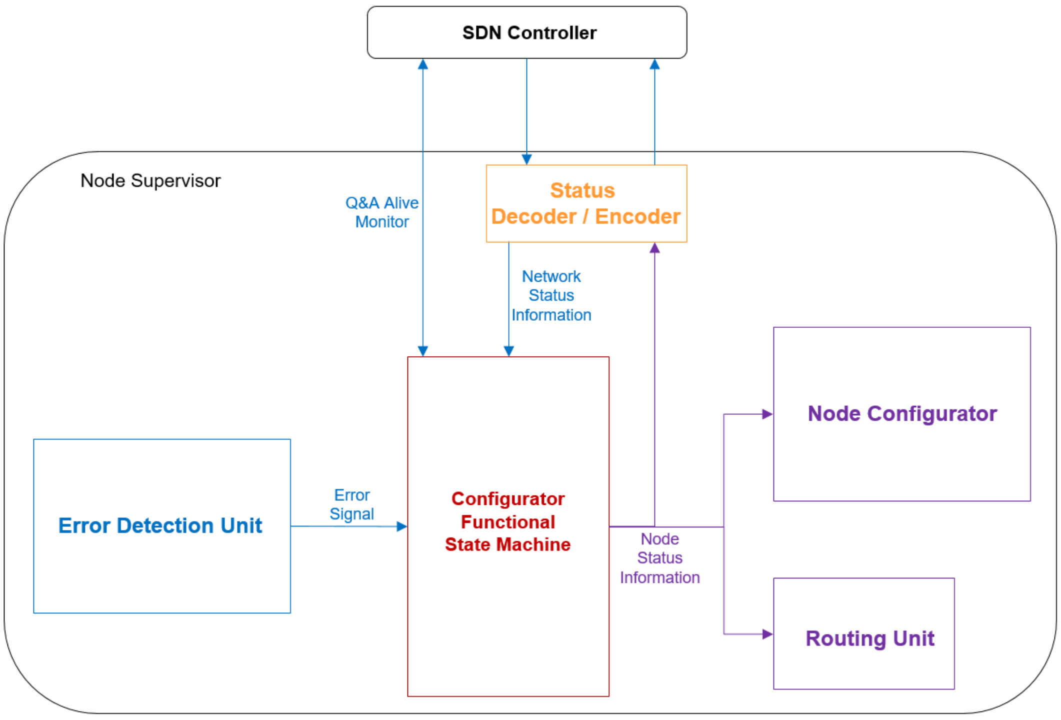

FIGURE 8. Node supervisor block diagram.

B. NODE SUPERVISOR COMPONENTS DESCRIPTION

As we can see in the NS block diagram (Fig. 8), the following modules are necessary to fulfil the functions of the NS.

- **Error Detection Unit:** Contains monitoring and error detection mechanisms for detection of:
  - Local node HW (permanent) failures
  - Link failure (for links physically connected to the node)
  - Node congestion
  - Queue overflow

This block can consist of a combination of already investigated technologies such as:

- Network Health Monitoring [74]
- Load Management Layer [45]

The output of this block is an error signal which will be transmitted to the Configurator Functional State Machine

This error signal is transmitted using an error signaling protocol that distinguishes between three classes of errors: High, medium and low criticality.

- **Node Configurator:** Static configuration of 8 signal groups in 6 different functional modes. In each mode, the node will be configured with a specific configuration profile implementing different network management algorithms such as:
  - Credit Based Shaper [38]
  - Time Aware Shaper [37]
  - Hierarchical Link sharing [75]
  - Active Queue Management [76]
  - Low Latency Low Loss Scalable Throughput (L4S) [77]

- ◦ Combination of the above-mentioned configuration profiles

The frames are gathered into signal groups of different priorities and each signal group is associated with one profile.

- **Routing Unit**: This block contains the different (static) routing tables corresponding to each state of the configuration FSM.
- **Status encoder/decoder**: This module enables to exchange status information between the SDNC and the NS. It encodes/decodes the status information into the selected protocol for this communication. This communication protocol shall provide some safety mechanism to ensure the validity of data, such as a CRC.
- **Configurator Functional State Machine:** The FSM determines the status of the node based on the inputs received and generates the error signal that enables the other NS modules to configure the node in the corresponding operation mode. Next, we describe the different states considered within the configurator FSM.
  - ◦ **S0 ➔ Nominal Mode**
    - ▪ Default state
    - ▪ Driver is in the loop
    - ▪ Sensors‘ data are used for L2 and below ADAS functions (e.g. AEB, EPS)
  - ◦ **S1 ➔ Highway mode**
    - ▪ Driver is out of the loop
    - ▪ Sensor data are used for L3 and above AD function (e.g. Automated Lane Keeping Systems, Adaptive Cruise Control, highway pilot)
  - ◦ **S2 ➔ Parking mode**
    - ▪ Driver is out of the loop
    - ▪ Sensors‘ data used for park space search, autonomous parking and autonomous parking exit
  - ◦ **S3 ➔ Error State 0**
    - ▪ Degradated mode 0
    - ▪ Bandwidth reduction
    - ▪ Link or node affecting first priority applications faulty or not available resulting in a bandwidth reduction (after compensation with available bandwidth of second priority application) for first priority application
    - ▪ Link or node affecting second priority applications not available which leads to a bandwidth reduction for first priority applications (for compensation reasons)
  - ◦ **S4 Error State 1**
    - ▪ Degradated mode 1
    - ▪ Bandwidth reduction
    - ▪ Link or node affecting first priority applications faulty or not available resulting in a bandwidth reduction (after compensation with available bandwidth of second priority application) for first priority application

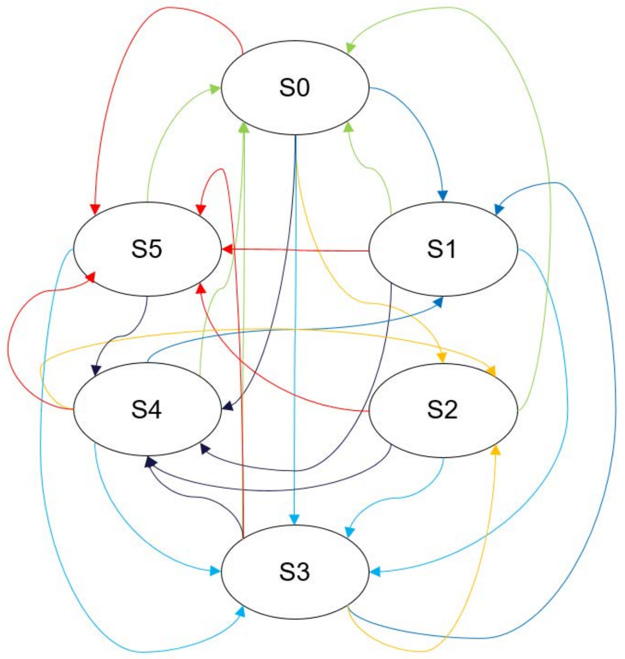

FIGURE 9. Configurator finite state machine.

    - ▪ Link or node affecting second priority applications not available which leads to a bandwidth reduction for first priority applications (for compensation reasons)
  - ◦ **S5 Emergency mode**
    - ▪ SDNC not responsive or specific node in the system not functional
    - ▪ If another fault appears, the system will not be able to recover from it without losing significantly in availability for first priority application

A state diagram describing the state transitions of the Configurator Functional State Machine is shown in Fig. 9, and the conditions that trigger the different transitions are exposed in Table 8.

In general, transitions from one state to another are triggered either by a specific condition (error detected, manual setting by user or environmental inputs) or by a specific order from the SDNC. The latter has the highest priority, i.e. when an instruction from SDNC indicates transition to a specific state, this transition is performed regardless of the other inputs.

In the case of detecting an error, the FSM selects the state for transition based on the error criticality. Once the error is recovered, the FSM goes back to the previous state. However, there are some exceptions. For instance, after a high criticality error, in case of no other errors present, only nominal mode (S0) is allowed, i.e. the system should not go from a high criticality error state to highway or parking mode.

It is important to note that the ‘‘ErrorRecovery’’ signal is only asserted when no errors are present. This means that after a high criticality error, if there is a medium or low criticality error in the system the new state will be S3 or S4 respectively, but never S0. Similarly, a medium criticality error will go back to S0, S1 or S2 only if all errors are cleared. Otherwise it would move to S3.

In normal operation, the FSM moves between S0, S1 and S2 according to the user inputs and detected conditions when AD features are enabled. Finally, it is also important to note that there are some forbidden transitions, such as directly changing from S2 (parking mode) to S1 (highway mode) or vice versa.

VI. NEW GENERATION AUTOMOTIVE GATEWAY CONTROLLER

In this section, we introduce the novel ''Elastic Gateway (eGW)'' architecture, as a new generation automotive gateway controller. In this context, the role of the eGW is the one of a VIU. We introduce the eGW as a necessary step for the understanding of the following sections, where we develop the functional safety concept of a VIU according to the safety requirements already derived, targeting an eGW device. We start by introducing the high-level functional requirements of eGW. Then, we describe the high-level architecture of the eGW and, afterwards, we introduce the main building blocks and the strategy followed across them. Finally, we deep dive in the FuSa concept for the novel eGW architecture.

A. VIU FUNCTIONAL REQUIREMENTS

The proposed system architecture for the eGW aims at covering the wide variety of requirements that are applicable to IVNs, and tries to remain as flexible as possible in order to integrate future requirements. A summary of the technologies and functionalities targeted by eGW is given in Table 9.

B. VIU ARCHITECTURE OVERVIEW

The architecture is composed of a set of HW IPCores that conform a custom HDL library defined for GW design. Similarly to the previous example illustrated in Figure 3, the architecture of the eGW controller is shown in Fig. 10. The eGW controller keeps as main blocks the host CPU, memories, transceivers (physical layer) for communications, SBC for power management and SPC for control and power supply of the zonal loads.

Moving forward to the functional details, Fig. 11 presents the main building blocks of the eGW SoC device. This SoC plays the real role of brain of the eGW controller shown in Fig. 10. As seen in the figure, eGW is composed of an ingress stage, a processing stage and an egress stage. The ingress stage consists of a normalization module that accommodates the ingress frame to the internal datapath and generates the initial metadata required for its processing. The processing stage is composed of one single Match & Action stage that allows to perform the required processing. Finally, the egress stage handles the scheduling of frames and selects when to send each frame to the egress ports. As seen in the figure, there are two loopback paths, allowing to send frames back after processing stage or egress stage. This allows to perform further processing of frames when required, such as recursive processing algorithms, providing the required flexibility and scalability to accomplish the goals of the architecture in terms of functionalities. Furthermore, it enables the reuse of HW resources, allowing to optimize the HW cost.

All of the IPCores are connected to the system CPU which is in charge of configuring the required parameters for the operation of each of them. For example, it can configure certain filtering rules to be implemented in the match or filtering stage. This configuration does not need to be static, i.e. it can change during the operation of the system, even at run-time without interrupting the operation of the system. The CPU can also intervene in the processing if required by the HW-SW partitioning of each application. Additionally, the main functional blocks receive the information of system time that allows to have a detailed control of the time involved in the processing of each frame.

C. VIU COMPONENTS DESCRIPTION

The architecture proposed for the high level HW IPCores follows the SDN approach. Each HW IPCore is composed of a control plane and a data plane, where the data plane is configured by the control plane. The main characteristics and functionalities of each of the IPCores are exposed below.

- **Frame Normalizer**: This block is intended to provide an abstraction layer able to handle any incoming protocol and generate an internal instruction frame format, common for all protocols, transporting the metadata required for internal processing. To do so, a new normalization concept is introduced, where metadata is transported in parallel with the data frame, allowing the control plane to decide what action should be taken for each frame. The definition of this block, and the SDN approach followed within the architecture, are described in [78].
- **Filtering (Match)**: This block allows to perform filtering on ingress frames based on a set of rules applied to the frames' header. It is composed of a Content Addressable Memory that provides parallel processing of all the rules in order to detect if the incoming frames match any of the predefined rules. It implements also a regular expression (RegEx) search engine applied to the frames' payload.
- **Processing (Action):** This block performs any processing or action to the frame that is required covering OSI layers 2 to 7 acting not only on headers, but also on the payload if needed. The Action or Processing block is composed of a stack of parallel tasks that can operate in parallel and over different frames at the same time. For this, the processing controller handles the interconnection between the intermediate queues and the processing stage allowing for interconnecting any queue with any processing task, providing thus maximum flexibility in terms of parallel processing. An example of how a safety related feature (IEEE802.1CB) is deployed within the eGW architecture is given in [79]. The output of the pro-

TABLE 8. Configurator finite state machine transitions.

FROM / TO	S0	S1	S2	S3	S4	S5
S0	-	HighwayEntry=T AND ManualHighwayModeOn=T	ManualParkingModeOn=T AND ParkingCompleted=T AND VehicleSpeed > ParkingSpeedLimit	Error=T and ErrorCriticality=X or SDNC_to_SX_order (X: 3-5) X=3 Low Criticality X=4 Medium Criticality X= 5 High Criticality		
S1	HighwayExit=T OR ManualHighwayModeOff=T	-	Not allowed			
S2	ManualParkingModeOff=T OR ParkingCompleted=T OR VehicleSpeed > ParkingSpeedLimit	Not allowed	-			
S3	{ErrorRecovery=T AND PreviousState = S0} OR SDNC_to_S0_order	ErrorRecovery=T AND PreviousState = S1 OR SDNC_to_S1_order	ErrorRecovery=T AND PreviousState = S2 OR SDNC_to_S2_order			
S4	{ErrorRecovery=T AND PreviousState = S0} OR SDNC_to_S0_order	ErrorRecovery=T AND PreviousState = S2 OR SDNC_to_S2_order	ErrorRecovery=T AND PreviousState = S2 OR SDNC_to_S2_order			
S5	{ErrorRecovery=T AND PreviousState = S0} OR SDNC_to_S0_order	Not allowed	Not allowed			

*T: TRUE, F: FALSE

TABLE 9. Functionalities and technologies targeted by elastic gateway.

Network Technologies	Packet Processing	Protocol Tunneling	Network Processing	TSN Standards
Ethernet 100/1000Base-T1	Frame (de-) normalization	IEEE 1722	Routing & Forwarding	IEEE 802.1AS
10Base-T1S	Error tracking	SAE J2602	Multicast	IEEE 802.1CB
CAN 2.0	CRC/CS	SAE J1939	Broadcast	IEEE 802.1Qci
CAN-FD	Priority management	ISO 17987	Cut-through	IEEE 802.1Qbu
CAN XL	Application level processing		Store & Forward	IEEE 802.1Qbv
FlexRay	Frame manipulation and transformation		Congestion-aware routing	IEEE 802.1Qav
LIN	Frames generation			IEEE 802.1Qch
				IEEE 802.1Qcr

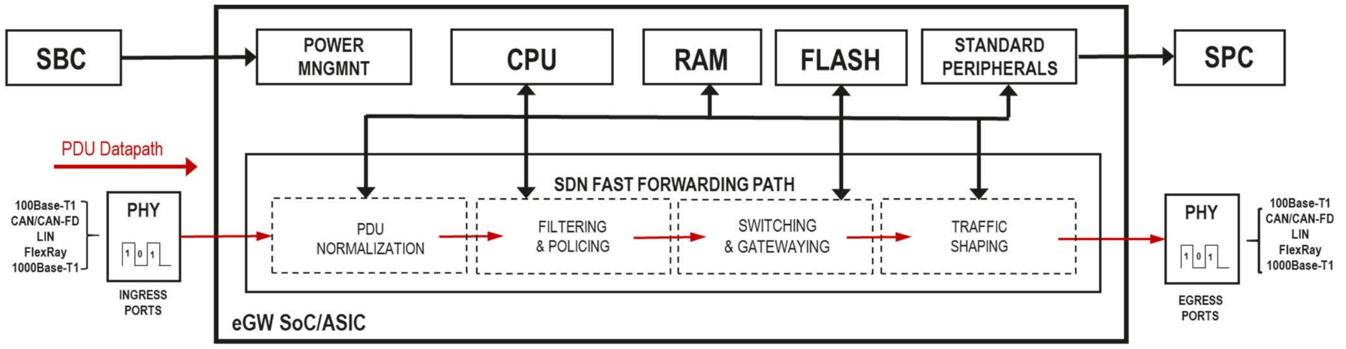

FIGURE 10. Block diagram of the eGW controller or VIU.

cessing stage can go to the egress stage, or back through the loopback path, if further processing is required. With this architecture, the pipelining or scheduling of tasks is achieved by looping the frame through the available loopback paths in a completely easy and flexible way. As shown in Fig. 11, one loopback path is directly going to the intermediate queues in order to go to the action module again, and the other one is connected to the ingress queues, allowing for performing a complete new M&A processing over the frame. These two loopback paths provide full flexibility to adapt to the needs of the application processing and provide the minimum latency possible, by allowing for connecting any path to any required task and handling the priorities. The loopback path for the processing stage is described in detail in [80], together with an example of application where it is beneficial. This stage hosts the previously described Node Supervisor, as one of the available tasks implemented in HW, as seen in Fig. 11.

- **Traffic Shaping:** This block provides control over the output scheduling of the Gateway ports according to the selected traffic shapers (Time Aware Shaper, Credit Based Shaper, Asynchronous Traffic Shaping, etc.). The architecture of the traffic shaping engine is described in [81]. As seen in blue in Fig. 11, this stage can also be extended with further processing by means of another loopback path. The reasoning behind this loopback path for the traffic shaping stage together with some use cases where it is beneficial are introduced in [82].

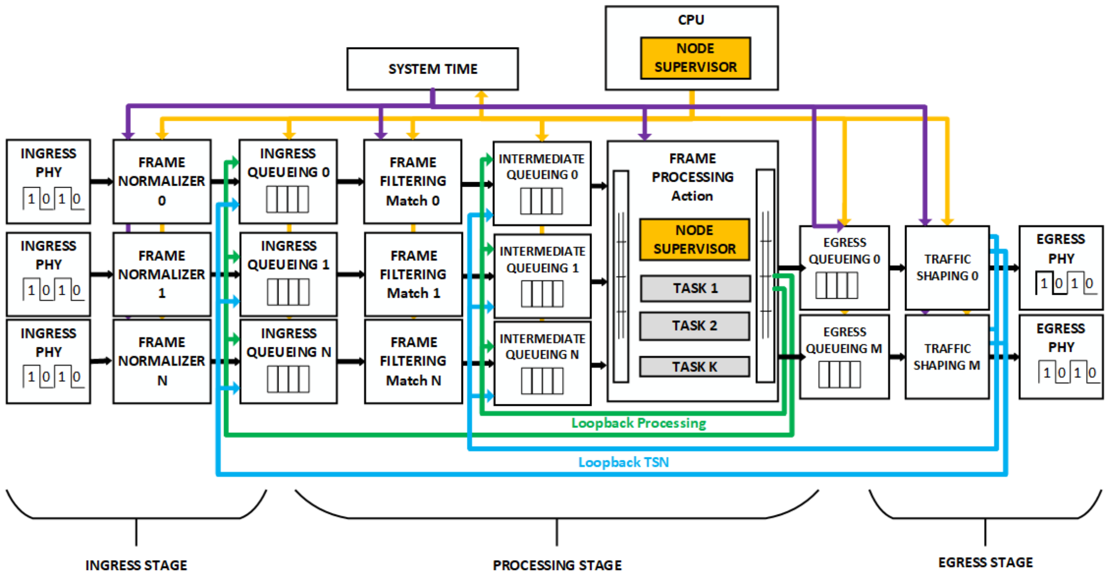

FIGURE 11. eGW SoC high level architecture.

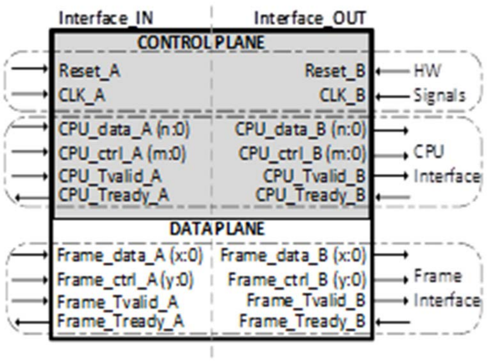

FIGURE 12. Common interface between blocks.

- **Queuing:** This provides the required buffering to accommodate traffic going from input to output ports without losses and effectively handle traffic congestions.

The architecture and strategy followed in the queueing engine, are described in [83].

All IPCores share the same I/O interface allowing to flexibly interconnect them. This interconnection flexibility is key in accomplishing the scalability and flexibility objectives of the design, allowing scalability of the GW as a product, from entry level to premium versions, by selecting which configuration is used for each of the HW IPCores in each specific product implementation.

The interface used in order to interconnect all the IPCores is depicted in Fig. 12. As seen in the figure, there is a host CPU bus that has access to I/O interfaces of the IPCores, representing the control plane communication within the SDN approach followed in the architecture. This CPU bus consists on a data line and a control line: (i) the data line carries the information shared between the CPU and the IPCores; (ii) the control bus supplements the data line information providing further commands that encode specific functionalities depending on the HW block. Apart from the CPU bus, the common interface provides also a frame data bus that, alike the CPU bus, consists on a data line that carries the frame information and a control line that provides commands to support the frame processing in the different stages. This communication bus between the CPU and the IPCores allows to deploy several safety relevant mechanisms, monitoring functions and to have redundancy on certain applications.

D. VIU FUNCTIONAL SAFETY CONCEPT

In this section, we derive the requirements (functional/non-functional, safety/non-safety related) allocated to a VIU as one of the elements of the NGIVN. These requirements are based on Stakeholder Requirements (section IV.A) and Safety Goals (section IV.B). They are organized into three sections:

1. A subset of Requirements for QM Functionalities – no impact from Safety Goals
2. Functional Safety Requirements – derived from Safety Goals
3. Non-Functional Requirements – requirements that are non-functional.

1) REQUIREMENTS FOR QM FUNCTIONALITIES

The listed requirement are derived from Class 1 Safety Goals. Note: This section is not supposed to be complete, as QM requirements are not the document's focus

[FUN_IVN_0001] Transmission of Non-Safety-Related Data

2) FUNCTIONAL SAFETY REQUIREMENTS

The FSRs allocated to VIU are divided into three groups:

1. Requirements related to gatewaying

Description:	The VIU shall transmit defined non-safety-related data by network communication.
Rationale:	
Use Case:	
Dependencies:	
Supporting Material:	This requirement defines the functionality for forwarding data from a defined source to a destination, while transmitted data is non-safety-related. Any fault of this communication ends in a QM classified scenario (HARA).
ASIL:	QM
Safe State:	N/A
FTTI:	N/A
Validation Criteria:	N/A
Parent ID:	STR_IVN_0001, SG_1*

2. Requirements Related to Hosting of Applications and Taking Over
3. Requirements Related to Fault Tolerance

a: FSRs RELATED TO GATEWAYING

The listed FSRs are derived from Class 2, 3 and 4 Safety Goals. The next sub-sections describe each FSR group in detail.

[FSR_IVN_0002] Fail-Safe Transmission of Safety-Related Data without Data Manipulation

Description:	The VIU shall transmit defined fail-safe PDUs by network communication without manipulating received safety-related PDUs.
Rationale:	
Use Case:	
Dependencies:	FSR_IVN_0020
Supporting Material:	Safety-related PDU means safety-related application payload (e.g., speed, lidar data) plus E2E protection header. Note that network headers (e.g., TCP/IP stack headers) or related protection mechanisms like CAN CRC or TCP CRC are not part of the safety-related PDU, and therefore they can be replaced or changed. As a result, as long as the VIU does not generate a valid E2E signature for new data (but only forwards the safety-related PDUs), the receiver can detect communication faults on the line between the safety-related sender and the receiver itself. Because the data is fail-safe, the error reaction error reporting or no transmission can be applied.
ASIL:	Up to ASIL D
Safe State:	1. Correct transmission of non-manipulated data 2. No transmission of data (in case of error) or 3. Error reported to the receiver (in case of error)
FTTI:	The FTTI may differ depending on the reaction (safe state) of the VIU: 1. No transmission: <10ms ... 500ms> 2. Error reported to the receiver: <10ms ... 500ms>
Validation Criteria:	OEM shall validate the FTTI at the vehicle level. Depending on a specific concept, specific actuators, or specific data, it can be different. 21
Parent ID:	STR_IVN_0001, STR_IVN_0002, SG_2*, SG_3*

[FSR_IVN_0004] Fault-Tolerant Transmission of Safety-Related Data without Data Manipulation

Description:	The VIU shall transmit defined independent multiple fault-tolerant PDUs by network communication without manipulating received safety-related PDUs.
Rationale:	
Use Case:	
Dependencies:	
Supporting Material:	In case the transmission is done over a subset of the channels, and the receiver expects that all channels are provided, then depending on a specific OEM solution, two options are possible: 1. The receiver notices that one of the channels does not transmit data, e.g., there is a transmission timeout, and reacts correspondingly. In other words, there was no explicit error reporting by IVN, just stopping sending over a given channel. 2. The receiver notices that one of the channels does not transmit data. Moreover, the correctly functioning channels provide additional error information about the failure of that channel.
ASIL:	Up to ASIL D
Safe State:	1. Correct transmission of non-manipulated data 2. No transmission of data (in case of error) or 3. Error reported to the receiver (in case of error)
FTTI:	The FTTI may differ depending on the reaction (safe state) of the VIU: 1. No transmission: <10ms ... 500ms> 2. Error reported to the receiver: <10ms ... 500ms>
Validation Criteria:	OEM shall validate the FTTI at the vehicle level. Depending on a specific concept, specific actuators, or specific data, it can be different.
Parent ID:	STR_IVN_0001, STR_IVN_0002, SG_4*

[FSR_IVN_0005] Fail-Safe Transmission of Safety-Related Data with Data Manipulation

Description:	The VIU shall transmit defined fail-safe PDUs by network communication with manipulating received safety-related PDUs.
Rationale:	
Use Case:	
Dependencies:	
Supporting Material:	
ASIL:	Up to ASIL D
Safe State:	1. Correct transmission of non-manipulated data 2. No transmission of data (in case of error) or 3. Error reported to the receiver (in case of error)
FTTI:	The FTTI may differ depending on the reaction (safe state) of the VIU: 1. No transmission: <10ms ... 500ms> 2. Error reported to the receiver: <10ms ... 500ms>
Validation Criteria:	OEM shall validate the FTTI at the vehicle level. Depending on a specific concept, specific actuators, or specific data, it can be different.
Parent ID:	STR_IVN_0001, STR_IVN_0002, SG_4*

[FSR_IVN_0006] Fault-Tolerant Transmission of Safety-Related Data with Data Manipulation

Description:	The VIU shall provide safety-related PDUs generated by IVN, combined from received QM or safety-related PDUs, or manipulated received data by QM or safety-related PDUs
Rationale:	
Use Case:	Combine four E2E-protected CAN messages received from four ABS sensors into one E2E-protected Ethernet message, delivered to SAE Level 5 HPCs.
Dependencies:	
Supporting Material:	This requirement defines the functionality for data communicated by VIU generated by VIU or is a 'new' combination of received information by VIU. A change in the payload is needed, e.g., alive counter, payload CRC. VIU is a data source like a sensor or a sender ECU.
ASIL:	Up to ASIL D
Safe State:	3. Correct combination and transmission of data 4. Correct combination and transmission over a subset of redundant channels of non-manipulated data, with or without error reporting (in case of error) or 5. Correct combination and transmission over a backup channel(s) of non-manipulated data, with or without error reporting (in case of error)
FTTI:	The FTTI may differ depending on the reaction (safe state) of the VIU: 6. No transmission: <10ms ... 500ms> 7. Error reported to the receiver: <10ms ... 500ms>
Validation Criteria:	OEM shall validate the FTTI at the vehicle level. Depending on a specific concept, specific actuators, or specific data, it can be different.
Parent ID:	STR_IVN_0001, STR_IVN_0002, SG_2*, SG_3*, SG_4*

[FSR_IVN_0020] Non-Generation of Safety-Related Data

Description:	For defined safety-related data that are supposed to be transferred by VIU without manipulation, VIU shall never generate a valid E2E-protected PDUs – VIU is only allowed to take the original PDUs and forward them to the destination.
Rationale:	
Use Case:	FSR_IVN_0002, FSR_IVN_0004
Dependencies:	
Supporting Material:	In case this forwarding of non-manipulated data is realized by a QM router application, this could be resolved by ensuring that this application has no access to the E2E algorithms, so it cannot generate a valid E2E-protected message.
ASIL:	Up to ASIL D
Safe State:	For fail-safe data: 1. Non-Generation of Safety-Related data
FTTI:	0 ms (by design)
Validation Criteria:	OEM shall validate the FTTI at the vehicle level. Depending on a specific concept, specific actuators, or specific data, it can be different.
Parent ID:	STR_IVN_0001, STR_IVN_0002, SG_2*, SG_3*, SG_4*

b: FSRs RELATED TO HOSTING OF APPLICATIONS AND TAKING OVER

[FSR_IVN_0011] Taking over the Application Functions of Faulty HPCs

Description:	In case of failure of one HPC, the VIU shall execute defined safety-related application software (from OEM) and deliver/route the outputs of this software to defined receiver ECUs.
Rationale:	
Use Case:	A replica of the HPC application function is deployed on the VIU. It is either cold standby or hot standby. Moreover, a simpler functionality is deployed on the VIU. In case of an HPC application failure, the VIU output gets active.
Dependencies:	
Supporting Material:	The execution of the application on a VIU is considered fail-safe. In combination with a fail-safe execution on an independent HPC, fault tolerance is to be achieved. Consequently, HPC and the corresponding VIU needs to be sufficiently independent. The independence needs to be ensured at the vehicle/item level. This requirement results in a requirement for the safety manual. The software is considered up to ASIL D. Moreover, it is assumed that this software uses only communication interfaces (e.g., Ethernet) and SoC features (e.g., GPU) but no I/Os (like in classic AUTOSAR).
ASIL:	Up to ASIL D
Safe State:	1. Correct execution of migrated application software and provision of outputs 2. Provision of error outputs 3. Provision of no outputs
FTTI:	The FTTI may differ depending on the reaction (safe state) of the VIU: 1. No transmission: <10ms ... 500ms> 2. Error reported to the receiver: <10ms ... 500ms>
Validation Criteria:	OEM shall validate the FTTI at the vehicle level. Depending on a specific concept, specific actuators, or specific data, it can be different.
Parent ID:	STR_IVN_0003, SG_4*

[FSR_IVN_0028] Hosting of Applications

Description:	The VIU shall execute defined safety-related fail-safe and fault-tolerant software. The fault-tolerant software shall be executed on independent VIUs or IVN elements.
Rationale:	
Use Case:	Wheel speed sensors provide a PWM signal over Ethernet. VIUs convert this signal to speed in [m/s].
Dependencies:	
Supporting Material:	
ASIL:	Up to ASIL D
Safe State:	1. Correct execution of application and provision of outputs 2. 2. For fail-safe applications, non-execution or reset of applications 3. 3. For fault-tolerant applications, correct provision of outputs of a subset of redundant applications, with or without error reporting or 4. 4. For fault-tolerant applications, correct provision of outputs from a backup application, with or without error reporting
FTTI:	The FTTI may differ depending on the reaction (safe state) of the VIU: 1. No transmission: <10ms ... 500ms> 2. Error reported to the receiver: <10ms ... 500ms>
Validation Criteria:	OEM shall validate the FTTI at the vehicle level. Depending on a specific concept, specific actuators, or specific data, it can be different.
Parent ID:	STR_IVN_0004, SG_1*, SG_2*, SG_3*, SG_4*

c: FSRs RELATED TO FAULT TOLERANCE

[FSR_IVN_0007] Network Quality of Service

Description:	The VIU shall provide defined quality of service mechanisms (i.e., traffic prioritization and resource reservation).
Rationale:	
Use Case:	
Dependencies:	
Supporting Material:	This requirement is needed for SG_4* to achieve high availability. It is also nice to have for other SGs.
ASIL:	Up to ASIL D
Safe State:	1. Correct operation of quality-of-service mechanisms (usage by the upper layer / by application + correct provision by the stack) 2. Correct prioritization of communication in case of bus overload.
FTTI:	The FTTI may differ depending on the reaction (safe state) of the VIU: 1. No transmission: <10ms ... 500ms> 2. Error reported to the receiver: <10ms ... 500ms>
Validation Criteria:	OEM shall validate the FTTI at the vehicle level. Depending on a specific concept, specific actuators, or specific data, it can be different.
Parent ID:	STR_IVN_0001, STR_IVN_0002, SG_4*

[FSR_IVN_0008] Network Node Monitoring

Description:	The VIU shall support the following monitoring functions: 1. Link monitoring (between two nodes) 2. Node monitoring (e.g., if it is alive) 3. Network load monitoring 4. Protocol monitoring 5. Application monitoring
Rationale:	
Use Case:	It is detected that a node is faulty, and as a result, it is reset.
Dependencies:	FSR_IVN_0019: the monitoring channels need to be independent so that a fault-tolerant system can be built from fail-safe channels.
Supporting Material:	This requirement is needed for SG_4* to achieve high availability. It is also nice to have for other SGs. This requirement is one of the most important ones of the VIU. Example possible mechanisms could be: 1. Link monitoring: checking if the Ethernet link is active 2. Node monitoring: check if node sends regular ping messages or other alive messages over UDP 3. Network load monitoring: monitoring of the bandwidth of an Ethernet link 4. Protocol monitoring: timeout of TCP/IP 5. Application monitoring: sign-of-life from a specific application
ASIL:	Up to ASIL D
Safe State:	1. Correct operation of monitoring mechanisms (usage by the upper layer / by application + correct provision by the stack). 2. Detecting an error, reporting it, and triggering an error reaction
FTTI:	The FTTI may differ depending on the reaction (safe state) of the VIU: 1. Detection of failures and triggering and executing error reaction: <10ms ... 500ms>
Validation Criteria:	OEM shall validate the FTTI at the vehicle level. Depending on a specific concept, specific actuators, or specific data, it can be different.
Parent ID:	STR_IVN_0001, STR_IVN_0002, SG_4*

[FSR_IVN_0017] Redundancy

Description:	The VIU shall support a redundant IVN system implementation (i.e., a redundant fabric of VIUs), including: 1. Redundant VIUs in NGIVN (i.e., multiple servers) 2. Parallel links between two network nodes 3. Redundant communication transceivers within one network node 4. Redundant transmission paths/routes between network nodes (e.g., over different VIUs)
Rationale:	
Use Case:	The electronic power steering ECU is connected over two independent Ethernet connections to VIUs.
Dependencies:	
Supporting Material:	This requirement is needed for SG_4* to achieve high availability. It is also nice to have for other SGs. Redundancy is a fundament for building fault tolerance. Redundancy does not necessarily mean a hot standby or a parallel execution of identical functions; it could also be, e.g., cold standby.
ASIL:	Up to ASIL D
Safe State:	1. Correct redundant operation 2. Operation over a single channel or a backup channel
FTTI:	The FTTI may differ depending on the reaction (safe state) of the VIU: 1. Transmission over a subset of redundant channels : <10ms ... 500ms> 2. Transmission over a backup channel : <10ms ... 500ms>
Validation Criteria:	OEM shall validate the FTTI at the vehicle level. Depending on a specific concept, specific actuators, or specific data, it can be different.
Parent ID:	STR_IVN_0001 STR_IVN_0002, STR_IVN_0003, SG_4*

[FSR_IVN_0009] Error Handling and Recovery Mechanisms

Description:	The VIU shall support the implementation of defined network error handling and reconfiguration mechanisms based on the configuration, including: Changing of the transmission routes Restarting of VIUs Restarting of network links or network transceivers Restarting applications Error reporting to receiver network nodes (e.g., report to actuator ECU).
Rationale:	
Use Case:	The electronic power steering ECU is connected over two independent Ethernet connections to VIUs.
Dependencies:	
Supporting Material:	The following is a possible candidate for changing transmission routes: 1. Fast reroute This requirement is needed for SG_4 to achieve high availability. It is also nice to have for other SGs. The network reconfiguration is application-independent. However, it still needs to be configured within IVN and can be done by an VIU
ASIL:	Up to ASIL D
Safe State:	1. Correct operation of recovery mechanisms
FTTI:	The FTTI may differ depending on the reaction (safe state) of the VIU: 1. Detection of failures and triggering and executing error reaction: <10ms ... 500ms>
Validation Criteria:	OEM shall validate the FTTI at the vehicle level. Depending on a specific concept, specific actuators, or specific data, it can be different.
Parent ID:	STR_IVN_0001 STR_IVN_0002, STR_IVN_0003, STR_IVN_0004, SG_4*

[FSR_IVN_0018] Independence between HPCs and VIUs

Description:	For defined fault-tolerant functions, the VIU shall be independent of HPCs.
Rationale:	
Use Case:	HPC fails. The function is migrated to an independent VIU , which is not to fail simultaneously
Dependencies:	
Supporting Material:	
ASIL:	Up to ASIL D
Safe State:	1. Correct operation of VIU and HPC. 2. In error cases, correct operation of at least VIU or HPC.
FTTI:	The FTTI may differ depending on the reaction (safe state) of the VIU: 1. Operation of VIU : <10ms ... 500ms>
Validation Criteria:	OEM shall validate the FTTI at the vehicle level. Depending on a specific concept, specific actuators, or specific data, it can be different.
Parent ID:	STR_IVN_0001 STR_IVN_0002, STR_IVN_0003, STR_IVN_0004, SG_4*

[FSR_IVN_0010] Fault Tolerance of VIU

Description	Through the QoS (FSR_IVN_0007), monitoring (FSR_IVN_0008), redundancy (FSR_IVN_0017), independence (FSR_IVN_0018), and recovery and error handling mechanisms (FSR_IVN_0009), the VIU shall be able to provide a defined set of functions (those requiring fault tolerance) in case of: 1. Failure of one NGIVN element (i.e., an SoC/node) 2. Failure of one network link 3. Failure of one communication transceiver (of an SoC) 4. Failure of one transmission path/route between network nodes 5. Failure of one connection/application-level protocol.
Rationale:	
Use Case:	
Depend.:	
Supporting Material:	For example, if a fail-safe signal is lost (as it is single-channel, going through a single link), the VIU would detect it. Consequently, the VIU, network links, etc., will need to be independent. The independence needs to be ensured within NGIVN. This will also impact the environment, e. g. redundant power supplies to supply the independent VIUs. This is needed for SG_4 to achieve high availability. It is also nice to have for other SGs. It is not requested to have a fault tolerance for failures of multiple VIUs (although it is nice to have). This will result in a requirement for the safety manual asking, e.g., redundant power supplies.
ASIL:	Up to ASIL D
Safe State:	1. Correct transmission 2. Transmission over a subset of redundant channels, with or without error reporting (in case of error) or 3. Transmission over a backup channel(s), with or without error reporting (in case of error)
FTTI:	The FTTI may differ depending on the reaction (safe state) of the VIU: 4. Transmission over a subset of redundant channels : <10ms ... 500ms> 5. Transmission over a backup channel : <10ms ... 500ms>
Validation Criteria:	OEM shall validate the FTTI at the vehicle level. Depending on a specific concept, specific actuators, or specific data, it can be different.
Parent ID:	STR_IVN_0001 STR_IVN_0002, STR_IVN_0003, STR_IVN_0004, SG_4*

[FSR_IVN_0016] Detection of Partial Failure of Fault-Tolerant Transmission

Description:	In case the fault-tolerant communication is realized using two (or more) redundant channels of the NGIVN, the Receiver nodes, in this case the VIU, shall decide which input channels to use based on: 1. Reported errors by the channels or 2. Failure (unavailability) of one of the channels.
Rationale:	
Use Case:	
Dependencies:	
Supporting Material:	
ASIL:	Up to ASIL D
Safe State:	3. Correct usage of the received data from the communication channels (in case of no error) 4. Emergency operation or degraded operation based on data from the partially faulty communication channels (e.g., application of ramp-down by actuator ECU, execution of Dynamic Driving Task (DDT) fallback (e.g., from HPC or vehicle driver)
FTTI:	The FTTI may differ depending on the reaction (safe state) of the VIU: 1. Detection and initiation of the emergency or degraded operation (DDT fallback): <10ms ... 500ms>
Validation Criteria:	OEM shall validate the FTTI at the vehicle level. Depending on a specific concept, specific actuators, or specific data, it can be different.
Parent ID:	STR_IVN_0001 STR_IVN_0002, STR_IVN_0003, STR_IVN_0004

3) NON FUNCTIONAL REQUIREMENTS

[FSR_IVN_0029] Legal and Normative Requirements

Description:	The VIU shall fulfil applicable legal and normative requirements.
Rationale:	
Use Case:	
Dependencies:	
Supporting Material:	
ASIL:	Up to ASIL D
Safe State:	5. Correct usage of the received data from the communication channels (in case of no error) 6. Emergency operation or degraded operation based on data from the partially faulty communication channels (e.g., application of ramp-down by actuator ECU, execution of DDT fallback (e.g., from HPC or vehicle driver)
FTTI:	The FTTI may differ depending on the reaction (safe state) of the VIU: 2. Detection and initiation of the emergency or degraded operation (DDT fallback): <10ms ... 500ms>
Validation Criteria:	OEM shall validate the FTTI at the vehicle level. Depending on a specific concept, specific actuators, or specific data, it can be different.
Parent ID:	STR_IVN_0001 STR_IVN_0002, STR_IVN_0003, STR_IVN_0004

[FSR_IVN_0030] Safety and Security Standards

Description:	The VIU shall be developed according to applicable safety and security standards.
Rationale:	
Use Case:	ISO 26262, ISO/SAE 21434
Dependencies:	
Supporting Material:	
ASIL:	
Safe State:	
FTTI:	
Validation Criteria:	
Parent ID:	STR_IVN_0001 STR_IVN_0002, STR_IVN_0003, STR_IVN_0004

[FSR_IVN_0031] Environmental Conditions, Lifetime and Quality Requirements

Description:	The VIU shall comply with applicable Environmental Conditions, Lifetime, and Quality Requirements.
Rationale:	
Use Case:	
Dependencies:	
Supporting Material:	
ASIL:	
Safe State:	
FTTI:	
Validation Criteria:	
Parent ID:	STR_IVN_0001 STR_IVN_0002, STR_IVN_0003, STR_IVN_0004

VII. DEPLOYMENT OF eGW FUNCTIONAL SAFETY CONCEPT

The deployment of the eGW functional safety concept is done by converting the FSRs listed in the previous sections into hardware, respectively software, safety requirements, thus, reaching a level of abstraction where specific safety capabilities of the eGW are allocated to a given hardware, respectively software, part of the eGW. At this stage, our next target is to match each hardware/software safety requirement with one or a combination of safety mechanisms (or safety measures).

To ease this matching process, we proceed to the identification, at NGIVN level, of the most relevant failure scenarios where the eGW is involved in. As introduced before, in the NGIVN architecture that we use as a reference, VDC is in charge of EPS and AEB functionalities, while MDC is in charge of ACC. This is enabled through the Service oriented Architecture used in this deployment.

A. IDENTIFICATION OF FAILURE SCENARIOS AND OF POSSIBLE FAIL-OPERATIONAL BACKUP PATHS

This activity consists of identifying the most relevant failure scenarios and to define a backup path for the frames affected by the failure. In this section we present a selection of link failure scenarios and their impact on the network dataflow:

TABLE 10. New dataflow for SGs connected to VIU1 after failure scenario #1

Sensor Group	Sensor Type	BW [Mbps]	Target	Path
SG_Wheel_FR	Analog / Digital IO	64	VDC	L5
SG_PT_FR	Analog / Digital IO	64	VDC	L5
SG_EPS_FR	Analog / Digital IO	64	VDC	L5
SG_TMS_FR	Analog / Digital IO	64	VDC	L5
SG_VISION_L3_FR	Radar Lidar	128	MDC	~~C1~~ L1, VIU2, C2
SG_VISION_L5_FR	Camera	196	MDC	~~C1~~ L1, VIU2, C2

TABLE 11. New dataflow for SGs connected to VIU1 after failure scenario #2

Sensor Group	Sensor Type	BW [Mbps]	Target	Path
SG_Wheel_RR	Analog / Digital IO	64	VDC	C2, SDNC, C5
SG_VISION_L3_RR	Radar Lidar	128	MDC	~~L6~~ C2
SG_VISION_L5_RR	Camera	196	MDC	~~L6~~ C2

- **Failure scenario #1** which consists of a permanent fault in the link C1, is shown in Fig. 13. Table 10 shows the consequences of failure scenario #1 on the network dataflow.
- **Failure scenario #2** consists of a permanent fault in the link L6 (Fig. 14). Table 11 shows the consequences of failure scenario #2 on the network dataflow.
- **Failure scenario #3** which consists of a permanent fault in the link C5 is depicted in Fig. 15. Table 12 shows the consequences of failure scenario #3 on the network dataflow.

B. IDENTIFICATION OF MATCHING SAFETY MECHANISMS

In order to cover these failure scenarios, we conceptualized a combination of safety mechanisms at NGIVN level. This combination of safety mechanisms has a direct impact on the eGW hardware and software architecture. It will contribute to the fulfillment of the eGW FSRs. *FSR_IVN_0010*, and through it *FSR_IVN_0007*, *FSR_IVN_0008*, *FSR_IVN_00017, FSR_IVN_0018, FSR_IVN_0009*, shall be particularly targeted by the selected combination of safety mechanisms.

The next sections describe the selected combination of NGIVN level safety mechanisms and its implementation at eGW level.

1) FAST FAIL-OPERATIONAL NGIVN RECONFIGURATION

The safety mechanism selected to cover the link failure scenarios presented in the previous section is the Fast Fail-Operational NGIVN Reconfiguration. This mechanism enables a protocol-based reconfiguration of network components. It is roughly based on the principle of the Software Defined Networking (SDN) but extends and modifies it for the purpose of safety-critical networks. Consequently, as in SDN-based networks, it provides stream-based (or flow-based) control with a dedicated protocol for reconfiguration

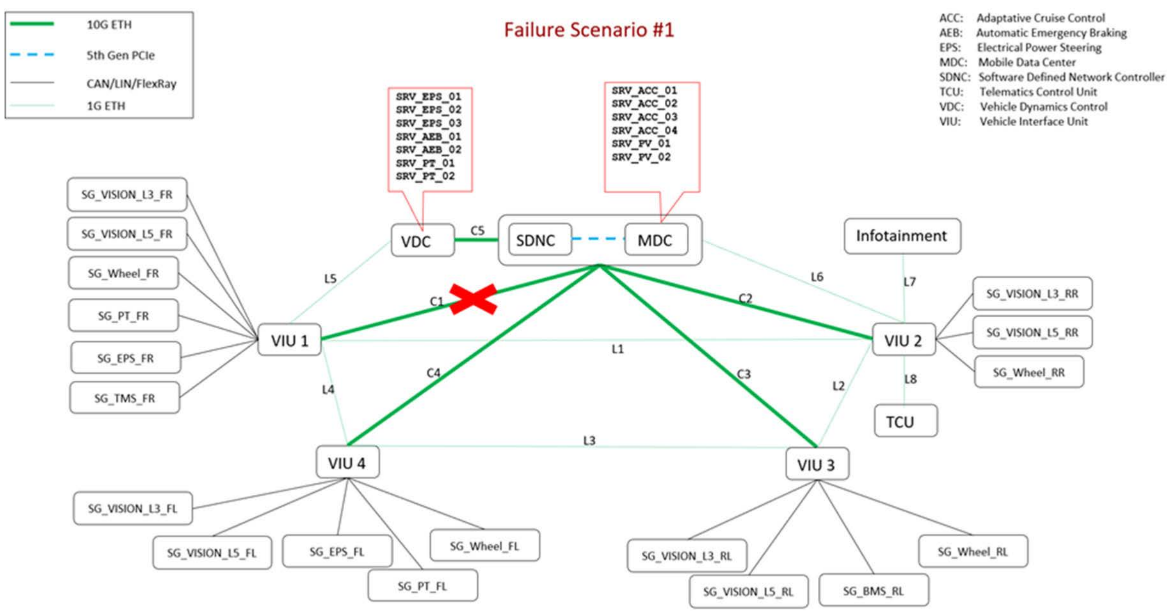

FIGURE 13. Failure scenario #1.

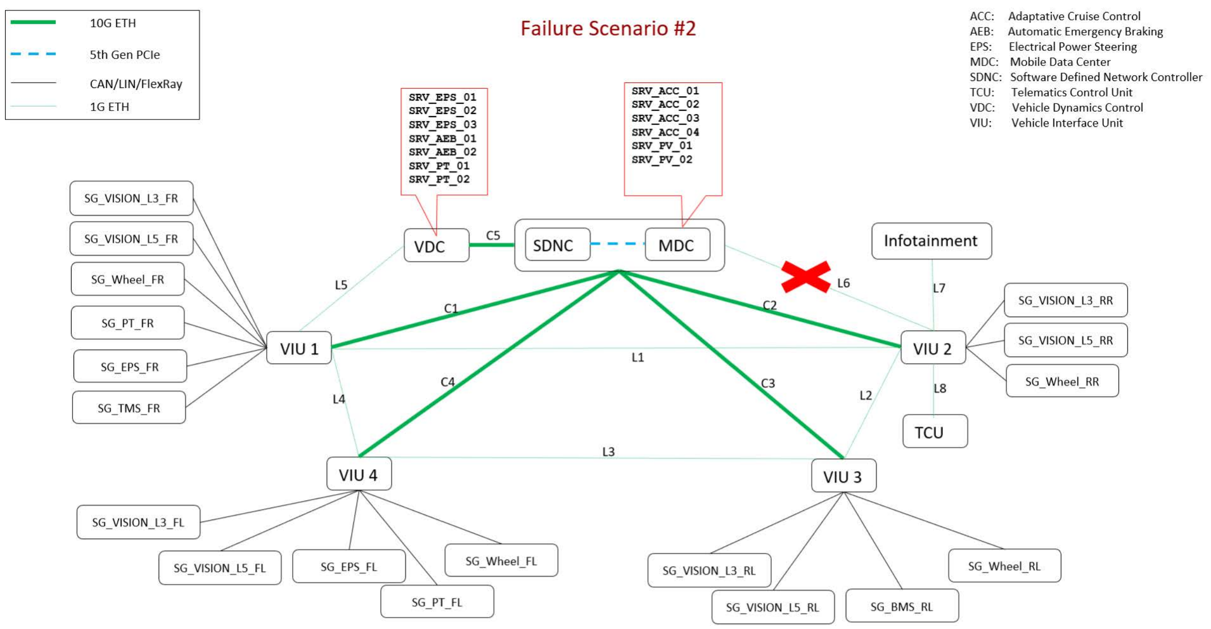

FIGURE 14. Failure scenario #2.

based on communication between network nodes and a central controller which introduces safe-mode changes [84].

Such a centralized arbitration based on the global state of the network offers the following advantages:

- It allows to achieve very low re-configuration latencies, when compared to decentralized protocols such as spanning tree protocol, where nodes must firstly agree on the state of the network;
- It decreases the amount of logic in switches which is necessary to handle reconfiguration, relatively simple switches and a single complex controller;
- It has fine-granular and safe configuration adjustments — only affected components are adjusted and the safe re-configuration order is preserved for preventing sporadic overflows during mode changes;

Also, each permanent failure (of link or node) triggers an error mode of the network where a working configuration for the network is prepared, which allows fail-operational behavior.

2) COMMUNICATION PROTOCOL BETWEEN SDNC AND eGWs/OTHER NETWORK NODES

For the fail-operational network reconfiguration mechanism to work, each end node must have a client whose task is to detect errors locally, communicate them to the SDNC and, if possible, request from the SDNC a transition to a specific

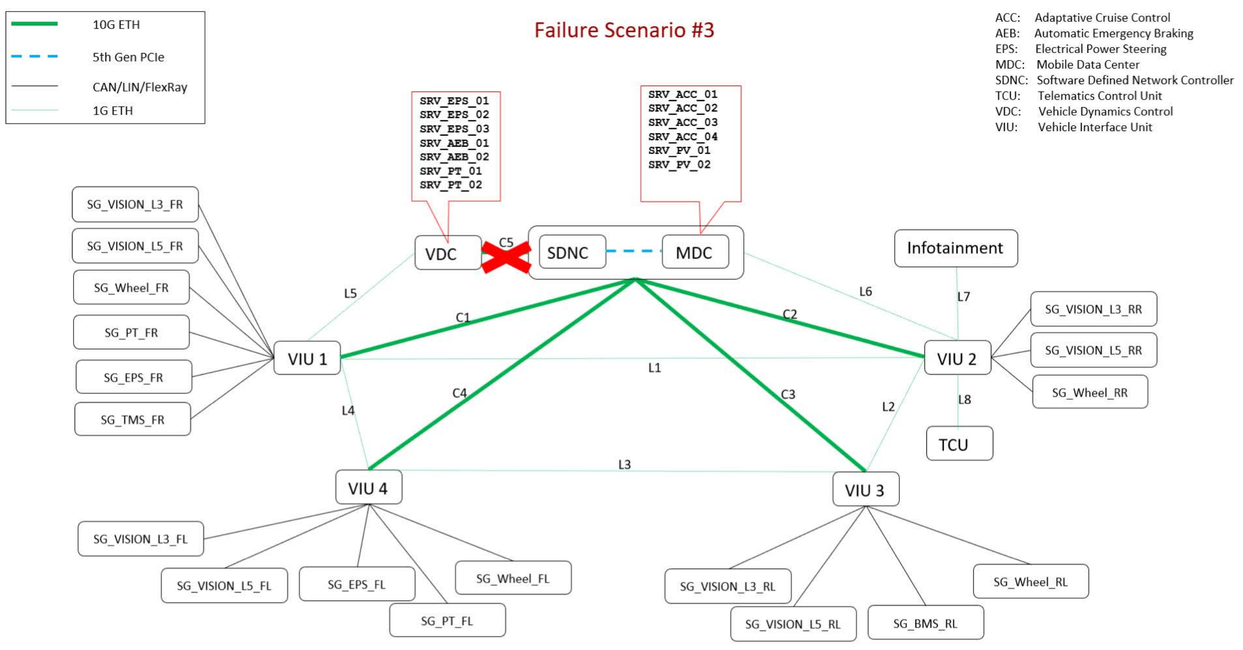

FIGURE 15. Failure scenario #3.

TABLE 12. New dataflow for SGs connected to VIU1 after failure scenario #3.

Sensor Group	Sensor Type	BW [Mbps]	Target	Path
SG_Wheel_RR	Analog / Digital IO	64	VDC	~~C2, SDNC, C5~~ L1, VIU1, L5
SG_VISION_L3_RR	Radar Lidar	128	MDC	L6
SG_VISION_L5_RR	Camera	196	MDC	L6
SG_Wheel_RL	Analog / Digital IO	64	VDC	~~C3, SDNC, C5~~ L3, VIU4, L4, VIU1. L5
SG_BMS_RL	Analog / Digital IO	64	MDC	~~C3, SDNC, C5~~ L3, VIU4, L4, VIU1. L5
SG_VISION_L3_RL	- Radar - Lidar	128	MDC	C3

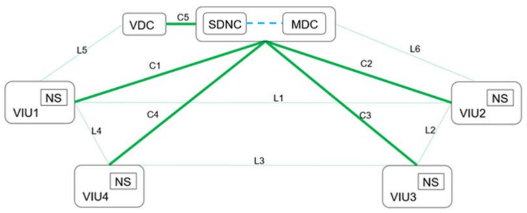

FIGURE 16. Node supervisor location within NGIVN.

state. This strategy is also required to meet *[FSR_IVN_0008]*. The enabler of this functionality is the Node Supervisor introduced in Section V. One instance of the NS is located in each VIU (and in each eGW, as a consequence, as seen in Fig. 11). The SDNC acts as the NS of both the MDC and the VDC. This is illustrated in Fig. 16.

The functions of the NS are the following:

- Error detection
- Node Configuration
- Frame routing

C. SAFETY MECHANISMS DEPLOYMENT

In order to achieve Fault-tolerance at network and VIU level, the different safety mechanisms described in Section II.B can be considered. The proposed methodology allows to select different safety mechanisms and perform the deployment to the network elements by matching the safety mechanisms to the previously defined FSRs. In Table 13 we provide a list of the possible safety mechanisms, specifying the level of application within the network (NGVIN, SDNC or VIU), the kind of functionality deployed at each level, how it is deployed (HW/SW) and what FSRs are targeted by each of them.

As seen above, the selected safety mechanism in our use case (Fast Fail-Operational NGIVN Reconfiguration) requires a combination of features deployed across the different components of the NGIVN. In essence, it is a combination of the Fast-Failover and Fast Re-Route mechanisms described in Section II.B. Additionally, node monitoring capabilities are required. The deployment of these safety mechanisms within the NGIVN and eGW are highlighted in Table 13. As seen in the table eGW provides HW support for these safety mechanisms, allowing to achieve the required performance and offloading the VIU CPU from this safety related processing.

VIII. AGILITY OF THE TOP-DOWN APPROACH FOR FUNCTIONAL SAFETY DEPLOYMENT

The novelty of our approach resides in the flexibility and elasticity provided to integrate new safety mechanisms in

TABLE 13. Safety mechanisms deployment – mapping from abstraction level to deployment strategy and corresponding FSRs.

Safety Mechanism	System Level involved	Functionality deployed	Deployment Strategy	Applicable FSRs
IEEE802.1CB	NGIVN (SDNC)	Define alternate paths for replicates	SW	*FSR_IVN_0004, FSR_IVN_0016, FSR_IVN_0017*
	eGW	Monitor / Generate / Eliminate replicates (Action stage)	HW	
Network Coding	NGIVN (SDNC)	Define alternate paths for encoded traffic	SW	*FSR_IVN_0004, FSR_IVN_0016, FSR_IVN_0017*
	eGW	Compute network coding function (Action stage)	HW	
Dual/Triple Modular Redundancy (DMR)	NGIVN (SDNC)	Du-/Tri-plicate VIUs	HW	*FSR_IVN_0017*
	eGW	Du-/Tri-plicate HW accelerators	HW	
gPTP and IEEE802.1AS	NGIVN	All NGIVN components must be gPTP capable	HW	*FSR_IVN_0004, FSR_IVN_0007*
	eGW	Perform gPTP and IEEE802.1AS HW functionalities (Action stage + loopback)	HW	
Static/Source Routing	NGIVN (SDNC)	Configure static routes	SW	*FSR_IVN_0004, FSR_IVN_0007*
	eGW	Perform in-line routing according to the established routes (Action stage)	HW	
AQM	eGW	Decide which frames to drop (Filtering / Queueing stage)	HW	*FSR_IVN_0007*
Smart Queueing	eGW	Decide where to store frames (Queueing stage)	HW	*FSR_IVN_0007*
Traffic Shapers	eGW	Perform TSN shaping algorithms (Traffic Shaping stage)	HW	*FSR_IVN_0007*
Watchdog	eGW	Count interval watchdog time (Action stage)	HW	*FSR_IVN_0008*
Built In Self Test	eGW	Identify faults across stages and notify CPU	HW/SW	*FSR_IVN_0008*
Event driven processing	eGW	Process data based on events (Match + Action stages)	HW	*FSR_IVN_0007*
Data Integrity Check	eGW	Calculate frames CRC (Normalizer stage)	HW	*FSR_IVN_0004*
Alive counter	eGW	Generate alive counter (Action stage)	HW	*FSR_IVN_0008*
Fast Failover	**NGIVN (SDNC)**	**Re-configure nodes to provide required functionality**	**SW**	***FSR_IVN_0004, FSR_IVN_0007, FSR_IVN_0011, FSR_IVN_0017, FSR_IVN_0028***
	eGW	**Host Node Supervisor and other required functionalities**	**HW/SW**	
Fast Re-Route	**NGIVN (SDNC)**	**Configure alternative paths**	**SW**	***FSR_IVN_0004, FSR_IVN_0007***
	eGW	**Detect failure and perform the alternate routing**	**HW**	
SDN node supervision	**NGIVN (SDNC)**	**Collect information of nodes status and determine if reconfiguration is required**	**SW**	***FSR_IVN_0008***
	eGW	**Monitor node status and exchange information with SDNC**	**HW**	
Load Balancing	NGIVN (SDNC)	Distribute load across VIUs	SW	*FSR_IVN_0004, FSR_IVN_0007, FSR_IVN_0028*
	eGW	Distribute load across HW resources	HW	

Highlighted lines correspond to the safety mechanisms chosen for implementation in the use case showcased in section VII

the NGIVN in a safety compliant manner. That is, we are able, with the process described in the previous sections, to specify and design a large variety of eGW safety concepts, each answering specific customer needs. Indeed, by adjusting the safety goals allocated to the NGIVN, we can either strip down or scale up the safety and availability capabilities of the eGW. This way, we can, with a minimal additional effort, design a family of gateway devices, each targeting specific customer/application needs. The safety concepts of the gateway devices belonging to the same family share the same trunk and are all derived from the safety concept of the superset device. For instance, we can specify a low-end eGW by removing all the Class 3 and Class 4 NGIVN safety goals and the FSRs and hardware/software safety mechanisms derived from them. Similarly, we can specify a mid-end eGW by removing all the Class 4 NGIVN safety goals and the FSRs and hardware/software safety mechanisms derived from them. The high-end eGW considers all the four classes of NGIVN safety goals.

This top-down approach also opens the door to safety related updates of already on the road gateway products and IVNs in general. Thus, enabling OEMs to provide additional safety related features in a vehicle already on the road, without having to go through the complete ISO 26262 cycle and without additional assessments.

Highlighted lines correspond to the safety mechanisms chosen for implementation in the use case showcased in section VII.

IX. SUMMARY AND FUTURE WORK

In this work we proposed a method to generate the safety concept and the specifications of SoCs/ECUs specialized for gateway applications. In this top-down method, we put the IVN in the center of the vehicle E/E architecture by defining and classifying the safety goals and stakeholders' requirements of a NGIVN. These safety goals and stakeholders' requirements, after following the ISO 26262 requirements for the specification and management of safety requirements [18], will eventually evolve into safety mechanisms that can be integrated on selected elements of the NGIVN as we have shown in Section VII-C. Also, by repeating this process multiple times, each time considering a different set of safety goals and stakeholder requirements, we are able to maintain a library of safety mechanisms with known characteristics and associated typical use cases. Such a library of safety mechanisms and the followed process enables to generate the safety concept of multiples elements of the NGIVN without additional efforts. This work could be complemented by considering the cybersecurity aspects of the NGIVN.

Thus, we could also reiterate this process, this time considering only cybersecurity goals and their potential interferences with the safety goals. Furthermore, we could also analyze the impact of the safety and cybersecurity goals of the NGIVN on the vehicle's power supply architecture.

ACKNOWLEDGMENT

This work is a product of collaboration between Huawei Munich Research Center and Exida Gmbh. The authors would like to thank all researchers involved in the project, especially Piotr Tański, Marek Niepiekło, Philipp Neumeier, Zhang Wei, and Qincheng Feng, for their reviews and valuable feedback.

REFERENCES

[1] *Five Trends Transforming the Automotive Industry*, McKinsey & Company, New York, NY, USA, 2018.

[2] *Race 2050—A Vision for the European Automotive Industry*, McKinsey & Company, New York, NY, USA, 2019.

[3] J. Bortolazzi, T. Hirth, and T. Raith, "Specification and design of electronic control units," in *Proc. Eur. Design Autom. Conf. EURO-DAC, EURO-VHDL Exhib.*, Sep. 1996, pp. 36–41, doi: 10.1109/EURDAC.1996.558053.

[4] J. J. Paulsen and J. M. Giachino, "Powertrain sensors and actuators: Driving toward optimized vehicle performance," *IEEE 39th Veh. Technol. Conf.*, vol. 2, Oct. 1989, pp. 574–594, doi: 10.1109/VETEC.1989.40134.

[5] P. Thoma, "Automotive electronics—A challenge for systems engineering," in *Proc. Design, Autom. Test Eur. Conf. Exhib.*, 1999, p. 4, doi: 10.1109/DATE.1999.761088.

[6] R. A. Garrini and N. J. D. Martin, "The need for a systems approach to automotive sensors-lessons hard learned," in *Proc. IEE Colloq. Automot. Sensors*, May 1992, pp. 1-1–1-5.

[7] D. Wang and S. Ganesan, "Automotive domain controller," in *Proc. Int. Conf. Comput. Inf. Technol. (ICCIT)*, Sep. 2020, pp. 1–5, doi: 10.1109/ICCIT-144147971.2020.9213824.

[8] A. Bucaioni and P. Pelliccione, "Technical architectures for automotive systems," in *Proc. IEEE Int. Conf. Softw. Archit. (ICSA)*, Mar. 2020, pp. 46–57, doi: 10.1109/ICSA47634.2020.00013.

[9] P. Pelliccione, E. Knauss, R. Heldal, S. M. Ågren, P. Mallozzi, A. Alminger, and D. Borgentun, "Automotive architecture framework: The experience of Volvo cars," *J. Syst. Archit.*, vol. 77, pp. 83–100, Jun. 2017.

[10] M. Lukasiewycz, S. Steinhorst, S. Andalam, F. Sagstetter, P. Waszecki, W. Chang, M. Kauer, P. Mundhenk, S. Shanker, S. A. Fahmy, and S. Chakraborty, "System architecture and software design for electric vehicles," in *Proc. 50th ACM/EDAC/IEEE Design Automat. Conf. (DAC)*, May 2013, pp. 1–6.

[11] J. Huang, M. Zhao, Y. Zhou, and C.-C. Xing, "In-vehicle networking: Protocols, challenges, and solutions," *IEEE Netw.*, vol. 33, no. 1, pp. 92–98, Mar. 2019, doi: 10.1109/MNET.2018.1700448.

[12] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin, "Intra-vehicle networks: A review," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 534–545, May 2015.

[13] J. Chen, H. Zhou, N. Zhang, W. Xu, Q. Yu, L. Gui, and X. Shen, "Service-oriented dynamic connection management for software-defined internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 10, pp. 2826–2837, Oct. 2017, doi: 10.1109/TITS.2017.2705978.

[14] O. Alparslan, S. Arakawa, and M. Murata, "Next generation intra-vehicle backbone network architectures," in *Proc. IEEE 22nd Int. Conf. High Perform. Switching Routing (HPSR)*, Jun. 2021, pp. 1–7.

[15] J. Walrand, M. Turner, and R. Myers, "An architecture for in-vehicle networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 7, pp. 6335–6342, May 2021.

[16] A. Ismail and W. Jung, "Research trends in automotive functional safety," in *Proc. Int. Conf. Qual., Rel., Risk, Maintenance, Saf. Eng. (QR MSE)*, Jul. 2013, pp. 1–4, doi: 10.1109/QR2MSE.2013.6625523.

[17] G. Xie, Y. Li, Y. Han, Y. Xie, G. Zeng, and R. Li, "Recent advances and future trends for automotive functional safety design methodologies," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 5629–5642, Sep. 2020, doi: 10.1109/TII.2020.2978889.

[18] *Road Vehicles—Functional Safety, Parts 1–12*, Standard ISO 26262:2018, 2nd ed., 2018.

[19] Y. Zhang, M. Yu, B. Wang, and C. Xiao, "Fault-tolerant control for electric power steering system using inverse system algorithm," in *Proc. 39th Chin. Control Conf. (CCC)*, Jul. 2020, pp. 4177–4181, doi: 10.23919/CCC50068.2020.9188371.

[20] T. Stolte, S. Ackermann, R. Graubohm, I. Jatzkowski, B. Klamann, H. Winner, and M. Maurer, "Taxonomy to unify fault tolerance regimes for automotive systems: Defining fail-operational, fail-degraded, and fail-safe," *IEEE Trans. Intell. Vehicles*, vol. 7, no. 2, pp. 251–262, Jun. 2022, doi: 10.1109/TIV.2021.3129933.

[21] T. Schmid, S. Schraufstetter, S. Wagner, and D. Hellhake, "A safety argumentation for fail-operational automotive systems in compliance with ISO 26262," in *Proc. 4th Int. Conf. Syst. Rel. Saf. (ICSRS)*, Nov. 2019, pp. 484–493, doi: 10.1109/ICSRS48664.2019.8987656.

[22] S. Ray, "Safety, security, and reliability: The automotive robustness problem and an architectural solution," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2019, pp. 1–4, doi: 10.1109/ICCE.2019.8662033.

[23] *Road Vehicles—Safety of the Intended Functionality (SOTIF)*, Standard ISO/PAS 21448:2019, 2019.

[24] L. Pintard, J.-C. Fabre, M. Leeman, K. Kanoun, and M. Roy, "From safety analyses to experimental validation of automotive embedded systems," in *Proc. IEEE 20th Pacific Rim Int. Symp. Dependable Comput.*, Nov. 2014, pp. 125–134, doi: 10.1109/PRDC.2014.23.

[25] R. Pfeffer, G. N. Basedow, N. R. Thiesen, M. Spadinger, A. Albers, and E. Sax, "Automated driving—Challenges for the automotive industry in product development with focus on process models and organizational structure," in *Proc. IEEE Int. Syst. Conf. (SysCon)*, Apr. 2019, pp. 1–6, doi: 10.1109/SYSCON.2019.8836779.

[26] *IEEE 802.1CB-2017—IEEE Standard for Local and Metropolitan Area Networks-Frame Replication and Elimination for Reliability*, Standard IEEE 802.1CB-2017, 2018.

[27] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[28] M. T. Sim and Y. Zhuang, "A dual lockstep processor system-on-a-chip for fast error recovery in safety-critical applications," in *Proc. IEEE Ind. Electron. Soc. Conf. (IECON)*, 2020, pp. 2231–2238.

[29] (2021). *Tasking. Freedom From Memory Interference*. [Online]. Available: https://resources.tasking.com/sites/default/files/2021-02/TASKING-Whitepaper-Freedom-from-Interference-Pt-1_WEB.pdf

[30] (2019). *Semiengineering. Safety Island in Safety Critical Hardware*. [Online]. Available: https://semiengineering.com/safety-islands-in-safety-critical-hardware/

[31] Y. Zhou, S. Samii, P. Eles, and Z. Peng, "ASIL-decomposition based routing and scheduling in safety-critical time-sensitive networking," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, May 2021, pp. 184–195.

[32] *IEEE Standard for Local and Metropolitan Area Networks—Timing and Synchronization for Time-Sensitive Applications—Corrigendum 1: Technical and Editorial Corrections*, Standard IEEE Std 802.1AS-2020/Cor 1-2021, Apr. 2022, pp. 1–33, doi: 10.1109/IEEESTD.2022.9765410.

[33] G. N. Kumar, K. Katsalis, and P. Papadimitriou, "Coupling source routing with time-sensitive networking," in *Proc. IFIP Netw. Conf. (Netw.)*, Jun. 2020, pp. 797–802.

[34] G. N. Kumar, K. Katsalis, P. Papadimitriou, P. Pop, and G. Carle, "Failure handling for time-sensitive networks using SDN and source routing," in *Proc. IEEE Conf. Netw. Softwarization (NetSoft)*, Jun. 2021, pp. 226–234.

[35] L. Chrost and A. Chydzinski, "On the evaluation of the active queue management mechanisms," in *Proc. Int. Conf. Evolving Internet*, Aug. 2009, pp. 113–118.

[36] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 34: Asynchronous Traffic Shaping*, Standard IEEE 802.1Qcr-2020, Nov. 2020, pp. 1–151, doi: 10.1109/IEEESTD.2020.9253013.

[37] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic*, Standard IEEE 802.1Qca-2015, Mar. 2016, pp. 1–57, doi: 10.1109/IEEESTD.2016.8613095.
[38] *IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams*, Standard IEEE 802.1Qav-2009, Jan. 2010, pp. 1–72, doi: 10.1109/IEEESTD.2010.8684664.
[39] *IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol (SRP)*, Standard IEEE 802.1Qat-2010, Sep. 2010, pp. 1–119, doi: 10.1109/IEEESTD.2010.5594972.
[40] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 26: Frame Preemption*, Standard IEEE 802.1Qbu-2016, Aug. 2016, pp. 1–52, doi: 10.1109/IEEESTD.2016.7553415.
[41] *Using the Built-in Self-Test (BIST) on the MPC5777M*, document AN5131, NXP Semiconductors, 2019. [Online]. Available: https://www.nxp.com/docs/en/application-note/AN5131.pdf
[42] A. Kostrzewa and R. Ernst, "Fast failover in Ethernet-based automotive networks," in *Proc. IEEE Int. Symp. Real-Time Distrib. Comput. (ISORC)*, May 2020, pp. 134–139.
[43] M. Chiesa, A. Kamisinski, J. Rak, G. Rétvári, and S. Schmid, "A survey of fast-recovery mechanisms in packet-switched networks," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1253–1301, 2nd Quart., 2021.
[44] J. Xie, W. Yin, and L. Wang, "Achieving flexible, low-latency and 100 Gbps line-rate load balancing over Ethernet on FPGA," in *Proc. IEEE Int. Syst., Chip Conf. (SOCC)*, Sep. 2020, pp. 201–206.
[45] K. Holzinger, F. Biersack, H. Stubbe, A. G. Marino, A. Kane, F. Fons, Z. Haigang, T. Wild, A. Herkersdorf, and G. Carle, "SmartNIC-based load management and network health monitoring for time sensitive applications," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Budapest, Hungary, Apr. 2022, pp. 1–6.
[46] *Explanation of Safety Overview*, document AP R20-11, AUTOSAR, 2020.
[47] *Overview of Functional Safety Measures in AUTOSAR*, document CP R20-11, AUTOSAR, 2020.
[48] OMG. (Mar. 2015). *Data Distribution Service (DDS) Version 1.4*. [Online]. Available: https://www.omg.org/spec/DDS/
[49] Synopsys. (2018). *What is ASIL?*. [Online]. Available: https://www.synopsys.com/automotive/what-is-asil.html
[50] Infineon. (2022). *32-Bit Tricore^{TM} AURIX^{TM}—TC4X*. [Online]. Available: https://www.infineon.com/cms/en/product/microcontroller/32-bit-tricore-microcontroller/32-bit-tricore-aurix-tc4x/
[51] NXP. (2021). *S32G Safe AND Secure Vehicle Network Processors*. [Online]. Available: https://www.nxp.com/docs/en/fact-sheet/S32G-VEHICLENW-FS.pdf
[52] *Datasheet—32-Bit Power Architecture Microcontroller for Automotive ASILD Applications*, document SPC574S64E3, Feb. 2020.
[53] (2011). *Safety Use Case Example, AUTOSAR CP Release 4.3.1*. [Online]. Available: https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_EXP_SafetyUseCase.pdf
[54] S. Shreejith and S. A. Fahmy, "Smart network interfaces for advanced automotive applications," *IEEE Micro*, vol. 38, no. 2, pp. 72–80, Apr. 2018.
[55] F. Fons and M. Fons, "FPGA-based automotive ECU design addresses AUTOSAR and ISO 26262 standards," *Xcell J.*, vol. 78, pp. 20–31, 1st Quart., 2012.
[56] S. Shreejith and S. A. Fahmy, "Security aware network controllers for next generation automotive embedded systems," in *Proc. 52nd ACM/EDAC/IEEE Design Automat. Conf. (DAC)*, Jun. 2015, pp. 1–6.
[57] F. Fons, M. Fons, P. Olivier, and A. Weimerskirch, "A modular, reconfigurable and updateable embedded cyber security hardware solution for automotive," in *Proc. Embedded World Conf.*, 2017, pp. 1–10.
[58] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for on-Road Motor Vehicles*, Standard SAE J3016:2021, 2021.
[59] *IEEE Standard for Local and Metropolitan Area Network—Bridges and Bridged Networks*, Standard IEEE 802.1Q-2018, Jul. 2018, pp. 1–1993, doi: 10.1109/IEEESTD.2018.8403927.
[60] M. Doering and J. Bierschenk, "Software-defined networking in automotive," Robert Bosch, Gerlingen, Germany, 2018.
[61] P. Fussey and G. Parisis, "Poster: An in-vehicle software defined network architecture for connected and automated vehicles," in *Proc. 2nd ACM Int. Workshop Smart, Auton., Connected Veh. Syst. Services (CarSys)*. New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 73–74.
[62] K. Halba and C. Mahmoudi, "In-vehicle software defined networking: An enabler for data interoperability," in *Proc. 2nd Int. Conf. Inf. Syst. Data Mining (ICISDM)*. New York, NY, USA: Association for Computing Machinery, Apr. 2018, pp. 93–97.
[63] M. Doering and M. Wagner, "Retrofitting SDN to classical in vehicle networks: SDN4CAN," in *Proc. 1st KuVS Fachgespräch Netw. Softwarization-From Res. Appl.* Tübingen, Germany: Eberhards Karl Univ., 2017.
[64] M. Häberle, F. Heimgaertner, H. Loehr, N. Nayak, G. Dennis, S. Schildt, and M. Menth, "An SDN architecture for automotive Ethernets," in *Proc. 2nd KuVS Fachgespräch Netw. Softwarization*. Tübingen, Germany: Eberhards Karl Univ., 2020.
[65] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements*, Standard IEEE 802.1Qcc-2018, Oct. 31, pp. 1–208, doi: 10.1109/IEEESTD.2018.8514112.
[66] T. Hackel, P. Meyer, F. Korf, and T. C. Schmidt, "Software-defined networks supporting time-sensitive in-vehicular communication," in *Proc. IEEE 89th Veh. Technol. Conf. (VTC-Spring)*, Apr./May 2019, pp. 1–5.
[67] H. Chahed and A. J. Kassler, "Software-defined time sensitive networks configuration and management," in *Proc. IEEE Netw. Function Virtualization Softw. Defined Netw. Conf. (NFV-SDN)*, Nov. 2021, pp. 124–128.
[68] K. Smida, H. Tounsi, M. Frikha, and Y.-Q. Song, "Efficient SDN controller for safety applications in SDN-based vehicular networks: Pox, Floodlight, ONOS or OpenDaylight?" in *Proc. IEEE 8th Int. Conf. Commun. Netw. (ComNet)*, Oct. 2020, pp. 1–6.
[69] A. A. Syed, S. Ayaz, T. Leinmüller, and M. Chandra, "Fault-tolerant dynamic scheduling and routing for TSN based in-vehicle networks," in *Proc. IEEE Veh. Netw. Conf. (VNC)*, Nov. 2021, pp. 72–75.
[70] A. Shukla and K.-T. Foerster, "Shortcutting fast failover routes in the data plane," in *Proc. Symp. Archit. Netw. Commun. Syst. (ANCS)*. New York, NY, USA: Association for Computing Machinery, Dec. 2021, pp. 15–22.
[71] J. Sonnenberg, "A distributed in-vehicle service architecture using dynamically created web services," in *Proc. IEEE Int. Symp. Consum. Electron. (ISCE)*, Jun. 2010, pp. 1–5.
[72] M. Wagner, D. Zöbel, and A. Meroth, "SODA: Service-oriented architecture for runtime adaptive driver assistance systems," in *Proc. IEEE 17th Int. Symp. Object, Compon., Service, Oriented Real, Time Distrib. Comput.*, Jun. 2014, pp. 150–157.
[73] *Road Vehicles—Safety and Cybersecurity for Automated Driving Systems—Design, Verification and Validation*, Standard ISO/TR 4804:2020, 2020.
[74] K. Holzinger, H. Stubbe, F. Biersack, A. G. Mariño, A. Kane, F. F. Lluis, Z. Haigang, T. Wild, A. Herkersdorf, and G. Carle, "Poster: Precise real-time monitoring of time-critical flows," in *Proc. 17th Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, Dec. 2021, pp. 489–490.
[75] N. Luangsomboon and J. Liebeherr, "A round-robin packet scheduler for hierarchical max-min fairness," 2021, *arXiv:2108.09864*.
[76] B. Briscoe and A. S. Ahmed, "TCP Prague fall-back on detection of a classic ECN AQM," 2019, *arXiv:1911.00710*.
[77] B. Mathieu and S. Tuffin, "Evaluating the L4S architecture in cellular networks with a programmable switch," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Sep. 2021, pp. 1–6, doi: 10.1109/ISCC53001.2021.9631539.
[78] A. G. Mariño, F. Fons, L. Ming, and J. Arostegui, "PDU normalizer engine for heterogeneous in-vehicle networks in automotive gateways," in *Proc. Int. Symp. Appl. Reconfigurable Comput. (ARC)*, Jun. 2021, pp. 140–155, doi: 10.1007/978-3-030-79025-7_10.
[79] A. G. Mariño, A. A. Kane, F. Fons, and J. M. M. Arostegui, "Enhancements for hardware-based IEEE802.1CB embedded in automotive gateway system-on-chip," in *Proc. Symp. Archit. Netw. Commun. Syst.*, Dec. 2021, pp. 31–37, doi: 10.1145/3493425.3502754.
[80] A. G. Mariño, F. Fons, Z. Haigang, and J. M. M. Arostegui, "Loopback strategy for in-vehicle network processing in automotive gateway network on chip," in *Proc. 14th Int. Workshop Netw. Chip Archit.*, Oct. 2021, pp. 22–28, doi: 10.1145/3477231.3490429.
[81] A. G. Marino, F. Fons, Z. Haigang, and J. M. M. Arostegui, "Traffic shaping engine for time sensitive networking integration within in-vehicle networks," in *Proc. IEEE Veh. Netw. Conf. (VNC)*, Nov. 2021, pp. 182–189, doi: 10.1109/VNC52810.2021.9644668.

[82] A. G. Marino, F. Fons, Z. Haigang, and J. M. M. Arostegui, "Loopback strategy for TSN-compliant traffic queueing and shaping in automotive gateways," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2021, pp. 47–53, doi: 10.1109/NFV-SDN53031.2021.9665092.

[83] A. G. Marino, F. Fons, A. Gharba, L. Ming, and J. M. M. Arostegui, "Elastic queueing engine for time sensitive networking," in *Proc. IEEE 93rd Veh. Technol. Conf. (VTC-Spring)*, Apr. 2021, pp. 1–7, doi: 10.1109/VTC2021-Spring51267.2021.9448758.

[84] A. Kostrzewa and R. Ernst, "Fast failover in Ethernet-based automotive networks," in *Proc. IEEE 23rd Int. Symp. Real-Time Distrib. Comput. (ISORC)*, May 2020, pp. 134–139, doi: 10.1109/ISORC49007.2020.00027.

ABDOUL AZIZ KANE received the master's degree in electrical engineering with a specialization/focus on embedded electronics from ESCPE Lyon, France, in 2013.

He currently works with the Huawei Technology's Munich Research Center as a Principal Functional Safety Researcher for level 5 compatible in-car communication networks. Previously, he worked at the Infineon Technologies as an Application and Concept Engineer for automotive microcontrollers and specialized on the application of the automotive functional safety standards and the safe integration of the microcontroller into a system. His research interest includes the conceptualization and design of safe in-car communication networks for automated driving.

ANGELA GONZALEZ MARIÑO received the bachelor's degree in telecommunications engineering from the Universidade de Vigo (UVIGO), Vigo, Spain, in 2015, and the master's degree in electronics engineering systems from the Universidad Politecnica de Madrid (UPM), Madrid, Spain, in 2016. She is currently pursuing the Ph.D. degree with the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain.

She worked at HP Inc., Barcelona, as a Research and Development Electronics Engineer, from 2016 to 2020, designing electronics for large format printers and supporting the full product lifecycle development. At present, she is with the Applied Network Technology Laboratory, Munich Research Center, Huawei Technologies, Munich, Germany, focusing on HW accelerators design for automotive networking solutions. Her current research interests include HW design for automotive in-vehicle networks and system on chip design.

FRANCESC FONS (Senior Member, IEEE) received the bachelor's degree in electrical engineering, the master's degree in automatic control and industrial electronics engineering, and the Ph.D. degree in electronics technology from Universitat Rovira i Virgili (URV), Tarragona, Spain, in 1995, 2001, and 2012, respectively.

He is currently with Huawei Technologies, where he has the role of a Chief Automotive In-Vehicle Network Researcher with the Applied Network Technology Laboratory, Huawei Munich Research Center. He has focused his professional career on the automotive electronics industry, working on research and development in the areas of embedded software, systems, hardware, and networks. Along his career, he has been with different automotive Tier 1 and Tier 2 suppliers from the USA, Germany, and China, and has participated in the successful launch of many commercial products for OEMs at Europe and Asia.

SANDRO NUEESCH was born in Bern, Switzerland, in July 1987. He received the B.Sc. and M.Sc. degrees in mechanical engineering from ETH Zürich, Switzerland, in 2010 and 2012, respectively, and the Ph.D. degree in mechanical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2015.

He is currently working with the Munich Research Center, Huawei Technology, as the Principal Engineer on automotive safety architecture. Previously, he has worked at Ford Motor Company as a Functional Safety and System Engineer. His current research interests include automotive safety in general and safety of automated driving specifically as well as model-based systems and safety engineering.

PIOTR SERWA graduated in industrial engineering from INSA Lyon and computer science from the Technical University of Karlsruhe and the master's degree from the University of Karlsruhe.

He has over 20 years of experience and extensive knowledge in the safety and reliability field. He has an extensive experience in system and software safety concepts, specification, design and implementation, and its verification. He has also an extensive safety-related experience of AUTOSAR and POSIX OSes. He is also experienced in safe machine learning. After graduating from INSA Lyon as an Industrial Engineer, he began working at the Institute of Telematics, University of Karlsruhe, and then the Institute of Mathematics as a Researcher, while completing another master's degree at the University of Karlsruhe as a Computer Scientist. In 2005, he joined Exida worked as a Safety Expert, where he participated in several projects in the domain of automotive, automation, process industry, and nuclear. Since 2005, he has also been an Active Member in AUTOSAR consortium on behalf of BMW. Since 2015, he has been involved in projects related to qualification of complex software, autonomous driving, and machine learning.

MICHAEL SCHOETZ received the degree from the University of Applied Sciences, Regensburg.

He has more than 22 years of experience and extensive knowledge in the safety and reliability field. After graduating from University of Applied Sciences, he began working for Siemens Semiconductors, where he was responsible for definition and testing of safety diagnostics for CMOS-based integrated airbag sensors. In 2000, he joined Siemens Automotive (SiemensVDO), where he became responsible defining applications for Multimedia Head Units. In 2003, he has been nominated as the Department Responsible regarding Functional Safety within Siemens VDO Body and Chassis. In 2008, he joined exida.com and he was named as an Exida Partner, in 2011. Within exida, his primary focus is on the automotive sector regarding applications drivetrain, chassis, and vehicle dynamics. Main fields are concept and system development, including the related integration and validation activities. Besides this, he is a member of the German Mirror Committee regarding ISO 26262, called NA 052-00-32-08.