Received 8 June 2022, accepted 21 June 2022, date of publication 27 June 2022, date of current version 5 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3186708

TOPICAL REVIEW

The Future Roadmap of In-Vehicle Network Processing: A HW-Centric (R-)evolution

ANGELA GONZALEZ MARIÑO[1], FRANCESC FONS[1], (Senior Member, IEEE), AND JUAN MANUEL MORENO AROSTEGUI[2]
[1]Huawei Technologies Düsseldorf GmbH, 80992 Munich, Germany
[2]Department of Electronic Engineering, Technical University of Catalonia (UPC), 08034 Barcelona, Spain

Corresponding author: Angela Gonzalez Mariño (angela.gonzalez.marino@huawei.com)

This work was supported by Huawei Technologies Dusseldorf GmbH.

ABSTRACT The automotive industry is undergoing a deep revolution. With the race towards autonomous driving, the amount of technologies, sensors and actuators that need to be integrated in the vehicle increases exponentially. This imposes new great challenges in the vehicle electric/electronic (E/E) architecture and, especially, in the In-Vehicle Network (IVN). In this work, we analyze the evolution of IVNs, and focus on the main network processing platform integrated in them: the Gateway (GW). We derive the requirements of Network Processing Platforms that need to be fulfilled by future GW controllers focusing on two perspectives: functional requirements and structural requirements. Functional requirements refer to the functionalities that need to be delivered by these network processing platforms. Structural requirements refer to design aspects which ensure the feasibility, usability and future evolution of the design. By focusing on the Network Processing architecture, we review the available options in the state of the art, both in industry and academia. We evaluate the strengths and weaknesses of each architecture in terms of the coverage provided for the functional and structural requirements. In our analysis, we detect a gap in this area: there is currently no architecture fulfilling all the requirements of future automotive GW controllers. In light of the available network processing architectures and the current technology landscape, we identify Hardware (HW) accelerators and custom processor design as a key differentiation factor which boosts the devices performance. From our perspective, this points to a need - and a research opportunity - to explore network processing architectures with a strong HW focus, unleashing the potential of next-generation network processors and supporting the demanding requirements of future autonomous and connected vehicles.

INDEX TERMS In-vehicle networks, gateway, network processing, HW accelerators, software defined networking.

I. INTRODUCTION

According to exhaustive market and future trends analysis done by consultancy experts [1], [2] there are at least 4 main mega-trends that shape the future of the Automotive Industry: Autonomy, Connectivity, Electrification and Sharing (ACES). These mega-trends are driven by social changes and new technologies that enable the paradigm change. They can be classified under two categories: enablers and motivators. On one hand, Autonomous Driving (AD) and shared mobility are classified as motivators because they represent the final vision of how mobility will be in the future. On the other hand, Connectivity and Electrification are classified as enablers because they represent the technology change required to enable AD and Shared mobility. In the field of AD there have been huge advances in the last years, and more new and exciting advances are yet to come. Driving automation levels are regulated by the Society of Automotive Engineers (SAE) in SAE J3016 [3] and go from Level 0 (driver support features limited to providing warnings or momentary assistance), to Level 5 (automated driving features that can drive the vehicle under all conditions). Industry leading

The associate editor coordinating the review of this manuscript and approving it for publication was Barbara Masini.

Original Equipment Manufacturers (OEMs) are working towards the highest levels of autonomy which will enable huge social changes when the highest level is achieved: the possibility to get a ride in a car without driver (and without driving license) will exist. This paradigm shift brings new requirements to the automotive industry. Today, technologies that were once only seen in data centers are being fully integrated into vehicles.

Accordingly, the needs for processing power are increasing incredibly quickly inside the vehicles. It is not only raw application processing that is becoming more complex, but also the exchange of information between the different processing units, sensors and actuators in the vehicle. For instance, new functionalities such as Autonomous Driving require real time reaction to events, which means that data needs to be collected at the sensors, transmitted to the processing unit, processed and sent to the corresponding actuator within a bounded (and usually short) period of time. This means that the transmission of data through the network needs to be reliable and deterministic which derives in new stringent requirements in terms of network processing capabilities. This is why the electronics architecture and the In-Vehicle Network (IVN) are now in the center of attention [4]. In this work we focus on the main component of the IVN: the gateway (GW). By definition, a gateway is a device that connects heterogeneous networks by translating from one communication protocol to another. Traditionally, it was separated from the router, which is in charge of delivering data within one single network, with one network technology, typically Ethernet. However, the trend in the recent years has been to combine both gatewaying and routing functionalities within one device. Hereafter, when we talk about a gateway device, we refer to a device capable of both gatewaying and routing features. As the main component of the In-Vehicle Network, the gateway is in charge of network processing functionalities and is therefore affected by all the new requirements concerning the IVN. Furthermore, the gateway can also be extended with application processing functionalities, increasing its own complexity and gaining even more importance within the vehicle electronics architecture.

The semiconductor industry is also seeing significant changes on its well-established design model. While at manufacturing level the race towards smaller technologies which lead to higher integration, lower production costs and higher performance continues to be the norm, the revolution is happening in the design of the Integrated Circuits (ICs) itself. Traditionally, big IC designers like Intel or Nvidia would design the best general purpose compute platforms in the market and companies in different industries would make their designs based on these platforms. Competition and differentiation between companies within a sector like the automotive would be based on the design of functionalities and capabilities reached with one platform, that could be the same one used in a competitor's product. However, now there is an ongoing trend to change this model of operation: big technology companies are starting to develop their own ICs customized to their specific needs in order to generate a competitive advantage. Tesla [5], Google [6], Amazon [7], Facebook [8] and Apple [9] have already made this decision and are ready to launch their products with their own customized ICs. There is of course a great entry barrier in this approach, but all the big companies that can afford it are shifting towards this paradigm because of the customization opportunity that can increase the performance of their products significantly. This way, they can also prevent competitors from using the same technology that they use, while bringing differentiation as a competitive advantage. This decision not only provides freedom for developing the hardware (HW) accelerators and co-processors that each of them needs, but also eliminates the dependence on IC designers both in economical aspects and capabilities, and adds the (HW) chipset to the list of core technologies for the company.

Observing the changes both in the automotive and the semiconductor industry, we identify a trend where the design of gateway controllers for in-vehicle network processing becomes an essential part of the vehicle as a product. In this work, we explore this trend and bring the following contributions (C):

- **C1. Requirements analysis of future In-Vehicle Networks**. We define the requirements for network processing platforms derived from the integration of new technologies and functionalities within IVNs. We classify these requirements under two categories: 1) Functional Requirements (FR): those covering the features that need to be supported by network processing platforms from a functional perspective, 2) Structural Requirements (SR): those related to the architectural design that ensure the feasibility, usability, scalability and future maintenance/evolution of the architecture.
- **C2. Study of currently available network processing platforms**. We analyze the available platforms from the architecture point of view, identifying their strengths and, more importantly, the existing bottlenecks related to the processing of the identified requirements (functional and structural).
- **C3. Identification of existing gaps and future trends**. We compare all the analyzed architectures and verify to what extent they are able to cover the requirements derived in C1. With this, we are able to identify current gaps and relate them to the current trends observed both in automotive and semiconductor industry, providing insights and guidelines for future research opportunities.

The remainder of this paper is structured as follows. In chapter II we provide relevant background information related to the automotive electronics industry, with special attention to IVN architecture evolution and challenges concerning network processing units. In Section III we derive the requirements of network processing platforms in order to meet the demands of future vehicles. We consider both functional and structural requirements of these network processing platforms. In section IV we analyse currently available network processing architectures, highlighting their advantages and disadvantages. We also analyse to which extent each architecture fulfills or not the aforementioned requirements. Then, we conclude the work in Section V discussing the gap identified in the state of the art and providing guidelines for future research directions. Finally, in Section VI we provide a classification of the most relevant research works discussed throughout this work.

II. AUTOMOTIVE ELECTRONICS INDUSTRY

The integration of technologies required in order to follow the automotive mega-trends is intrinsically linked to the development of new electric/ electronic (E/E) in-vehicle architectures composed of several interconnected computing platforms.

The traditional automotive computing platform, which is the Electronic Control Unit (ECU), was born in order to provide simple control of one specific sensor/actuator. On a first approach, for each electronic function in the car one dedicated ECU with a very specific control task would be incorporated. When new sensors or actuators were required, new ECUs would also be incorporated. With the exponential increase of electronic functions in the car this approach did not make sense any more because the amount of ECUs and their interconnections became impossible to handle.

Therefore, on a second iteration, ECUs started to group sensors based on functionality or domains. This way, there would be an ECU in charge for lights, another one for infotainment, another one for motion control, etc. This is a logical distribution which permits to handle the different sensors and actuators in groups, attending to the specific requirements of each group. For instance, in terms of safety and security, different requirements apply to the infotainment domain or to the motion control domain. This architecture is the one used in today's vehicles, and is exposed in Fig. 1-a.

However, for most of the functionalities available in a vehicle, there are sensors and actuators in different physical places in the car, e.g. there are lights in the front and in the back, the same for cameras, proximity sensors, etc. Due to this, several different cables need to be routed across the whole vehicle to interconnect all the sensors and actuators of each domain, as seen in Fig. 1-a. This dissociation between the logical/functional and physical distribution of sensors within the IVN, is the main disadvantage of the domain architecture, which becomes a relevant issue when moving towards autonomous driving. This is because, once again, the amount of sensors and actuators that need to be integrated in the IVN is increasing, making the domain-based approach not scalable from architecture and topology point of view. Furthermore, the cabling cost associated becomes too high.

Now, on a third iteration, a new generation of ECUs, and IVNs is needed in order to provide the required functionality. This need has been already identified in the state of the art, and there are several works about it. In [10] an overview of current ECUs and IVN solutions is exposed together with some insights into possible architectures for future IVNs. In [11], authors review the different network protocols used within vehicular networks and analyze the benefits of Ethernet as a backbone interconnecting a central processing unit with several distributed network processing nodes. In [12] authors discuss the evolution of IVNs and focus on analysing alternatives for the backbone of the network. In [13] authors focus also on Ethernet as a backbone of the IVN and propose an architecture and specific configuration for a particular use case, which, similarly to the proposal in [10] and [12], is also composed of a central node and several distributed nodes interconnected between them. In [14] authors give an overview on how all the new functionalities can be mapped to the IVN central processing unit, while providing the required level of reliability and performance. They also identify the current bottlenecks at software level in terms of handling the required complexity.

As seen above, according to the works available in the literature, one of the proposed architectures which seems to fulfill the requirements of future vehicles encompasses both centralized and distributed features. The reasoning for such an architecture is that some centralized control is needed, but fully centralized control is neither feasible nor scalable. Therefore, defining zones within the architecture which have their own ECU or zonal gateway controller, and are interconnected between them, provides the required flexibility, performance and scalability for future vehicle architectures. In this architecture, shown in Fig. 1-b, sensors and actuators are connected to an ECU that is physically placed in the same zone, reducing dramatically the cabling cost. ECUs distribution changes from functionality or domain to physical zone inside the vehicle, i.e. the distribution of computing platforms is no longer logical, but physical, solving the scalability problem identified in the domain architecture. This new proposal is called "Zonal Architecture". This change in the IVN from domain to zonal architecture is shown in Fig. 1.

The zonal architecture helps in solving the limitations of the domain-based architecture but, it brings new challenges to overcome. Before, each ECU was only in charge of sensors and actuators of one domain, being able to handle its requirements without issues. Now, each ECU or zonal controller needs to handle sensors and actuators of different domains, which are placed in the same physical area. This involves mixing traffic of different functionalities and with different requirements.

With this paradigm shift, the computation platforms become much more complex than a traditional ECU or domain controller in terms of mixing heterogeneous and cross-domain functionalities, distributed and synchronized across several vehicle zones. Therefore, the design of these computing platforms becomes of utmost importance in order to incorporate the desired functionalities and technologies. One of the challenges of future automotive network processing platforms is to find the right balance between the different requirements imposed by this heterogeneous set of functionalities and technologies.

The resulting architecture can be leveraged from low-end vehicles up to high-end vehicles by changing the performance

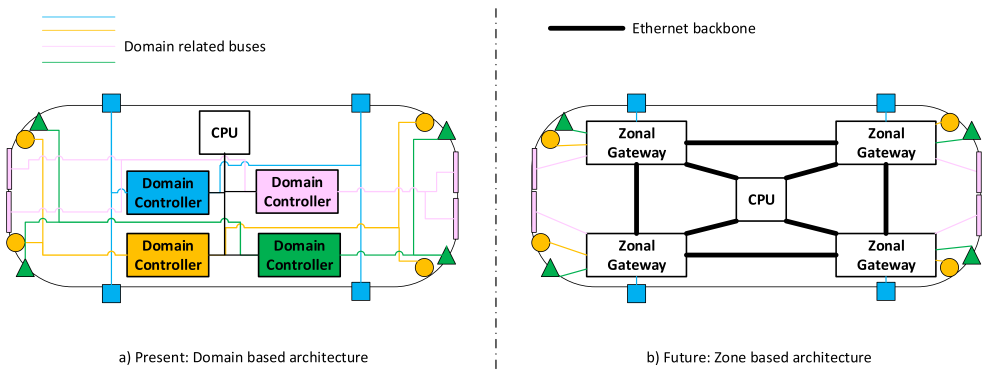


FIGURE 1. **Evolution of intra-vehicular networks.**

of the included computation platforms. The integration of High Performance Computers (HPCs) can be left for the high-end vehicles where all the functionalities and best quality features are deployed, while in the low-end vehicles, ECUs with dedicated HW accelerators can have enough computation power to support the required functionalities with the adequate level of performance. This solution provides important cost savings by reusing the same architecture and components for both low-end and high-end vehicles, since both benefit from high production volume of the shared components. At the same time, the most expensive component (HPC) could be required only in the high-end scenario where although volumes are lower, margins are higher.

Observing trends and evolution in other computing platforms like Micro-Controller Units (MCUs) based in the Advanced Reduced instruction set Machines (ARM) architecture, which started as low end MCUs and are now taking the data centres world [15], it does not seem unreasonable to think or foresee that HPCs will be completely removed from the vehicle architecture by including new custom HW accelerators in ECUs when these reach the proper performance and cost.

III. NETWORK PROCESSING PLATFORMS REQUIREMENTS

As seen in the previous section, going towards future zonal architectures, the Gateway (GW) controller becomes the central piece of the In-Vehicle Network in terms of network processing. Therefore, we focus this work on the study of Gateway platforms from different perspectives.

In this section, we identify the requirements that need to be fulfilled by future automotive Gateway devices. These requirements are classified into functional requirements and structural requirements.

- Functional requirements relate to a functionality that must be provided by the GW platform as part of the network infrastructure in a vehicle.

- Structural requirements relate to design practices and the viability of the architecture design towards a real Network/System on Chip (NoC/SoC).

1) FUNCTIONAL REQUIREMENTS

- **FR0. Generic Network Processing Management**
  As the main network processing device within the IVN, the gateway must be able to perform generic network processing management, common to other network infrastructures. This includes the routing and forwarding of frames at layer 2 or higher, packet processing or frames priority management. Furthermore, given the increase in the computation capabilities of IVN Gateways, it is also expected to perform some application layer tasks such as frame manipulation and transformation, or frames generation, shifting functionality from the HPCs to the zonal GW controllers. This enables the autonomy of the zonal gateway as a computing entity that takes full control of the functions in a delimited region (zone) of the vehicle, and enables the distributed zonal architecture.
- **FR1. Heterogeneous Networks Management**
  Linked with the aforementioned Connectivity mega-trend, IVNs are evolving according to the new requirements and technologies integrated into vehicles. Nowadays, the heterogeneity of IVNs and how to manage them is one of the biggest challenges to overcome. Traditionally, the IVN was composed of different buses: Controller Area Network (CAN) up to 1 Mbps [16], Local Interconnect Network (LIN) up to 19.2 kbps [17], FlexRay (10 Mbps) [18], Media Oriented System Transport (MOST) up to 150 Mbps [19], and more recently also CAN with Flexible Datarate (CAN-FD) and CAN-XL up to 10 Mbps [20], [21]. Although Ethernet is becoming more and more used for IVNs, (since few years ago and 100Base-T1S and 1000Base-T1S are already adopted), this heterogeneity will continue to exist in the following years due to two main factors [22]:

  – Backwards compatibility is a requirement in the automotive industry. Sensors and actuators that are in the market today use legacy buses such as CAN, LIN, FlexRay and MOST as an interface and will continue to do so in the near future. Introduction of new technologies will be gradual so coexistence of legacy and new technologies is mandatory.

– There is currently no alternative that can cover all the use cases of IVNs. Requirements for a subsystem that is using LIN protocol differ enormously from requirements for a subsystem that is using Ethernet. Nowadays, the universal bus that covers high-speed to low-speed, with wide bandwidth (BW) and fault detection mechanisms at reasonable cost, does not exist.

Although the new Ethernet-based technology 10Base-T1S aims at replacing other technologies like CAN-XL and FlexRay in the range of up to 10Mbps, this is a trend still in development phase today. The reasoning behind this trend is to harmonize the network as an Ethernet-centric solution and reduce thus complexity in gateways in terms of handling heterogeneous network protocols. Anyhow, whatever technology transition in automotive occurs gradually through integration steps that take considerable time due to the long standardization, validation and adoption process.

Therefore, in future automotive platforms there is a need to efficiently handle this heterogeneity and, as in existing architectures, the component that enables the translation between different protocols, is the Gateway. The current solution used to handle the translation from the different automotive protocols to Ethernet is the IEEE1722 encapsulation [23]. This standard provides a common frame format to embed frames of different protocols into Ethernet frames.

Apart from the need for a strategy to translate between network protocols, there is also the challenge of efficiently handling the required QoS for all of them. As a reference, Table 1, gives an overview of the requirements in terms of BW and latency of different applications classified in domains together with the technologies involved in each of them. In this regard, having a backbone that operates at a higher rate than the other protocols in the network allows to allocate the requirements of each protocol and ensure that the worst case latency is guaranteed across the network for all of them, as well as ensuring the throughput of each gateway withstands the aggregated BW of its ports. However, the QoS of this backbone must take into account the needs of all the sensors and actuators present in the network in order to satisfy these requirements. The topics of QoS and determinism for Ethernet networks are discussed later in this section, in FR4.

There are also several works in the literature trying to provide solutions to handle different network protocols. In [24], authors present a software (SW) based solution for the management of this heterogeneity of networking technologies within automotive gateways. In [25], authors deal with the diversity of communication protocols in a different environment (generic embedded applications) and propose a virtualization model for the management of this heterogeneity.

TABLE 1. **BW and latency requirements of different domain applications.**

Domain	Bandwidth	E2E Latency	Network Technology
Powertrain	<10 Mbps	<1 ms	CAN / LIN / FlexRay / 10Base-T1S
Body	<100 Mbps	<100 ms	CAN / LIN / 10Base-T1S / 100Base-T1
Infotainment	>1 Gbps	<10 ms	1000Base-T1
ADAS	>1 Gbps	<1 ms	1000Base-T1

- **FR2. Safety and Security**

Safety has always been an intrinsic requirement within the automotive industry. Due to the fact that people's safety is at stake, the system must be absolutely safe. With the vehicle electrification, strict measures need to be taken with regards to safety. Electrical and electronics systems must be intensely tested in order to cover all possible cases and detect potential issues, and also life tests must be performed in order to guarantee the absence of failures. With the increased complexity of systems, failure modes may also increase and even be almost impossible to cover. It is under these conditions when safety becomes even more important and the safety concept turns a key aspect of the initial design.

Making systems which are safe by design turns out to be the best option in order to comply with the required safety standards that apply when going to higher levels of autonomy. Another important aspect is the logging feature, that allows to have visibility of what is going on in the network and to perform the correct diagnosis [26].

Next, we describe the available standards regarding safety related requirements in automotive:

- ISO 26262 [27]: Road vehicles — Functional safety. This is the first specific standard for Functional safety with regards to vehicles, published by the International Organization for Standardization (ISO), and is based on IEC 61508 from International Electrotechnical Commission (IEC) [28]. The standard describes different levels according to safety classification, called ASIL (Automotive Safety Integrity Level), going from ASIL-A to ASIL-D, being ASIL-D the highest level of safety described in this standard.

- ISO 21448 [29]: Going towards higher levels of autonomy, ISO 26262 becomes insufficient to cover new use cases like Software Over The Air updates (SOTA), autonomous driving, unintended behavior, etc. Therefore, new standards trying to cover the gap are being currently developed. SOTIF standard (Safety Of The Intended Functionality) specifies how to consider known/unknown use cases and how to identify limitations of countermeasures. It introduces verification and validation concepts.

- UL4600 [30]: Standard for Safety for the Evaluation of Autonomous Products. It covers the safety requirements

for fully Autonomous Vehicle (AV) operation, where there is no human driver/supervisor.
In line with the definition of these standards and the increasing complexity introduced by AD, some of the world's major mobility stakeholders are working together and have created "The Autonomous" association [31], in order to shape the future of safe autonomous vehicles. Within this working group, an overview of the autonomous vehicle governance ecosystem with guidelines for decision makers has been published [32]. According to this, the industry is putting great effort in advancing towards functional safety systems.
There are already some companies which are providing a functional safety assessment of their products targeting the autonomous driving industry, like TTTEch Auto, with their MotionWise platform [33], NXP with their Safe Assure program for the S32G processor [34] or Renesas with their Functional Safety support for automotive [35].
Regarding security in IVNs, the complete architecture is secured via 4 layers of security deployment: secure (external) interfaces, secure GW (internal communication), secure network and secure processing [36]. The focus of this work is on GW architectures, therefore the security layer that is further analysed as a requirement is layer 2: Secure Gateway. The basis of securing GWs relies on network domain division and firewall rules that are applied to determine which communications between domains are allowed. The deployment of this layer of security is nowadays done via SW solutions following the Evita Project [37] pillars. Data encryption/decryption like Media Access Control Security (MACSec) [38] is supported by the Hardware Security Module (HSM). Other security functions demanded in automotive gateways are firewall functionalities such as IPTABLES [39] and Network Intrusion Detection Systems (NIDS) like SNORT [40] or SURICATA [41]. All of these solutions focus on providing rules to determine how the system should react when a specific kind of traffic is present in the network. The resolution can be to allow or not allow the traffic, but also to log some frames or to generate system alerts. They do not only set rules considering frame's header, but also by inspecting the data contained in the payload. The flexibility required in defining these rules is very high, and there is a need of having run time update capabilities so that the system can adapt the rules according to previously seen traffic and learn from it, e.g. in the case of NIDS upgrades against new cyber-security attacks.
Nowadays, the implementation of these secure approach is completely SW-based, and represents a significant overload in Gateway Central Processing Units (CPUs). There are some works on the literature about the challenges of integrating safety and security in IVNs [42].

TABLE 2. Expected future sensors needed for ADAS features.

Sensor	Quantity	Bandwidth
Lidar	1-10	<100 Mbps
Radar	4-8	<100 Mbps
Camera	2-16	1-5 Gbps
Ultrasonic	8-16	<10 Mbps
Motion sensors, inertia measurement unit (IMU)	1	<10 Mbps

- **FR3. Throughput and Performance**
With the new technologies integrated within IVNs, especially the sensors required for autonomous driving (cameras, radar, etc.), the amount of data that needs to be exchanged in the network increases exponentially. Actually, it is not only the amount of data but also the speed at which it needs to be delivered across the network that increases dramatically. Today, most flows within a vehicle require less than 1 Gbit/s throughput. However, in order to enable future use cases, this is expected to increase to the range of some Gbit/s per flow, as analysed by authors in [11]. Table 2 shows an estimation of sensors that will be integrated into vehicles in order to enable autonomous driving features, the amount of sensors of each type and the BW consumed by each of them according to [43], [44].
When several Gbit/s flows need to be handled at the same point, the network processing devices must be able to process at much higher data rates, in order to avoid becoming a bottleneck. Therefore, it is a requirement for network processing devices to provide multi Gbit/s throughput. At the same time, lower data rate flows will continue to exist in the network [45], as also seen in Tables 1 and 2. This means that the network processing device must be able to handle a variety of throughput rates and smartly provide the Quality of Service (QoS) required by each of them.
Once again we see how IVN require a combination of technologies with different purposes. Thus, the challenge is on how to combine all of them at the right level of performance, rather than at providing one single item with maximum quality.
- **FR4. Quality of Service: Determinism**
Future IVN gateways must provide deterministic and reliable transmission of packets with the right QoS. We consider the future IVN architecture depicted in Fig. 1-b with Ethernet as a backbone of the network. In such network, the capability to provide reliable performance for distributed applications depends on the determinism of the communication through this backbone. For instance, an application that collects information from sensors in the four zones and generates an action that also needs to be deployed in the four zones requires guarantees in terms of end to end transport latency, e.g. synchronization of lights.
In terms of determinism, the IEEE group on Time Sensitive Networking (TSN) has been developing a set of

standards that set the ground for deterministic and reliable Ethernet networks. Given the fact that TSN covers a broad range of applications, different profiles are being defined for the different use cases.
In the specific case of automotive, the P802.1DG [45] profile is currently under development [46]. As stated in the P802.1DG web page, "This standard specifies profiles for secure, highly reliable, deterministic latency, automotive in-vehicle bridged IEEE 802.3 Ethernet networks based on IEEE 802.1 Time-Sensitive Networking (TSN) standards and IEEE 802.1 Security standards". As of today, P802.1DG considers that the required TSN standards to be applicable in automotive IVNs, are the ones listed below.
- 802.1AS [47] (base profile, required): The time synchronization standard coming from Audio Video Bridging (AVB) standards, provides time synchronization across the network. It relies on the generalized Precision Time Protocol (gPTP) to exchange synchronization messages. This is the basis of operation for other TSN standards that provide enhancements of traffic management in a synchronized network.
- 802.1Qci [48] (base profile, required): Per Stream Filtering and Policing standard describes the mechanisms to define traffic flows and which characteristics shall be used in order to filter and classify them. Within IVNs, this is of great importance given the fact that it is necessary to define critical flows like brake system or steering column and guarantee there are no interference of traffic going back and forth to these subsystems as the consequences can be highly critical.
-802.1Qav [49] (base profile, required): Forwarding and Queuing Enhancements for time-sensitive streams standard describes the Credit Based Shaper (CBS) algorithm used to limit the BW consumption of certain TSN flows.
- 802.1CB [50] (extended profile, required): Frame Replication and Elimination for Reliability standard describes the strategy to comply functional safety requirements based on physical redundancy already available in the network. Basically, critical defined flows are replicated via alternate paths in order to guarantee the arrival of information even if one path is suddenly interrupted.
The required functions to generate frame replicates, and also to detect them on reception side are defined. Diagnosability obtained out of the replicates detected can be very powerful not only for status of physical links but also for network congestion.
- 802.1Qbv [51] (extended profile, optional): Enhancements for scheduled traffic standard describes the system architecture and method for the Time Aware Shaper (TAS) of TSN flows.
- 802.1Qbu [52] (extended profile, optional): Frame preemption standard describes the system mechanisms required in order to provide frame preemption capabilities. This mechanism distinguishes between express and preemptable frames, where express frames are allowed to interrupt the transmission of preemptable frames, reducing significantly the worst case bounded latency of highly critical or express flows.
- 802.1Qch [53] (extended profile, optional): Cyclic Queueing and Forwarding standard introduces a double buffering strategy, which allows bridges to synchronize transmissions in a cyclic manner. In each cycle, one of the buffers is in transmission mode and the other is in reception mode.
This way, the maximum delay of frames between two nodes is bounded to one cycle, assuming that one cycle is enough to empty the transmission buffer. Likewise, maximum latency across the network can be bounded with the cycle time and number of hops between nodes.
- 802.1Qcr [54] (extended profile, optional): Asynchronous Traffic Shaping (ATS) standard describes a methodology to provide reduced latency and high QoS without the need of time awareness [55], [56]. The concept is based on Urgency Based Scheduler and the implementation of a token based algorithm to assign transmission windows. According to some recent research works, ATS can be very useful in combination with CBS when handling egress queues [57].
Nowadays, the state of the art is focused on how to optimally integrate all the required TSN functionalities into IVNs providing high performance at minimum cost. In [58], authors evaluate a specific traffic management scheme that enables the integration of event-driven traffic for IVNs. There are already several solutions available in the market incorporating some of the TSN standards, and with a roadmap to be able to provide most of them. Examples of this are the Multi TSN switch Intellectual Property Core (IPCore) from SoC-e [59], TSN IPCores from Fraunhofer [60] or TTTech TSN solutions [61]. One of the biggest challenges is the management of all the configuration options and how to be able to automatically reconfigure the network. For this purpose, Software Defined Networking (SDN) technology, seems a reasonable option that can provide the required flexibility and performance. Currently, there are several research works exploring this direction [62]. Authors in [63] review TSN related research works and identify research trends in this area.

- **FR5. Quality of Service: Latency**
According to AVNU [64] and IEEE802.1DG/D1.4 [45], current IVNs require a network latency around 500 μs to 1 ms across the different hops in the network. At the same time, the new use cases to be covered by future vehicles (autonomous driving, connectivity) require more sensors integrated in the vehicle (e.g. cameras, lidar, etc.). This leads to an increase in BW usage (from Mbits or 1Gigabit to multi Gigabit Ethernet links), which makes latency requirements difficult to meet under high load situations. The goal for each of

the nodes or hops is to introduce the minimum latency possible in the data plane, becoming practically transparent, with almost no penalty in frames processing. This way the latency in the network would be mainly caused by the intrinsic traffic conflicts and availability of connection paths, that can be handled through correct traffic scheduling and management. Considering that traffic related to Autonomous Driving features will require high bandwidth and low latency at the same time (see Tables 1 and 2), there is a need for strategies on how to police this traffic in order to be able to meet such requirement. Just to give an idea of the magnitude we are talking about, the maximum latency for cut-through forwarding in a gateway should be kept below few μs. This is a hard requirement, that remarks the need for high performance and low latency computation platforms.

In order to handle the different latency requirements across different flows, the Data Distribution Service (DDS) technology appears as a good fit. DDS provides means to define, at application layer, some flow attributes such as deadline, latency budget or priority. With this, network processing platforms get the required information to make decisions in case of conflicts. There are several works on the state of the art studying the integration of DDS in automotive networks, both for Ethernet [65] and FlexRay [66]. Being a high level application, DDS needs to be combined with lower level strategies in order to guarantee the required QoS in real time. For this purpose, the combination of TSN and DDS is an interesting field, which is currently being explored in industrial networks [67], [68]. However, the suitability of DDS and TSN combined together for IVNs is still an open area which, from author's perspective, could enable the stringent requirements in terms of latency of IVN processing platforms. This is explored by authors in [69]. The most similar alternative to DDS currently available in automotive is the Service Oriented MiddlewarE over IP (SOME/IP) technology [70]. However, SOME/IP does not provide timing guarantees, which is why DDS emerges as a good candidate to fulfill the time determinism requirement.

- **FR6. Run-Time Reconfiguration**

  Software Defined Networking (SDN) techniques emerged back in 2000 with the idea of network programmability, based on the separation of control and data planes. This separation allowed networks to become scalable and easily updateable, decoupling network behavioral configuration from HW devices. In the beginning, SDN was introduced in big IT networks like data centres or enterprises. In these fields, it has been very successful and allowed the advancement of network configuration techniques exponentially.

  OpenFlow is one of the first and most widely extended technologies enabling Software Defined Networking. It emerged from a research project at Stanford University back in 2008 [71], and was presented as a tool for researchers to be able to run experiments easily over networks. The concept evolved to become a standard in today's networks configuration for packets switching in large networks.

  Nowadays, the option of migrating SDN technologies to IVNs is becoming more and more close to reality. The evolution of IVNs makes them very similar to big IT infrastructures, but with harder latency, safety and security requirements. The flexibility provided by SDN, especially in the control plane, makes it the perfect candidate for network configuration orchestration. Moreover, with the integration of TSN, the amount of configuration parameters that need to be handled, and the frequency with which those may change, make the need for SDN solutions integrated within IVNs more evident.

  In [72] an analysis of why SDN technology is interesting for the automotive industry and some applicable use cases are exposed. There are also works in the literature exploring the benefits of using SDN in IVNs [73], [74]. In [75] authors propose the use of SDN in automotive CAN-based networks while [76] proposes an SDN architecture for ethernet-based automotive networks. Authors in [77], [78] explore the integration of TSN with SDN and propose different strategies for this combination.

  Apart from TSN, safety and security are also functionalities that require high configuration capabilities, even at run time. The capability of changing the rules operating in the GW according to traffic events, the result of a diagnostic function or via an instruction in the control plane, needs to be present in future gateway platforms, e.g. update of new attack patterns in the NIDS or new filtering rules on the firewall.

  Again, SDN plays an important role in the deployment of this strategy, by providing the required mechanisms to perform such configuration updates. There are several works in the literature regarding the combination of SDN with safety and security mechanisms for automotive applications. In [79] authors propose an SDN-based strategy for fast fail over routes in the data plane. In [80] an SDN controller for safety applications is proposed and in [81] a fault-tolerant dynamic scheduling for TSN networks is introduced. Authors in [82] propose a safety-critical SDN-based dynamic reconfiguration strategy for aeronautical systems.

  Run-time reconfiguration is also needed to enable a dynamic Service oriented Architecture (SoA). A service is understood as a discrete unit of functionality that can be accessed remotely and updated independently, and applications are built by combining services. By enabling the possibility to shift some service or application from one zonal controller to another at run-time, new strategies towards network fault-tolerance emerge. For instance, in case of failure in one zonal controller, another one could take over the functionality keeping

the vehicle fully functional. At application level, a distributed SoA enables new functionalities like sharing services or applications between the vehicle and user smart-phones as explored in [83], or the interaction with the infrastructure for autonomous driving, as proposed in [84].
Overall, we see how run-time reconfiguration is a key requirement, and an enabler of many of the functionalities and technologies to be integrated in future IVNs.

- **FR7. Application-related and In-line Processing**
  Another new possibility that comes with the zonal architecture, is the capability to process application layer functionalities in the zonal gateway controllers.
  On one side, this enables to offload the HPC from this processing and to balance the load with the zonal gateway controllers available in the network.
  On the other side, it also reduces the overall latency in the application processing, since some functionalities could be resolved locally, e.g. diagnostic of a light in one zone and the corrective action associated to its status, which could save time not only on the transmission towards the HPC, but also on the translation between network protocols (in-line processing). Furthermore, having application processing capabilities in the zonal controllers, combined with the aforementioned reconfiguration requirement (FR6), enables the reallocation of functions across nodes, which can be used in case of failures or for load-balancing purposes. Eventually, zonal controllers could process all the functionalities required in the vehicle, removing the need for the HPC, enabling a fully distributed architecture.

2) STRUCTURAL REQUIREMENTS

- **SR0. Scalable**
  In order to ensure the feasibility of a future IC design, the architecture of the network processing platform must be able to extend its capabilities without major redesigns, i.e. the system architecture must be fully scalable. For example, the introduction of new input/output ports must come at no design cost. Furthermore, the introduction of new processing features must be feasible within the definition of the architecture, as long as these features are related to the previously defined FRs.
- **SR1. Flexible**
  When considering the datapath architecture, a high degree of flexibility is needed in order to support all the functional requirements at minimum cost. Therefore, the capability of the architecture of adapting to the required processing, or of accommodating it seamlessly, is an important requirement to ensure the viability of the IC development.
- **SR2. Configurable**
  The processing required in an IVN is composed of a wide variety of functionalities, ranging from the management of different protocols in different ports, to different TSN algorithms or safety features, as seen above. Hosting every possible functionality within a single chip would be a great waste of HW resources, and not even feasible since worst cases depend on each specific use case. Over-provisioning the chipset in this way is not an option since in most of the cases only a subset of the functionalities will be requested. This results in the need for highly configurable devices and architectures that permit to select the capabilities and functional features in place in order to optimize each deployment. This configuration is not only related to the functional run-time configuration listed in the functional requirements. It relates to the configuration options available in the architecture that allow for creating a range of products, from high-end to low-end segments, by selecting the required modules and the amount of each of them, based on the same architecture.
- **SR3. Stateful**
  An important aspect of IVNs is to be able to react to events based on observed data patterns. This enables capabilities such as detecting attacks or failures in the network. This means that network processing devices need to be aware of past events and therefore, the device needs to be stateful. This fact involves the need for memory elements in order to store certain information that occurred in the past and that will be considered in the present or in the future. An analysis of why statefulness is needed in network processing architectures is given in [85], reinforcing the need for this requirement.
- **SR4. Compact HW**
  The cost of area in silicon is an important metric for manufacturable ICs. Therefore, it is also an important metric of the architectures that are candidates to be the basis of future real chipset devices for automotive GWs.
- **SR5. Reasonable Complexity**
  The proposed solution shall be able to decrease the level of complexity versus other alternatives. For example, it is well-known that multi-core CPU solutions are getting more and more complex in terms of implementation and maintenance, and even debugging in development phase. However, having several cores for processing can definitely be beneficial in terms of performance. Therefore, there is a need for new solutions that allow to reduce the complexity of the existing solutions, such as the multicore architecture. This requirement emphasizes the easiness to program in terms of engineering effort and time, aimed at being realistically implementable and integrated in the future fast-paced development of automotive products. Another driver for this reduction of complexity by design is the reuse optimization.

These are the six main structural requirements selected. Other relevant parameters, like for instance low power consumption, are not evaluated due to the general lack of information available in the literature to run this benchmark.

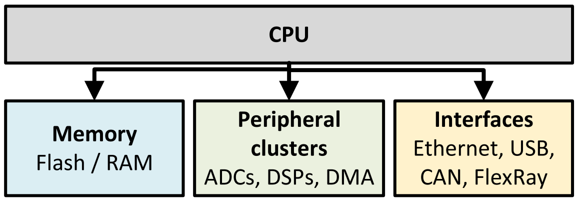


FIGURE 2. MCU-based architecture block diagram.

IV. ANALYSIS OF NETWORK PROCESSING ARCHITECTURES

In this section, an analysis of the main architectures available in the state of the art for Gateway devices is performed. We highlight the main characteristics of each of them, focusing on their specific goals, advantages and disadvantages. Furthermore, we relate the capabilities of the presented architectures to the requirements introduced in the previous sections and expose whether they are fulfilled or not.

A. MCU BASED GATEWAY CONTROLLER

Traditional Automotive GWs used to be implemented by means of a CPU that performs the routing, switching and tunneling [86] of messages between sensors/actuators and the main compute platform in the vehicle, covering all the required protocols for automotive applications. Fig. 2 depicts this architecture, where the gateway is essentially programmed on an MCU.

In this architecture, the CPU is the only processing device and is supported by dedicated memory as well as peripheral clusters and interfaces. These handle the physical layer and provide means to exchange information between the physical layer and the SW running in the CPU.

In this section A, we deal with general purpose processors that can be programmed in SW to perform networking functions, without any specific HW support for it. We start with AUTOSAR, which is the main SW architecture used in today's vehicles, both for application and network processing. Then we analyse the SW architecture of Open vSwitch and the Linux Data Plane Development Kit (DPDK), as some of the major SW-based tools used for network processing in general, although not particularly in automotive. Finally, we also review different configurations regarding CPU cores within a network processor, focusing on the many-/multi-core architecture.

1) AUTOSAR SW ARCHITECTURE

The standard SW architecture used nowadays in the design of vehicle Electronic Control Units is the AUTOmotive Software ARchitecture (AUTOSAR) [87], whose architecture is shown in Fig. 3. AUTOSAR provides a framework to develop ECUs that are compliant with automotive regulations and that provides the main traditional functional requirements of IVNs. There are two major versions of AUTOSAR: Classic and Adaptive. AUTOSAR Classic was the first version and was designed according to the needs of the traditional E/E architecture. When new requirements in terms of Service-oriented Architecture, SW updates, and flexibility in general emerged, it evolved into AUTOSAR Adaptive.

Regarding network processing, Fig. 3 highlights the support for different network protocols within the AUTOSAR communications software stack. On the lower level different drivers are available for the different protocols, which are then abstracted in the communication HW abstraction layer, and finally handled in the Protocol Data Unit Router (PDU Router). The PDU Router exchanges communication messages with the upper layer (Runtime Environment) which interacts with the Application Layer. As an example, authors in [88], propose a software stack for the management of internal and external communications of the vehicle based on the AUTOSAR Classic architecture. There are several examples of automotive specific MCUs in the market which implement the AUTOSAR Classic software stack, e.g. NXP MPC574xB-C-G.

However, when going to high BW usage and high communication speeds, the software solution is on one side loading the CPU significantly, and on the other side not able to guarantee neither timing requirements nor Quality of Service. Due to this, the functional requirements of throughput, determinism and latency are not met (FR3, FR4, FR5).

Related to the SDN concept previously introduced, the Software Defined Vehicle (SDV) appears. The focus is on a SW defined distribution of application level functionalities, following the Autosar architecture, which would allow to fully reuse the SW on different HW platforms. One example is the proposal made by ElectroKnox Corporation with their Gateway SGW1000, offering domain controller and smart central GW solutions targeting SDVs based on the Xilinx Zynq® Platform [89]. Their focus is on high-computing products that allow OEMs to adapt the changing vehicle network topology to meet the future needs of SDVs, without requiring HW changes. The focus of this solution is completely software-centric, leveraging the capabilities provided by the CPU integrated in the Xilinx Zynq SoC, with a very close low level interface to the different HW interfaces (CAN, LIN, FlexRay, Ethernet, etc) and performing all the required processing in the SoC CPU, at software level.

Another company that is currently working on the definition of SDVs is Ethernovia [90]. Ethernovia is focusing on the virtualization of Ethernet communications in order to enable the Software Defined Vehicle. These examples are a first step that highlights the trend towards software defined solutions within the automotive industry following a completely software-centric approach. However, they are not focusing on the lower level execution of those tasks and exchange of information, and therefore are not always able to meet the stringent real time and deterministic constrains imposed by future IVNs.

2) OPEN vSwitch

Following the SDN paradigm, Open vSwitch appeared in 2009 as an open source solution to orchestrate large scale networks [91]. In essence, it is a software switch running

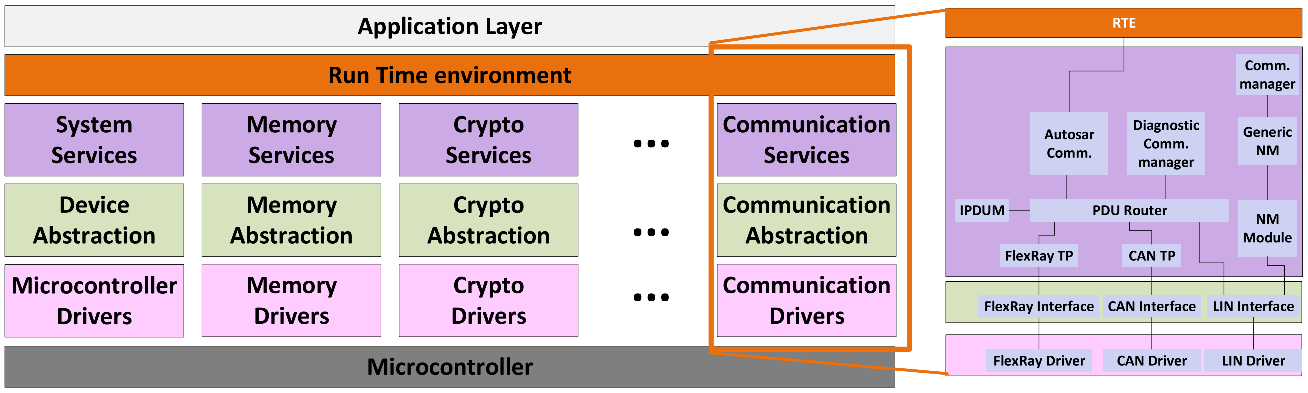


FIGURE 3. **AUTOSAR SW architecture.**

in Linux (both kernel and user space) that provides the capability to distribute the configuration and network telemetry across networks. Following the open source approach and aiming for standard compatibility, it makes use of open source configuration protocols such as NetFlow from CISCO or Link Aggregation Control Protocol (LACP).

Open vSwitch is targeted to large scale environments, providing support for network virtualization and simplifying the management and supervision of the network. Recently, there are several efforts about porting Open vSwitch to HW platforms, trying to offload some of the switch features in order to improve the performance and reduce the software complexity [92]. Additionally, Open vSwitch also incorporates some support for QoS features such as traffic policing and shaping. However, these are not oriented to provide real time determinism of packets transmission nor follow any of the TSN standards.

As of today, there is no intersection between the goals of Open vSwitch (large scale networks) and the needs of IVNs, since these are not large scale. However, as seen before, many technologies originally designed for Data Centers applications are now being integrated into vehicles, and some of the concepts behind Open vSwitch can also be of interest for IVNs. For instance, the concepts of software defined configuration and network telemetry are some of the new needs of IVNs, as derived in the requirements section (FR6).

3) DPDK

On an effort for providing a common strategy for packet processing in general, the Data Plane Development Kit (DPDK) emerged in 2010. Originally developed by Intel, DPDK has evolved into an open source project hosted by the Linux foundation. DPDK is a set of software libraries and drivers, running in userspace, that accelerates packet-processing workloads running on all major CPU architectures. Essentially, it provides a ''bridge'' for packets to skip the OS kernel space, allowing to process them in user space faster. DPDK was traditionally a full software solution, however in the recent years it has been extended to support combination with HW extensions and Smart Network Interface Cards, combining the advantages of HW acceleration and SW based programming [93]. Recently, DPDK has evolved to support TSN features, such as IEEE802.1Qbv and IEEE802.1Qav.

In [94], [95], authors propose an Environment for Generic Intra-vehicular Network Experiments (EnGINE) based on the use of Commercial Off-The-Shelf HW solutions with DPDK. The framework provides the capability of defining and executing network experiments in an easy way, as well as to collect, process and represent the experimental results. Being a fully SW oriented solution, real time performance is not optimized. However, the configurability, flexibility and scalability provided by this framework are also required in IVNS, which we keep in mind throughout this work.

4) MANY-/MULTI-CORE ARCHITECTURE

With the increase in complexity of GW controllers, handling all the processing in a single core CPU becomes too complex and results in inefficient performance.

Therefore, the natural evolution of the single CPU core architecture appears: the many-core architecture. Here, the ''divide and conquer'' principle applies, splitting the processing tasks into several processing cores which can execute tasks in parallel. In this architecture, the challenge is in how the split of processing tasks between cores is done.

One approach is to make a functional division, assigning each core to specific tasks. Authors in [96] make a proposal in this direction, defining a 3-core gateway controller. In their proposal one core is in charge of receiving and forwarding packets to the other two cores, while the other two manage the egress stage of their associated ports independently, as depicted in Fig. 4. This way frames are only stored in the

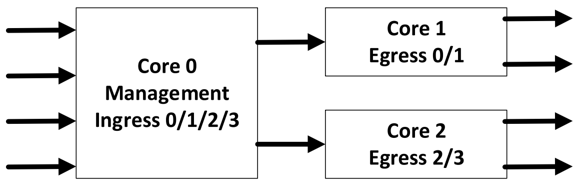


FIGURE 4. **Many-core architecture with functional distribution per cores.**

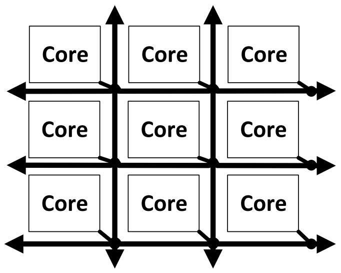


FIGURE 5. Multi-core architecture with mesh network on chip.

egress queue of the core assigned to a specific port, reducing the amount of frames to be stored in one single core. This is a simple approach which reduces the complexity of the processing performed in each of the cores. However, the resources usage will depend on the traffic patterns flowing through the system, which can result in one core being saturated while another one is free.

In order to improve the resources usability problem, processing tasks can be split among the different processing cores regardless of functionality. This means that in run-time different tasks are sent for processing to different cores on-demand, taking into account the processing load of each core, aiming at an even distribution of the processing load. In this architecture, known as Multi-core, all CPU cores are interconnected through a mesh Network on Chip that allows for fast routing of packets within the chip, and that provides also in/out connectivity for the processor, as depicted in Fig. 5. One industrial example of this architecture is the Epiphany multicore processor from Adapteva [97].

Authors in [98] perform an analysis of Worst Case Response Time (WCRT) of message processing in a multi-core gateway processor. They evaluate the WCRT with 2 cores and 4 cores, and find out via experiments that the WCRT for the two-cores implementation is below 1ms, and the WCRT for the four-cores reduces to $< 300\ \mu$s. In their analysis they also see that the 4-cores eliminates almost completely interferences between processing nodes in their use case, meaning that further cores would not bring a significant reduction in WCRT. This is related to the nature of network processing itself, which can be parallelized only up to a certain limit. Considering the real time requirement applicable in IVN gateways (FR5) where the end to end delay is bounded between 500 μs and 1 ms, we can derive that the multi-core architecture may not be able to meet this requirement. With a WCRT around 300 μs per hop, after two hops the end to end latency could be compromised depending on the application.

5) IDENTIFIED BOTTLENECKS

- **Bottleneck 0: Lack of real time performance**: To some extent, CPUs do not provide by design the required real time performance that allows to meet the stringent latency and throughput requirements imposed by the new use cases identified in future IVNs (FR3 and FR5 not met).

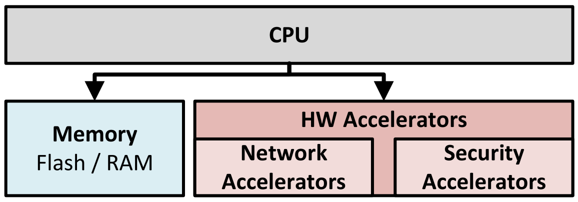


FIGURE 6. HW-offloading architecture block diagram.

- **Bottleneck 1: Lack of determinism**: additionally, they are not capable of providing the time determinism that is required in order to enable reliable and safe networks (FR2 and FR4 not met).

B. NETWORK PROCESSORS WITH HW-BASED NETWORK OFFLOADING CAPABILITIES

In order to overcome the previously mentioned bottlenecks 0 and 1, the option to combine software cores with some specific HW peripherals or accelerators emerged. The main idea is to accelerate specific functions that are identified as repetitive and time/resources consuming with a specific HW implementation for that functionality, offloading thus the CPU. This architecture is depicted in Fig. 6. As seen in the figure, the main processing device is still the CPU, however, it is now supported by specific HW accelerators which can be related to networking or other functionalities. There are several studies on the benefits of such an architecture and how to identify good candidates for HW acceleration, such as the Accelerometer study from Facebook [99], a framework for discovering Non-Conventional Inline Accelerators from IBM (NOVIA) [100] or EVITA project for embedded security functions with the HSM [101]. Networking in particular, is an area with great potential for HW acceleration because of its intrinsic characteristics. On one side, the requirements for high performance, high BW, reliability and QoS state the need for a HW implementation. On the other side, the layered structure defined by OSI model that is followed in all network protocols, establishes similarities in the processing required by most networking functionalities, allowing for the definition of custom HW that accelerates certain functions.

Network Interface Cards (NICs) emerged as an extension to general purpose CPUs for network purposes, with the main goal of increasing the device throughput capabilities (FR3). Today, they have evolved into SmartNICs, which are not only a simple interface extension, but also a processing extension for networking functionalities. SmartNICs allow to offload some network related processing tasks resulting in a smarter combination of HW and SW than the first NICs. In [102] authors analyse the performance of application specific instruction-set routers in NoC.

Big companies have also realised about the opportunity that SDN combined with HW brings to the networking world, and have started to deploy their custom solutions.

Azure Accelerated Networking (AccelNet) is the proposal from Microsoft [103] developed within the Catapult project [104]. AccelNet relies on the off-load of networking features to dedicated HW co-processors in SmartNICs in order to free CPUs of repetitive processing, gaining performance for application processing. AccelNet covers only the parsing and de-parsing engines of packet processing, but does nothing on application layers.

SDNet [105] is the proposal from Xilinx, aiming at exploiting the capabilities of SDN combined with their prototyping platforms: Field Programmable Gate Arrays (FPGAs), SoC or Multi-Processor SoC (MPSoC). However, SDNet solution is too wide to be precise and therefore, does not get to automate the full process of going from high level description to HW generation. There is the possibility of creating IP cores from a SDNet definition, but building a full system interconnecting several IP cores is not possible.

The proposal from Intel is the COnfigurable network Protocol Accelerator (COPA) framework [106] which provides a customizable framework that integrates communication with computation on an FPGA platform.

The advantages of HW implementations for switching purposes in Automotive Gateways are discussed in [107], and several proposals of HW co-processors for isolated specific tasks like [108], [109] expose the advantages of HW-SW co-design applied to Automotive Gateways. In [110] authors propose a flexible gateway platform based on the extension of the CPU processing via specific FPGA network accelerators for the different required protocols in automotive. In [111] authors discuss the benefits of SmartNICs for automotive applications. Reconfiguration and security aspects of IVNs supported by HW accelerators are also discussed in [112] and [113]. This need and opportunity has also been identified in the industry, and there are several processing platforms following this architecture: NXP S32G [114], Infineon Aurix TC4x [115], Broadcom BCM88480 [116], Marvell 88Q5050 [117], Xilinx SN1000, NVIDIA Data Processing Unit (DPU), Intel Infrastructure Processing Unit (IPU), NXP Bluebox [118], TTTech Arion IP [119].

Next, we analyse some of the most relevant industrial products for network processing following this architecture. We focus on two different kinds of products: networking chipsets provided by IC companies, and IPCores ready to integrate in an FPGA platform or in a new Application-Specific Integrated Circuit (ASIC) design.

1) BROADCOM BCM88480 PACKET PROCESSOR

One of the most recent products for Ethernet switching from Broadcom is the BCM88480 [116]. It is a SoC oriented to high-performance switching purposes with some support of HW engines for specific functionalities. Fig. 7 shows a high level block diagram of the main functional blocks included in this device for packet processing. As seen in the figure, the Network Interface delivers incoming packets to the ingress pipeline, where a Packet Processor (PP) performs Lookup operations to identify what processing needs to be done to the packet and makes the corresponding header modifications. The Lookup operations are supported by a specific HW accelerator. Then a Traffic Manager (TM) handles the schedule of

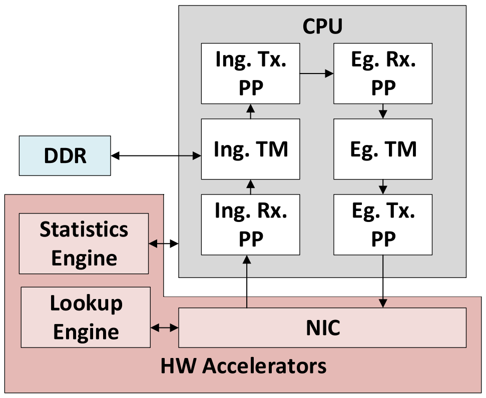


FIGURE 7. BCM88480 packet processor high level architecture.

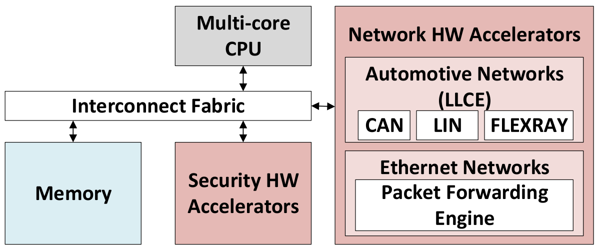


FIGURE 8. NXP S32G high level architecture.

packets storing them in external memory while they are not eligible for transmission.

The scheduling is configurable and can support different TSN standards with a software-based implementation, thus satisfying FR4. Then another packet processor delivers packets to the egress pipeline through the switching fabric. The egress pipeline performs again packet processing and traffic management according to the system configuration. The device is also supported by a HW engine for statistics purposes, providing monitoring capabilities. The device is fully focused on network switching, without any application processing capabilities considered (FR7 not met). As we can see, this proposal follows a conservative approach in terms of HW offloading with very few functionalities supported by HW accelerators, providing most of the required functionalities in software, therefore penalizing in the previously mentioned limitations.

2) NXP AUTOMOTIVE PROCESSOR WITH NETWORK ACCELERATION

NXPS32G family is one of the most complete examples of this architecture in the state of the art targeting the automotive industry. As an example, the block diagram of S32G is shown in Fig. 8. As seen in the figure, the CPU system is supported by several accelerator modules focusing on the functionalities required in IVNs such as security features and heterogeneous network management, thus fulfilling FR1 and FR2. In this case, application processing can be integrated in the system CPU, fulfilling FR7.

The network acceleration module is composed of a Low Latency Communication Engine (LLCE) supporting automotive related networks, and a Packet Forwarding Engine (PFE) supporting Ethernet Networks. Deep diving in the description of the PFE [120], we identify the specific features supported in HW. PFE permits to offload the Cyclic Redundancy Check (CRC) operations both in transmission and reception, as well as some L2 switching functions such as Media Access Control / Virtual Local Area Network (MAC/VLAN) lookup, header modification of address/port based on the lookup result and fast forwarding. It also supports traffic classification into managed/unmanaged/reserved based on user configuration, which is then used to perform ingress policy algorithms like Weighted Random Early Drop (WRED) to avoid congestion. On the egress it supports HW based scheduling algorithms such as round robin or strict priority, and it also supports the Credit Based Shaper algorithm included in TSN standards [49].

Other functionalities claimed by the PFE are supported via drivers that run in the host CPU kernel space, such as a flexible parser/router that allows for handling frames header with custom definitions and not only MAC/VLAN fields, or more advanced classification algorithms, such as Intrusion Detection and Protection Systems. When dealing with protocol tunneling (e.g. CAN to Ethernet), PFE interacts with LLCE, offloading the protocol management.

Going to the details of LLCE [121], we see that it is an MCU with 4 cores that handles the Automotive Networks. The LLCE manages the automotive-related network processing without intervention of the host CPU, but being an MCU itself, the "acceleration" comes from task isolation and HW redundancy, but does not rely on dedicated HW for this purpose. As we can see, S32G combines the HW offloading architecture with the previously discussed many-core architecture, which splits processing based on functionality.

In spite of being one of the most complete solutions available in the market, there are some key features still missing in NXP Network Acceleration, such as the support for further TSN standards (FR4), or the offloading of the current firmware based functionalities to the HW core in order to improve performance (FR5). Finally, the broad set of functionalities covered by this solution is achieved by using many MCU cores, which apart from the performance limitations previously mentioned, introduce a non-negligible silicon cost and complexity (SR5 not met). It is understood by authors that this solution provides the resources required for design exploration, however seems quite oversized for a final product, compromising the requirement for a compact HW solution (SR4 not met).

3) INFINEON AURIX TC4

Recently, Infineon announced their new Aurix TC4x processor family targeting automotive processors with a strong focus in HW network acceleration. Although today there is little information publicly available, we can infer the key features of this proposal from their website [115].

As shown in Fig. 9, TC4x provides HW accelerators for CAN and Ethernet networks, covering also the protocol tunneling in HW with the Data Routing Engine (DRE) and CAN Routing Engine (CRE) accelerators. With this, it moves a step forward from the SG32 processor from NXP, providing pure HW acceleration for protocols tunneling (FR1 and FR5 met). Similarly to SG32, application processing can be hosted in the system CPU (FR7 met).

Aurix TC4x also claims TSN support which, given that is not specified in the new proposal, we assume it is the same as the previous generation, providing support for IEEE 802.1Qav and IEEE802.1Qbv in HW, similarly to SG32 from NXP. IEEE802.1AS is also supported, however in this case the deployment is in SW (FR4 met). On the safety and security aspect, it includes a new Cyber Security Real-Time Module (CSRM) which consists on a CPU with HW support for encryption features, and a new Cyber Security Satellite which provides parallel HW accelerators to support the safety applications and provide freedom of interference (FR2 met).

The support of further TSN standards to be compliant with IEEE802.1DG seems to be excluded from this proposal, and the capabilities and configuration options of the HW accelerators, specially the ones related to traffic filtering and safety features, are not clear with the available information.

4) MARVELL 88Q5050

Marvell is another important player in the network processing industry who is also providing solutions for the automotive sector. Their 88Q5050 is a SoC for Automotive Ethernet switching purposes, with a strong focus in providing the security and TSN functionalities required within IVNs (FR2 and FR4 met). However, this product deals only with Ethernet networks (FR1 not met).

The block diagram of the 88Q5050 device is shown in Fig. 10. As seen in the figure, it is composed of a CPU with a number of HW support modules that accelerate the network processing. TSN features such as Qbv and Qav are provided

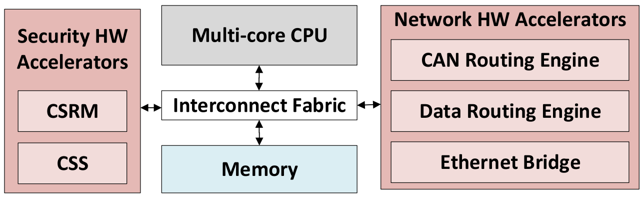


FIGURE 9. **Aurix TC4x high level architecture.**

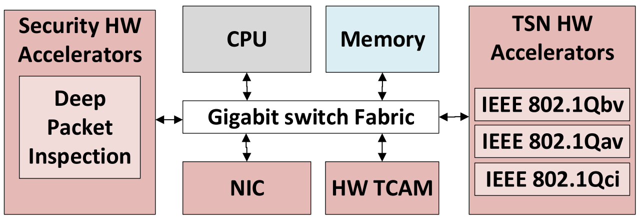


FIGURE 10. **Marvell 88Q5050 high level architecture.**

by a HW accelerator, as well as some filtering capabilities such as Ingress AVB Policy and Rate Limiting. The 88Q5050 also provides a fast switching fabric to support frames routing without the intervention of the CPU when it is not necessary, and a Ternary Content Addressable Memory (TCAM) to offload lookup operations (FR5 met).

The device has a strong focus on security aspects, and hence provides Trusted Boot technology and Deep Packet Inspection features. Fast system configuration and fast boot times are claimed as one of the main features of the device. However it seems from the available information that most of this functionalities are provided in the CPU core with a software implementation, and that little tasks are actually offloaded to the HW support module.

5) TTTech ARION IP EVALUATION PLATFORM

TTTech is one of the most active promoters of HW-based TSN solutions for network processing, in the form of IPCores to be integrated in the customer system. One of their most recent products is the Arion IP evaluation platform [119], which demonstrates their Ethernet Switch IP Technology. At the core of this product there is the Edge IP solution [122], which consists on a library of IPCores that provide the switching capabilities. Application processing is not provided by the IPCores (FR7 not met).

The support for TSN is one of the most complete in the market, including time synchronization (IEEE802.1AS), traffic scheduling (IEEE802.1Qbv and IEEE802.1Qav), frame preemption (IEEE802.1Qbu and IEEE802.3br), traffic filtering and policing (IEEE802.1Qci) and frames replication and elimination for reliability (IEEE802.1CB), fulfilling thus FR4. Even though it is a HW-oriented architecture, there is still some software support for the IPCores. Which specific functionalities are deployed in software and which are fully supported in HW is not clear from the available information. Being an IPCore based solution, the option to integrate new HW-supported functionalities is always available, extending the capabilities of the IPCores provided. However, this requires a full custom design and is limited by the available interfaces in the IPCores. Finally, in terms of usability, the solution is relatively simple since the IPCores are self-contained (SR5 met). However, this is also limiting the flexibility in terms of configuration options (SR2 not met).

6) SoC-e AVB/AUTOMOTIVE ETHERNET SWITCH IP-CORE

SoC-e provides FPGA-based Networking IP Cores and System-on-Modules. Targeting the automotive industry, they have an AVB/Automotive Ethernet Switch IP-Core and a Multi-Port TSN-Switch IPCore, with a wide selection of TSN functionalities, fulfilling FR4.

Even though this product is targeted to the automotive industry, only Ethernet is supported (FR1 not met).

In spite of being an IPCore solution, which part of the implementation is supported in HW and which part is supported in SW is not clear from the available information, since one of the requirements for the instantiation of this IPCore is to use a platform with an integrated CPU. The proposal is fully focused on the switching functionality (FR7 not met).

7) FRAUNHOFER TSN IPCores

Fraunhofer also provides several IPCores oriented to providing TSN support in FPGA or ASIC based designs. They offer a TSN switch core [123] and two TSN end-points (one switched and another one not switched) [124]. Similarly to TTTech, their coverage of TSN technologies is quite broad, fulfilling thus FR4. They provide HW-based switching enabling low-latency transmission of frames (FR5 met). However, some functionalities are still supported in SW, specially on the TSN processing.

8) IDENTIFIED BOTTLENECKS

Considering the available options in the state of the art, we identify several bottlenecks that prevent this architecture from being the best one for an automotive GW SoC:

- **Bottleneck 2: Important HW cost**: the main drawback of the HW offloading architecture is the cost increase in terms of HW associated to it, fact that needs to be properly justified and paid off through functional and performance gains. Furthermore, in the available solutions, offload engines are usually independent and disconnected, requiring a new HW module for each functionality instead of providing a configurable HW that can solve several networking functionalities (SR4 not met). Overall, we observe a certain disconnection in the datapath processing stages, or a lack of a smooth path to some extent.
- **Bottleneck 3: Limited configurability**: related to the previous item, offload peripherals usually provide few configuration options, which limits its usability and future extensions. When the peripheral is an FPGA, configurability is of course provided, however a custom HW design must be developed (SR2 not met). Usually, this shortcoming is overcome via SW-based implementations, providing in SW the functionalities that are missing in the HW offloading solution. However, with this approach, the bottlenecks identified in the previous architecture apply, and the benefits of the offloading architectures are not applicable for such features.

C. PROTOCOL INDEPENDENT SWITCH ARCHITECTURE (PISA)

In order to overcome bottlenecks 3 and 4, a new concept emerged following the SDN paradigm, which is the Protocol Independent Switch Architecture (PISA) that enables programmable data planes. The idea behind PISA architecture is to provide a common HW that enables processing of different network protocols (from Layer 2 and above, focused only on Ethernet communications) and that can be reconfigured to perform different processing tasks. Together with it, the Programming Protocol-independent Packet Processors (P4)

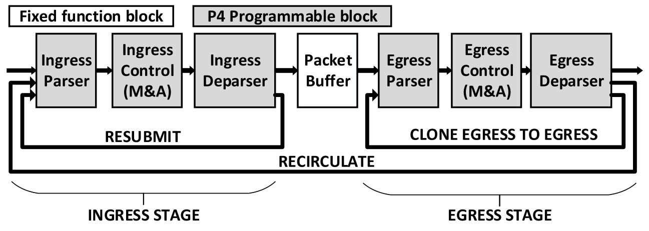


FIGURE 11. Protocol independent switch architecture block diagram.

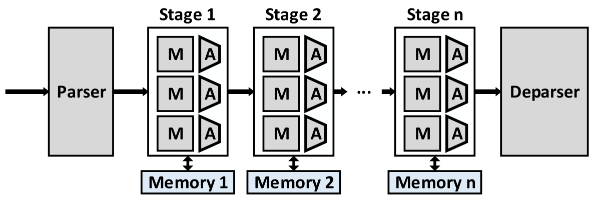


FIGURE 12. Reconfigurable match tables architectures block diagram.

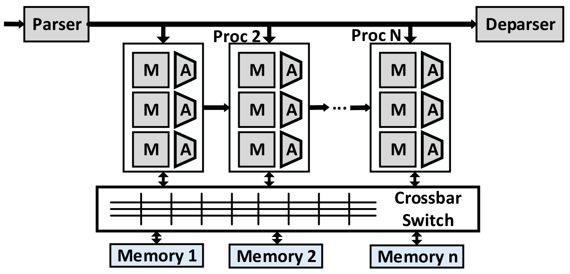


FIGURE 13. Disaggregated reconfigurable match tables architectures block diagram.

language appeared, providing the capabilities to program such a configurable device in an easy and flexible way [125].

There is an extensive collection of works dealing with P4 in the state of the art, some of them focusing on the description of P4 packet processors [126], surveys of P4 potential applications [127]–[129] performance analysis [130] or extensions to support heterogeneous dataplanes [131] as well as the definition of an abstract model for programmable data planes [132]. The capabilities to exploit application layer processing are explored by authors in [133], enabling the fulfillment of FR7.

The basic architecture of a P4 programmable switch, named Portable Switch Architecture (PSA) [134] is depicted in Fig. 11. It is composed of an ingress stage and an egress stage connected by packet buffers.

Both ingress and egress are composed of a parser and deparser, with a control stage based on a pipeline of several Match & Action (M&A) stages in between. The idea behind this architecture is that all header processing required in networking can be resolved with a series of match & action stages that can be software-defined, giving the flexibility of defining the protocol processing via software. Many of the implementations of P4 definitions are deployed in software in a regular CPU core. When exploring HW implementations, FPGAs have been identified as a suitable candidate for the prototyping of this architecture since they provide the flexiblity required at system prototyping stages. Some frameworks are available for this purpose such as P4-FPGA [135].

The advantages of the flexibility provided by this approach have been also identified by the industry, and we can already see commercial solutions enabling P4 programmability like Tofino 1/2/3 ICs from Barefoot and Intel [136], Capri from Pensando Systems [137] or NFP4000 from Netronome [138]. Some of these solutions integrate the PISA pipeline via software and some of them implement it in hardware, trying to optimize the performance and taking advantage of the reconfigurable match and action concept.

Next, we analyse different implementations or variations of the PISA architecture. First we analyse the currently available options for implementing the Match and Action processing stages, that are at the core of PISA architecture. Then, we review two industrial products based on the PISA architecture: Intel Tofino and Netronome NFP4000. Finally we also analyse two different proposals for PISA architecture coming from Academia: the first one is a heterogeneous architecture combining a CPU with a Graphics Processing Unit (GPU), and the second one presents a folded pipeline architecture which, although conceived before PISA architecture itself, introduces interesting concepts that are very relevant and up to date in PISA architecture.

1) MATCH AND ACTION ARCHITECTURES

The most popular architecture for the M&A stages is the **Reconfigurable Match-Action tables (RMT)** architecture [139], depicted in Fig. 12. RMT is composed of a pipeline of several stages of M&A tables, where each of them has its dedicated access to memory and allows for performing the required packet processing sequentially. Regarding flexibility, although RMT allows to "extend" Match tables to different stages when there are not enough table entries in one stage, in that case action units are wasted, and it is not possible to skip stages when they are not needed, penalizing with the full pipeline latency all frames, regardless of how simple their required processing may be. These issues have been addressed by the **disaggregated RMT (dRMT)** architecture introduced in [140], whose architecture is depicted in Fig. 13. dRMT replaces the M&A stages with run-to-completion processors that can be assigned M&A tasks in any order. In addition, memory is not individually assigned to individual stages/processors, but shared across all of them and accessed via a crossbar switch. This way, packets will only traverse as many stages as required by each of them, and can be assigned to several processors in parallel when the M&A operations are independent, reducing latency for simple processing.

Another limitation of RMT architecture is the lack of state awareness, i.e. RMT is stateless. This has been identified in the state of the art, and there are available solutions like the proposal from authors in **FlowBlaze** [141]. Mainly, the proposal is to enable the introduction of stateful elements in

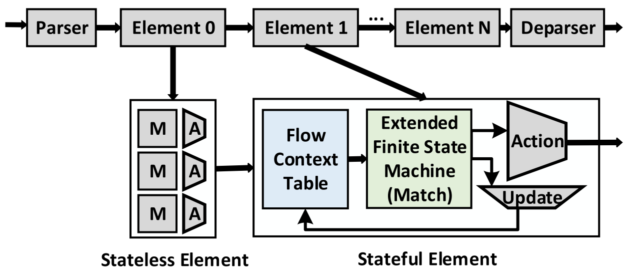


FIGURE 14. FlowBlaze architecture block diagram.

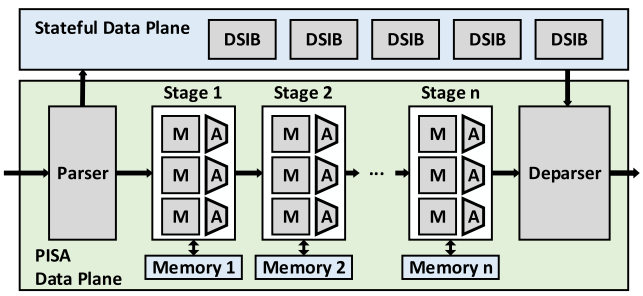


FIGURE 16. Extension of PISA architecture with stateful data plane.

the PISA pipeline, as depicted in Fig. 14. Therefore, combining stateless and stateful elements, the P4 flexibility can be kept and enhanced with statefulness, enabling the update of the processing itself depending on the identified states. These states are determined by the received traffic, therefore, this solution permits to update the processing according to traffic patterns and events detected. The main drawback is the additional HW resources and latency introduced for the sake of statefulness. Another proposal to extend the PISA pipeline with statefulness is introduced by authors in [85]. In this case, the proposal is to reuse the PISA data plane as it is, and to extend the architecture with a **Stateful Data Plane** which enables state awareness, as depicted in Fig. 16.

In this case, there is no extra latency added to the data plane processing, since the state is processed in parallel.

PSA provides several packet paths that somehow permit to overcome the rigidity of the fixed M&A pipeline architecture (either RMT or dRMT). For recursive processing, the recirculate path depicted in Fig. 11 allows for sending packets to the ingress stage after going through the egress stage if required. However, this recirculation presents several shortcomings, as identified in [142]. First, forwarding of a packet to several ports is not allowed, hence a packet is either forwarded to egress ports or to the ingress again, but not to both in parallel. Second, when re-sending packets to ingress the pipeline cannot accept new ingress frames since parallel processing is not supported in PSA, increasing the latency.

To overcome the limitations in parallel forwarding the clone egress to egress or resubmit paths are included, permitting to make copies of frames which can be processed again and finally forwarded where required. However, this approach is, again, penalizing in latency. Another alternative trying to somehow overcome the rigidity of the fast-forward pipelining has been proposed by authors in [143]. The main idea is to interconnect the M&A stages of different pipelines in a matrix, allowing for redefining the path followed by packets in a programmatic way, supporting **network function virtualization (NFV)**. The architecture corresponding to this proposal is depicted in Fig. 15. This alternative permits to virtually define different paths combining pipeline resources allowing for better configurability, more complex processing and to perform parallel processing to some extent. However, in this proposal the resources allocation is not done in run-time and therefore may result in an inefficient use of the resources available in the system. Furthermore, recirculate paths are not considered in the proposal, sharing the limitations stated above with the other architectures presented. From our perspective, the flexibility provided by P4 programmability allows for defining the required network processing with a SW-based approach. However, the deployment in the architecture presents critical limitations when it comes to processing that cannot be solved with fast-forward M&A stages and imposes hard penalties to overcome these limitations, resulting in inefficient performance and/or HW resources usability, as pointed out by authors in [144], [145].

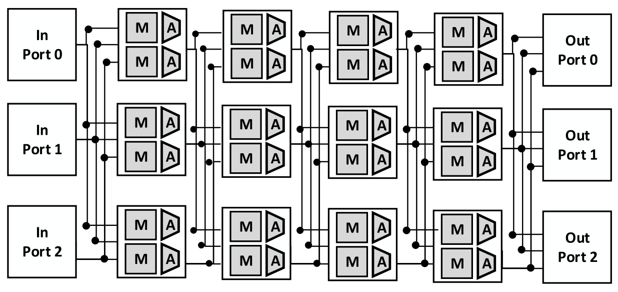


FIGURE 15. Packet processor architecture for network function virtualization.

2) INTEL TOFINO

One example of industrial processor following this architecture is Tofino from Barefoot, and its evolved versions Tofino 2 and Tofino 3 from Intel [136]. The Tofino Native Architecture (TNA) based on the PSA architecture is depicted in Fig. 17. In this case, the interconnection between ingress and

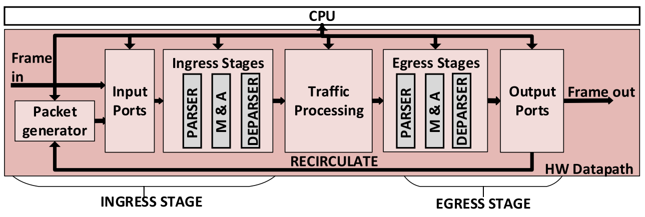


FIGURE 17. Tofino native architecture block diagram.

egress stage is a bit more complex than simple buffers, allowing interconnection with the CPU to perform any processing that cannot be resolved within the M&A stages. A packet generator is also included in TNA allowing to inject traffic in the network coming from the CPU. Regarding the Match & Action stages, Tofino uses the RMT architecture described above, inheriting its limitations.

The recirculate path of PSA is also present in Tofino, allowing to perform recursive processing, although with the previously mentioned limitations. The packet generator is also in charge of handling the recirculated frames and re-injecting them in the pipeline.

3) NETRONOME NETWORK FLOW PROCESSOR

Another industrial processor that provides support for PISA-based programmable data planes is the Network Flow Processor (NFP) from Netronome [138]. NFP is a SmartNIC with a PISA-based offload engine for flexible packet processing. The architecture of Netronome NFP is depicted in Fig. 18. As seen in the figure, frames coming in the processor are first directed to a Packet Processor Core (PPC) island, which consists of an MCU supported by HW accelerators to perform small packet modifications and forward the traffic.

After this initial processing, the distributed switch fabric permits to send the header and metadata of a frame to a Flow Processing Core (FPC) island, while the payload is stored in one of the available memory units. Each FPC island is composed of a number of FPCs, which are custom 32-bit cores where P4 instructions can be executed to perform the required packet processing. On completion, the distributed switch fabric recovers the payload of the frame from the memory and delivers the recomposed frame either to the host CPU via PCIe interface, or to the egress PPC for direct transmission. As we can see from the architecture, Netronome NFP mixes several of the previously described architectures. On one side it follows a many-core architecture with functionality distribution, allocating different functionalities to different cores, however it also provides multiple cores for each of the functionalities, in the so-called islands. Furthermore, it also provides HW offloading in some of the cores with specific HW accelerators such as cryptography functionalities, as seen in Fig. 18.

Authors in [146] analyse NFP-4000 SmartNIC limitations. They focus on identifying which are the limitations of the PISA architecture as an offload accelerator from a holistic perspective, accounting for the time to offload frames. With this approach, they are able to provide realistic insights on the performance of this architecture. Their study highlights the negative impact in latency of the PISA pipeline depth, finding that for more than 5 M&A stages, both latency and throughput are affected significantly.

This suggests that, although PISA architecture is potentially flexible and could allow complex processing to be performed through an unlimited number of cascade stages, in practice this is limited. In other words, there is a finite (and not very high) number of stages that can be pipelined

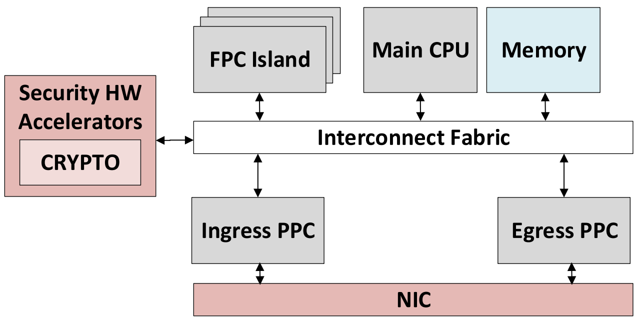


FIGURE 18. Netronome network flow processor high level architecture.

without affecting the device metrics. A solution to this shortcoming could be to reuse the pipeline stages via recirculate paths. However, they also identify a limitation related to the recirculate paths, where custom metadata cannot be recirculated with frames, restricting the deployment of complex algorithms. Finally, they also analyze the impact of performing basic cryptography functions, such as Cyclic Redundancy Check (CRC), and find out that performing more than 10 of these operations results in a noticeable latency increase, affecting the capabilities to fulfill the safety requirements of Gateway controllers.

From our understanding, the specific numerical findings of [146] are dependent on the particular use cases run on the NFP-4000 as well as the NFP-4000 device itself. However, they give a good grasp of what a PISA-based network processor can achieve and which are the limitations. In this particular case, the PISA pipeline is deployed in software, so it seems reasonable to think that a HW implementation would push the identified limits to bigger thresholds. However, due to the previously mentioned architectural limitations the same problems would be found in a larger scale.

4) CPU-GPU HETEROGENEOUS ARCHITECTURE

On an effort to scale the performance of P4-based data planes, authors in [147] propose an heterogeneous architecture based on the combination of a CPU with a Graphics Processing Unit (GPU). Their main goal is to exploit the high performance parallel computation offered by GPUs and propose a toolchain to deploy P4 programs to the CPU-GPU architecture. Mainly, the Match & Action Tables are implemented in the GPU, while parser/deparser functionalities are run in the CPU. They also present latency hiding techniques to overcome the problem of data transmission between the CPU and the GPU by pipelining the execution of several Match & Action stages over different frames.

This proposal potentially scales the performance of PISA architecture by betting on an aggressive acceleration of the offloaded processing, achieving a throughput of hundreds of Gbps. However, it inherits the previously mentioned limitations of the fast-forward pipelining architecture and comes at an important HW cost and power consumption due to the GPU integration.

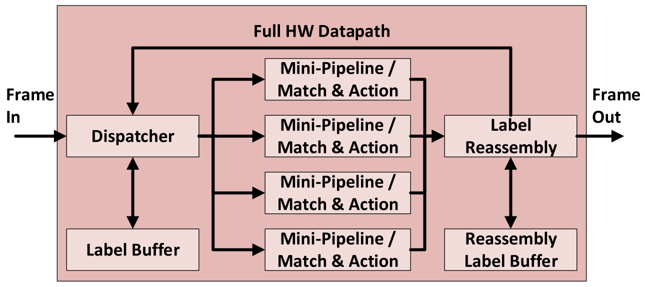

Full HW Datapath
Frame In
Dispatcher
Label Buffer
Mini-Pipeline / Match & Action
Mini-Pipeline / Match & Action
Mini-Pipeline / Match & Action
Mini-Pipeline / Match & Action
Label Reassembly
Reassembly Label Buffer
Frame Out

FIGURE 19. Folded pipeline network processor architecture.

5) FOLDED PIPELINE NETWORK PROCESSOR ARCHITECTURE

Targeting the rigidity problems of fast-forward pipeline architectures for network processing, authors in [148] propose a Folded Pipeline Network Processor Architecture, which is depicted in Fig. 19. Prior to the appearance of P4 and PISA, this work defined "Mini-Pipelines" which can be understood as the M&A stages of the network processor.

The HW defined for the Mini-Pipelines was focused on the processing of Multi Protocol Label Switching for Ethernet frames, and the number of available Mini-Pipelines was set to 4 for the evaluated use cases.

One interesting contribution of this work is the ability to "recirculate" frames when they need to traverse more than one Mini-Pipeline, allowing to adjust the data-path to the processing required by each frame, removing thus the rigidity of the fast-forward pipelines and improving the usability of the available HW resources.

6) IDENTIFIED BOTTLENECKS

Considering the presented analysis of this architecture and the previous findings in the state of the art, we identify several bottlenecks that prevent it from being the optimal architecture for an automotive GW SoC:

- **Bottleneck 4: Only Ethernet oriented**. The architecture does not provide the heterogeneity required in automotive devices (FR1 not met).
- **Bottleneck 5: Stateless**. The architecture does not allow to adapt the data plane configuration to events or traffic patterns detected during run time (SR3 not met in general, solved in FlowBlaze and Stateful Data Plane only).
- **Bottleneck 6: Limited scalability**. Pipeline depth affects all frames equally (except in dRMT and folded pipeline architectures), imposing a strong penalty to frames that do not require complex processing (SR0 not met). In practice the number of stages also affects the capabilities of the offload engine, reducing its growth capabilities.
- **Bottleneck 7: Limited flexibility**. Once the pipeline is defined, there are very few options to overcome the rigidity of the architecture (recirculate paths in folded pipeline, matrix interconnection in NFV processor) and are not sufficient to provide/increase the required functionalities in terms of safety or determinism (SR1, FR2 and FR4 not met).
- **Bottleneck 8: Non-optimum usability of HW resources**: In this architecture the pipeline depth has to be over-dimensioned in order to cope with worst case processing requirements. Furthermore, extra latency is introduced when aiming at reusing the resources via recirculate paths since the whole pipeline needs to be traversed in every loop (SR4 not met in general, solved only in the folded pipeline architecture).

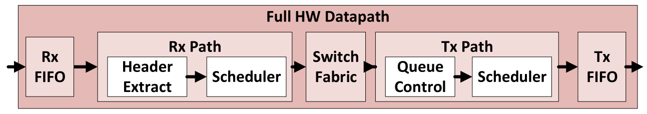


FIGURE 20. VEGa architecture block diagram.

D. VEHICULAR GATEWAY ARCHITECTURE (VEGa)

A different approach is followed in the Vehicular Gateway Architecture (VEGa) proposed by authors in [107]. The VEGa architecture is focused in providing a low-latency switching platform for IVNs (FR5 met), targeting the heterogeneity of protocols present in these networks (FR2 met). However, the maximum throughput for which it is designed is 1Gbps, which is below the throughput requirements we foresee for future IVNs (FR3 not met). Being focused only on the switching functionality, application processing is not considered (FR7 not met). The high level architecture of VEGa is depicted in Fig. 20. As shown in the figure, VEGa also separates the ingress and egress stage with an interconnection fabric similarly to the PISA architecture. However, inside those stages the architecture is different, completely customized for the specific use case considered by authors.

On the ingress or reception path, there is dedicated logic for the processing of Ethernet frames, with some custom extensions for the processing of FlexRay or CAN frames. Afterwards an scheduler organises the traffic depending on the result of the header extraction phase. On the egress or transmission path, there is dedicated logic for queue control and traffic scheduling, handling the delivery of frames to the egress ports. Although not explicitly stated by authors in [107], it is our understanding that such a stage would be suitable to allocate TSN scheduling algorithms, fulfilling FR4, modifying the custom scheduler logic. Finally, the simplicity of this architecture allows to reach the required level of performance at a reasonable HW cost (SR4 met).

1) IDENTIFIED BOTTLENECKS

There are some limitations that prevent VEGa from fulfilling all the previously gathered requirements, some of them already identified in previously presented architectures:

- **Bottleneck 3: Limited configurability**. Although some options for Ethernet header processing are offered, the configurability required in a future GW platforms goes

TABLE 3. Network processing architectures analysis - summary.

Architecture	FR0	FR1	FR2	FR3	FR4	FR5	FR6	FR7	SR0	SR1	SR2	SR3	SR4	SR5
MCU-based														
Classic Autosar [87]	✓	✓	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✗
DPDK [93]	✓	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✗
Open vSwitch [93]	✓	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✗
Many-/Multi-core [96], [97]	✓	✓	✗	✗	∼	✗	✓	✓	✓	✓	✓	✓	✓	✗
HW-offload														
Broadcom BCM88480 [116]	✓	✗	✗	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓	✗
NXP-S32G [114]	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✗	✓	✗	✗
Infineon Aurix TC4 [115]	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✗	✓	✗	✗
Marvell 88Q5050 [117]	✓	✗	✓	✓	✓	✓	✗	✓	✓	✗	✗	✓	✗	✗
TTTech Arion IP [119]	✓	✗	✗	✓	✓	✓	✗	✗	✓	✗	✓	✓	✗	✓
SoC-e Switch IP Core [59]	✓	✗	✗	✓	✓	✓	✗	✗	✓	✗	✓	✓	✗	✓
Fraunhofer TSN IPCores [124], [123]	✓	✗	✗	✓	✓	✓	✗	✗	✓	✗	✓	✓	✗	✓
PISA [134]														
RMT [139]	✓	✗	∼	∼	∼	∼	✗	∼	✗	✗	✓	✗	✗	✗
dRMT [140]	✓	✗	∼	∼	∼	∼	✗	∼	✓	✗	✓	✗	✗	✗
FlowBlaze [141]	✓	✗	∼	∼	∼	∼	✗	∼	✗	✗	✓	✓	✗	✗
Stateful Data Plane [85]	✓	✗	∼	∼	∼	∼	✗	∼	✗	✗	✓	✓	✗	✗
Packet Processor for NFV [143]	✓	✗	∼	∼	∼	∼	✗	✗	✓	✓	✓	✗	✗	✗
Tofino [136]	✓	✗	∼	∼	∼	∼	✗	✓	✗	✗	✓	✗	✗	✗
Netronome NFP [138]	✓	✗	∼	∼	∼	✗	✗	✓	✗	✗	✓	✗	✗	✗
CPU-GPU [147]	✓	✗	∼	∼	∼	✓	✗	✓	✗	✗	✓	✗	✗	✗
Folded Pipeline [148]	✓	✗	∼	∼	∼	✓	✗	✗	✓	✓	✓	✗	✓	✗
VEGa [107]	✓	✓	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓	✓

FR0. Generic Network Processing, FR1. Heterogeneous Networks management, FR2. Safety and Security, FR3. Throughput
FR4. Quality of Service: Determinism, FR5. Quality of Service: Latency, FR6. Run-time reconfiguration, FR7. Application and inline processing
SR0. Scalable, SR1. Flexible, SR2. Configurable, SR3. Stateful, SR4. Compact HW, SR5. Reasonable complexity

beyond that, including more than one option for protocols. Moreover, the extension/modification of protocols as well as the configuration of frames processing and manipulation rules are required, and not provided (SR2 not met).

- **Bottleneck 7: Limited flexibility**. VEGa focuses in processing only Ethernet frames headers, with specific support for CAN and FlexRay, but using a very specific HW design. Further protocol modifications would probably require a major redesign of the custom logic (SR1 not met).
- **Bottleneck 9: No run-time reconfiguration**: VEGa does not provide means to change the configuration while running, and therefore does not allow to adapt the system to the incoming traffic, failing to meet FR6.

E. SUMMARY

As seen along this section, all of the analysed architectures for network processing have some advantages and disadvantages. Together they provide the different requirements needed for an IVN computing platform. However, none of them is able to fulfill the whole list of requirements.

A summary of the coverage of these requirements by each of the architectures is exposed in Table 3. Note that, to simplify the table, we mark features as provided when a reasonable set of the sub-features discussed for each section is provided, e.g. for TSN support, we mark S32G and Aurix TC4 as "supported" since they provide some of the most typical TSN features, although not all of them as detailed in the previous sections.

When requirements are not applicable, we mark them as ∼, e.g. TSN support within MCU-based multi-core architecture will depend on the software running, which could be AUTOSAR (not supported today) or DPDK (supported).

Looking at the table, we see that the different groups of architectures tend to be more effective at fulfilling either the functional or structural requirements. For example, MCU-based architectures provide most of the structural requirements but struggle to meet some of the most demanding functional requirements, especially the ones related to Quality of Service (FR3, FR4, FR5), even when many-/multi-core architectures are used.

On the other side, architectures targeting the automotive use case such as the industrial examples of the HW-offload architecture, or VEGa, provide almost all the functional requirements but fail at many of the structural requirements, especially at flexibility, scalability, HW cost and complexity (SR0, SR1, SR4, SR5).

PISA architecture is somewhere in between, trying to provide more functionality and performance keeping key structural aspects (configurability in this case, SR2), and in exchange sacrifice other structural requirements such as scalability and HW cost (SR0, SR4).

Some of the structural requirements can be fulfilled with the PISA modifications analyzed (statefulness with FlowBlaze, flexibility with dRMT, etc.) but still scalability and resources consumption issues persist.

This behavior corresponds to the typical compromise between general-purpose and customized platforms where functional and structural requirements push the architecture design in different directions. However, from authors perspective, this is precisely what needs to change in order to advance towards the network processing platforms of the future, as we discuss in the next section.

V. CONCLUSION

In this work, we provide an in depth analysis of state of the art and future challenges of automotive network processing devices. We start by analysing the trends within the automotive industry, the changes in the vehicular E/E architecture (from function-based to domain-based to zonal-based) and how they affect the IVN.

We derive a set of requirements that need to be fulfilled by future IVN processing platforms in order to successfully satisfy the future industry demands. We look at requirements from two perspectives: (i) functional requirements that correspond to the functionalities or features that need to be supported by Gateway controllers and (ii) structural requirements that correspond to design and architecture aspects that ensure the viability of the Gateway as a product.

Apart from collecting the list of requirements, we discuss the reasoning behind each of them, as well as the available technologies that can be key enablers, even if they have not been used in automotive industry yet. With this, we aim at providing a comprehensive understanding of the automotive in-vehicle networking use case as well as the technologies involved today, or in the future.

Second, we review the main state of the art architectures for network processing, looking at their internal architecture details. We analyse examples available as industrial products in the state of the art and also provide the most recent findings in the literature regarding network or packet processing architectures. In this review we show how each of the architectures complies (or not) with the previously inferred requirements and highlight the main bottlenecks of each of them.

Finally, we provide a summary of this analysis in Table 3 as a survey and a taxonomy of related research works in Section VI. From our analysis, we are able to extract the following conclusions:

- **Current solutions do not solve the problem completely**.
  As we have seen, none of the solutions currently available provides both functional and structural requirements at the same time.
  This results in limitations on what can be achieved in a GW chipset for a vehicular network, restricting performance and functionality or flexibility and scalability of the products.
- **High performance solutions are missing**.
  As an outcome of this analysis, we see that there is a lack of high performance solutions combining both functional and structural requirements. Based on the analysis shown before, we can depict the landscape of available network processing solutions and map them in a 2D graphic, as done in Fig. 21. On the X axis we have the functional requirements and on the Y axis we have the structural requirements.
  As seen before, the state of the art moves between MCU-based solutions on the mid-left side of the graph, and custom HW solutions which reside on the lower right side of the graph. In between, PISA-based or HW-offloading solutions fill the mid and left areas of the landscape.
  However, there are no solutions residing on the top right side of the graph, which points to a lack of high performance solutions able to provide the maximum of both functional and structural requirements, which are needed to meet the demands of future vehicles.
- **Functional and Structural requirements need to grow together**.
  The only way to achieve a successful solution for future automotive in-vehicle network processing is to maintain a balance between functional and structural requirements.
  From our perspective, by analysing very carefully the functionalities that need to be integrated in the solution and prioritizing structural aspects at the same level as functional, the sweet spot between these two dimensions could be defined.
- **Hardware accelerators can be an enabler of this paradigm**.
  We identify HW-based system architectures as the right paradigm that will enable this high performance solutions, especially for autonomous driving related functions where time determinism quite often becomes a hard real-time requirement as consequence of functional safety implications.
  Given that best performance is achieved with HW support, and performance requirements will do nothing but increase, it is our understanding that the work resides on how to provide these HW capabilities maintaining the structural aspects at a reasonable level for the application. This combined with the aforementioned functionality analysis can be the best roadmap towards the future of network processing solutions.

To conclude, we find an interesting research opportunity that emerges from the revolution that is currently undergoing both in automotive and semiconductor industry. We expect this work to contribute to the evolution of the current landscape, where we see a gap pointing to a lack of high performance solutions for automotive network processing. We also

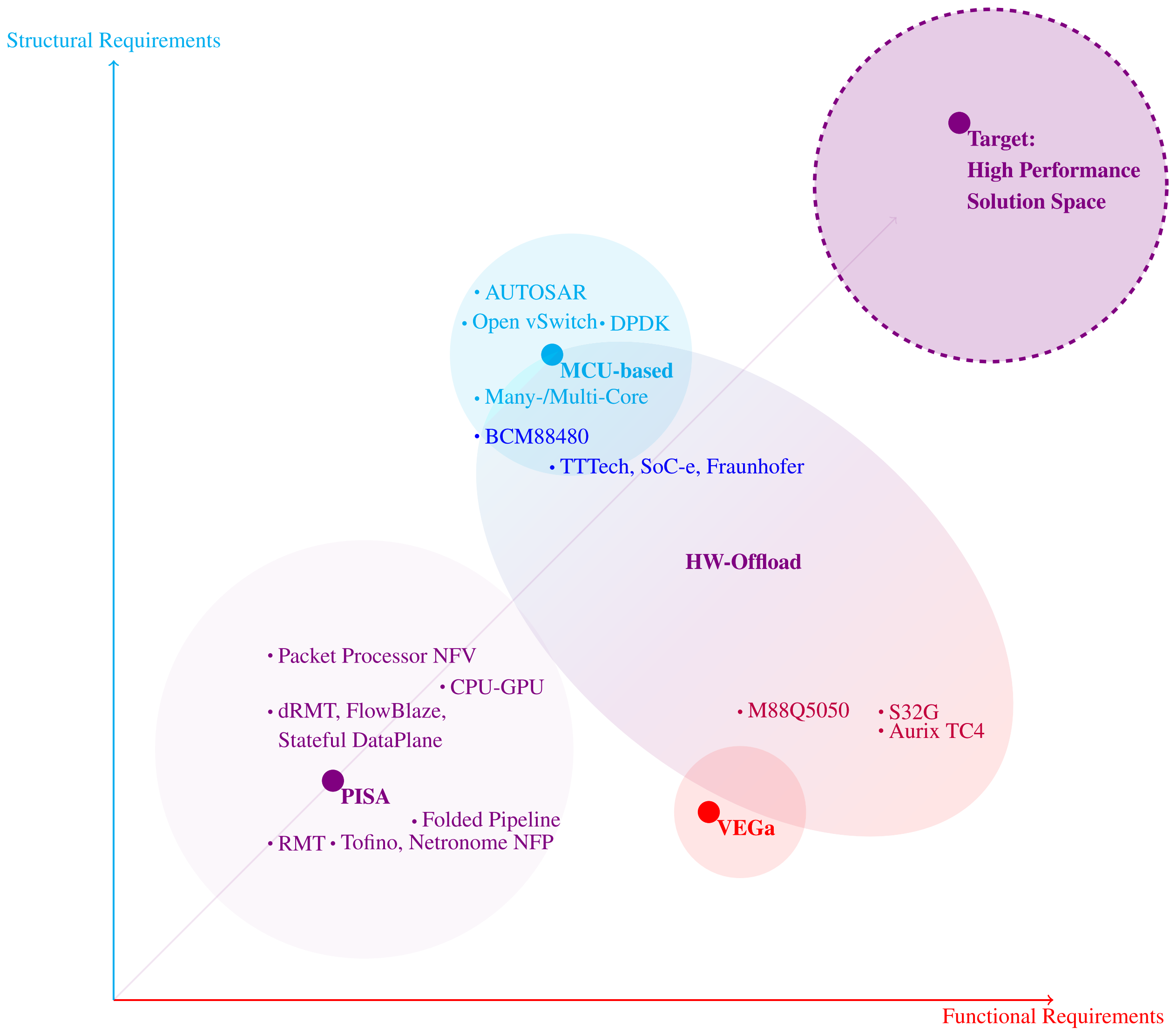


FIGURE 21. Network processing architectures analysis - landscape overview.

identify HW-centric architectures as an enabler for these future solutions.

With this, we hope that this analysis can bring some insights and encourage further works on this direction in the community, and that we can see a solution filling the gap in the near future.

VI. TAXONOMY OF RELATED WORK

For completeness, in this section we present a taxonomy of the main research papers discussed throughout this work. This taxonomy is split between works related to automotive electronics industry (Fig. 22) and works related to HW acceleration (Fig. 23).

Fig. 22 starts with In-Vehicle Network architectures, their foundations and evolution, together with experiments and frameworks. Then it continues with the wide variety of technologies that are of interest for IVNs as previously analyzed in this work: management of heterogeneous networks, safety and security, Time Sensitive Networking, Data Distribution Services, Software Defined Networking, Programmable Data Plane and the interactions of two or more of these technologies within IVNs.

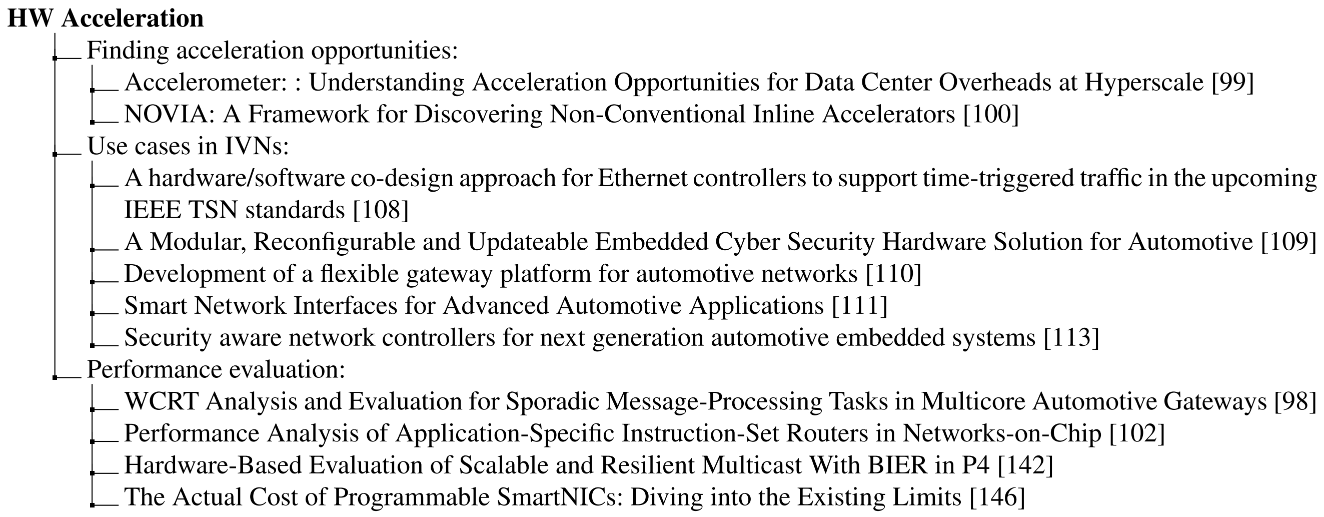
Fig. 23 goes from methods to identify HW acceleration opportunities to HW acceleration use cases in IVNs and performance evaluation of HW offloading architectures.

This classification, together with our discussion about requirements and architectures analysis, provides a complete and thorough overview of network processing platforms for future IVNs. With this taxonomy, we aim at facilitating the task of gaining broad knowledge in the topic both for experts and newcomers, softening the learning curve and accelerating research on the matter.

Automotive Electronics Industry

- **IVN architecture:**
  - Foundations:
    - Future Vehicle Networks and ECU [10]
    - Intra-Vehicle Networks: a Review [11]
    - Next Generation Intra-Vehicle Backbone Network Architectures [12]
  - Experiments:
    - An Architecture for In-Vehicle Networks [13]
    - Environment for Generic In-vehicular Network Experiments - EnGINE [94], [95]
- **IVN Technologies (present and future)**
  - Heterogeneous Networks Management:
    - In-Vehicle Networks outlook [22]
    - Extensible Multiprotocol Gateway [24]
    - Heterogeneous Communication Virtualization [25]
  - Safety and Security:
    - Automotive Ethernet in On-Board Diagnosis & in-vehicle networking [26]
    - Cooperation or competition? Coexistence of safety and security in next-generation Ethernet-based automotive networks [42]
    - Secure Automotive On-Board Electronics Network Architecture [101]
  - TSN:
    - Assessments of Real-Time Communications over TSN Automotive Networks [58]
    - Timely Survey of Time-Sensitive Networking: Past and Future Directions [63]
  - DDS and DDS with TSN:
    - DDS middleware on FlexRay network [66]
    - Multi-Level Time-Sensitive Networking Using the Data Distribution Services for Synchronized Three-Phase Measurement Data Transfer [67]
    - Using DDS over TSN to support NATO Generic Vehicle Architecture for Land Systems [68]
  - SDN:
    - OpenFlow [71]
    - Software-Defined Networking in Automotive [72]
    - An In-Vehicle Software Defined Network Architecture for Connected and Automated Vehicles [73]
    - In-Vehicle Software Defined Networking: An Enabler for Data Interoperability [74]
    - Retrofitting SDN to classical in-vehicle networks: SDN4CAN [75]
    - An SDN Architecture for Automotive Ethernet [76]
  - SDN with TSN:
    - SDN-based configuration solution for IEEE 802.1 Time Sensitive Networking [62]
    - Software-Defined Networks Supporting Time-Sensitive In-Vehicular Communication [77]
    - Software-Defined Time Sensitive Networks Configuration and Management [78]
  - SDN for safety applications:
    - Fault-Tolerant Dynamic Scheduling and Routing for TSN based In-vehicle Networks [81]
    - Dynamic Network Reconfiguration in Safety-Critical Aeronautical Systems [82]
  - Service oriented Architecture:
    - A distributed in-vehicle service architecture using dynamically created web Services [83]
    - SODA: Service-Oriented Architecture for Runtime Adaptive Driver Assistance Systems [84]
  - Programmable Dataplane:
    - A Survey on Data Plane Programming with P4 [127]
    - An Exhaustive Survey on P4 Programmable Data Plane Switches: Taxonomy, Applications, Challenges, and Future Trend [128]
    - The programmable Data Plane: Abstractions, architectures, algorithms, and applications [129]
    - Modeling and Performance Analysis of P4 Programmable Devices [130]
    - P4 to FPGA-A Fast Approach for Generating Efficient Network Processors [135]
    - One for All, All for One: A Heterogeneous Data Plane for Flexible P4 Processing [149]

FIGURE 22. Taxonomy of related work on automotive electronics industry.



HW Acceleration

- Finding acceleration opportunities:
  - Accelerometer: : Understanding Acceleration Opportunities for Data Center Overheads at Hyperscale [99]
  - NOVIA: A Framework for Discovering Non-Conventional Inline Accelerators [100]
- Use cases in IVNs:
  - A hardware/software co-design approach for Ethernet controllers to support time-triggered traffic in the upcoming IEEE TSN standards [108]
  - A Modular, Reconfigurable and Updateable Embedded Cyber Security Hardware Solution for Automotive [109]
  - Development of a flexible gateway platform for automotive networks [110]
  - Smart Network Interfaces for Advanced Automotive Applications [111]
  - Security aware network controllers for next generation automotive embedded systems [113]
- Performance evaluation:
  - WCRT Analysis and Evaluation for Sporadic Message-Processing Tasks in Multicore Automotive Gateways [98]
  - Performance Analysis of Application-Specific Instruction-Set Routers in Networks-on-Chip [102]
  - Hardware-Based Evaluation of Scalable and Resilient Multicast With BIER in P4 [142]
  - The Actual Cost of Programmable SmartNICs: Diving into the Existing Limits [146]

FIGURE 23. Taxonomy of related work on HW acceleration for IVNs.

REFERENCES

[1] *Five trends transforming the Automotive Industry*, McKinsey & Company, Atlanta, GA, USA, 2018.

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

[3] *Taxonomy and Definitions for Terms Related to on-Road Motor Vehicle Automated Driving Systems*, Standard SAE, J3016, 2019.

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

[5] *Tesla New Self Driving Chip is Here and This is Your Best Look Yet*, Verge, New York, NY, USA, 2020.

[6] *Google Reveals the Mysterious Custom Hardware That Powers AlphaGo*, Verge, New York, NY, USA, 2020.

[7] *Deep Dive into Amazon Inferentia: A Custom-Built Chip to Enhance ML and AI*, Cloud Manage. Insider, Dover, DE, USA, 2020.

[8] *Facebook Joins Amazon and Google in AI Chip Race*, FinantialTimes, London, U.K., 2020.

[9] *Apple is Switching Macs to its Own Processors Starting Later This Year*, Verge, New York, NY, USA, 2020.

[10] L. V. Dijk, "Future vehicle networks and ECUs—Architecture and technology considerations," NXP Semicond., Chandler, AZ, USA, Tech. Rep. FVNECUA4WP REV 0, 2017.

[11] 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, Apr. 2015.

[12] 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.

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

[14] H. Askaripoor, M. H. Farzaneh, and A. Knoll, "E/E architecture synthesis: Challenges and technologies," *Electronics*, vol. 11, no. 4, p. 518, Feb. 2022.

[15] *La Camisa de Fuerza de Intel y el Viaje Hacia Los Apple Silicon*, Apple, Cupertino, CA, USA, 2020.

[16] *Road Vehicles Controller Area Network (CAN)*, Standard ISO 11898, International Organization for Standardization (ISO), 2015.

[17] *LIN Specification Package Revision 2.2A*, LIN Consortium, Dallas, TX, USA, 2010.

[18] *FlexRay Communications System Protocol Specification version 3.0.1*, FlexRay Consortium, Stuttgart, Germany, 2010.

[19] *Specifications According ISO 21806-1 Road Vehicles -MOST—Part 1: General Information and Definitions*, MOST Cooperation, New York, NY, USA, 2011.

[20] *CAN With Flexible Data-Rate Specification version 1.0*, BOSCH Gmbh, Gerlingen, Germany, 2012.

[21] *CAN XL Documents Under Development*, CAN Newsletter Online, Singapore, 2022.

[22] W. Zeng, M. A. S. Khalid, and S. Chowdhury, "In-vehicle networks outlook: Achievements and challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1552–1571, 3rd Quart., 2016.

[23] *IEEE Standard for a Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks*, Standard 1722-2016 (Revision of IEEE Std 1722-2011), 2016, pp. 1–233.

[24] H.-Y. Li, L.-B. Chen, W.-J. Chang, J.-J. Tang, and K. S.-M. Li, "Design and development of an extensible multi-protocol automotive gateway," in *Proc. IEEE Int. Conf. Consum. Electron.-Taiwan (ICCE-TW)*, 2016, pp. 1–2, doi: 10.1109/ICCE-TW.2016.7521011.

[25] T. H. Pham, S. Shreejith, S. Steinhorst, S. A. Fahmy, and S. Chakraborty, "Heterogeneous communication virtualization for distributed embedded applications," in *Proc. 24th Euromicro Conf. Digit. Syst. Design (DSD)*, Sep. 2021, pp. 251–258.

[26] C. Varun and M. Kathiresh, "Automotive Ethernet in on-board diagnosis (Over IP) & in-vehicle networking," in *Proc. Int. Conf. Embedded Syst. (ICES)*, Jul. 2014, pp. 255–260.

[27] *Road Vehicles Functional Safety*, document ISO 26262-1:2018, 2018.

[28] *Functional Safety of Electrical/Electronic/Programmable Electronic (E/E/PE) Systems*, 2010.

[29] *Road Vehicles Safety of the Intended Functionality*, document ISO/PAS 21448:2019, 2019.

[30] *Standard for Safety for the Evaluation of Autonomous Products*, document ANSI/UL 4600, 2019.

[31] The Autonomous, TTTech Auto, Vienna, Austria, 2021.

[32] *The Autonomous Vehicle governance Ecosystem: A Guide for Decision-Makers*, World Economic Forum and The Autonomous, Cologny, Switzerland, 2021.

[33] *MotionWise Completes Functional Safety Assessment*, TTTech Auto, Wien, Austria, 2021.

[34] *S32G Safe and Secure Vehicle Network Processors*, NXP, Eindhoven, The, Netherlands, 2021.

[35] *Renesas Functional Safety Support for Automotive (2)—A Customizable Option for Precise Safety Analysis*, Renesas, Tokyo, Japan, 2021.

[36] *4 Layers of Automotive Security*, Van Roermund (NXP) Andy Birni-eTimo, Eindhoven, The, Netherlands, Aug. 2016.

[37] *EVITA: E-Safety Vehicle Intrusion Protected Applications*, Fraunhofer Inst. Secure Inf. Technol., Darmstadt, Germany, 2011.

[38] *IEEE Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) Security*, Standard 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006), 2018, pp. 1–239.

[39] *IPTABLES: Linux Userspace Packet Filtering Ruleset*, Linux Found., San Francisco, CA, USA, 2021.

[40] *Open Source Intrusion Prevention System*, SNORT IPS, CISCO Syst., San Jose, CA, USA, 2021.

[41] *Open Source Threat Detection Engine*, Suricata, Vancouver, BC, Canada, 2021.

[42] C.-W. Lin and H. Yu, ''Invited—Cooperation or competition: Coexistence of safety and security in next-generation Ethernet-based automotive networks,'' in *Proc. 53rd Annu. Design Autom. Conf.*, Jun. 2016, pp. 1–6.

[43] H. Y. Yeo, ''Automotive SerDes—Enabling better ADAS camera sensors,'' Keysight Technol., Santa Rosa, CA, USA, Tech. Rep., 2021.

[44] E. Canoglu, ''Lidar: A coherent vision for autonomous vehicle sensors (Part 2),'' NeoPhoton., San Jose, CA, USA, Tech. Rep., 2018.

[45] *Draft Standard for Local and Metropolitan Area Networks—Time-Sensitive Networking Profile for Automotive in-Vehicle Ethernet Communications*, Standard P802.1DG/D1.4, 2021.

[46] H.-T. Lim, D. Herrscher, L. Volker, and M. J. Waltl, ''IEEE 802.1AS time synchronization in a switched Ethernet based in-car network,'' in *Proc. IEEE Veh. Netw. Conf. (VNC)*, Nov. 2011, pp. 147–154.

[47] *IEEE Standard for Local and Metropolitan Area Networks—Timing and Synchronization for Time-Sensitive Applications IEEE Standard*, Standard 802.1AS-2020, 2020.

[48] *IEEE Standard for Local and metropolitan area Networks—Bridges and Bridged Networks—Amendment 28: Per-Stream Filtering and Policing*, Standard 802.1Qci-2017 (Amendment to IEEE Std 802.1Q-2014 as Amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, and IEEE Std 802.1Qbz-2016), 2017, pp. 1–65.

[49] *IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams*, Standard 802.1Qav-2009, 2010.

[50] *IEEE Standard for Local and Metropolitan Area Networks–Frame Replication and Elimination for Reliability*, Standard 802.1 CB-2017, L A N Man, Standards Committee, and IEEE Computer, 2017.

[51] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks Amendment 25: Enhancements for Scheduled Traffic*, Standard 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as Amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015), 2016, pp. 1–57.

[52] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 26: Frame Preemption*, Standard 802.1Qbu-2016 (Amendment to IEEE Std 802.1Q-2014), 2016, pp. 1–52.

[53] *IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks—Amendment 29: Cyclic Queuing and Forwarding*, Standard IEEE 802.1Qch-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd(TM)-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, IEEE Std 802.1Qbz-2016, and IEEE Std 802.1Qci-2017), 2017, pp. 1–30.

[54] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks Amendment 34: Asynchronous Traffic Shaping*, Standard 802.1Qcr-2020 (Amendment to IEEE Std 802.1Q-2018 as Amended by IEEE Std 802.1Qcp-2018, IEEE Std 802.1Qcc-2018, IEEE Std 802.1Qcy-2019, and IEEE Std 802.1Qcx-2020), 2020, pp. 1–151.

[55] E. Mohammadpour, E. Stai, M. Mohiuddin, and J.-Y. Le Boudec, ''Latency and backlog bounds in time-sensitive networking with credit based shapers and asynchronous traffic shaping,'' in *Proc. 30th Int. Teletraffic Congr. (ITC)*, Sep. 2018, pp. 1–6.

[56] Z. Zhou, Y. Yan, M. Berger, and S. Ruepp, ''Analysis and modeling of asynchronous traffic shaping in time sensitive networks,'' in *Proc. 14th IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, Jun. 2018, pp. 1–4.

[57] L. Zhao, P. Pop, and S. Steinhorst, ''Quantitative performance comparison of various traffic shapers in time-sensitive networking,'' *IEEE Trans. Netw. Service Manag.*, doi: 10.1109/TNSM.2022.3180160.

[58] L. Lo Bello, G. Patti, and G. Vasta, ''Assessments of real-time communications over TSN automotive networks,'' *Electronics*, vol. 10, no. 5, p. 556, Feb. 2021.

[59] *1G MTSN Multiport TSN Switch IP Core*, SoC-e, Erandio, Spain, 2018.

[60] *Time Sensitive Networking IP Cores Fraunhofer IPMS*, Fraunhofer IPMS, Dresden, Germany, 2021.

[61] *Time Sensitive Networking Portfolio*, TTTech, Vienna, Austria, 2021.

[62] S. B. H. Said, Q. H. Truong, and M. Boc, ''SDN-based configuration solution for IEEE 802.1 time sensitive networking (TSN),'' *ACM SIGBED Rev.*, vol. 16, no. 1, pp. 27–32, Feb. 2019.

[63] Y. Seol, D. Hyeon, J. Min, M. Kim, and J. Paek, ''Timely survey of time-sensitive networking: Past and future directions,'' *IEEE Access*, vol. 9, pp. 142506–142527, 2021.

[64] D. Pannel, ''Automotive Ethernet AVB functional and interoperability specification,'' AVNU Alliance, OR, USA, Tech. Rep., 2019.

[65] Z. Abdellaoui, I. Ben Mbarek, R. Bouhouch, and S. Hasnaoui, ''DDS middleware on FlexRay network: Simulink blockset implementation of wheel's sub-blocks and its adaptation to DDS concept,'' in *Proc. IEEE 9th Int. Symp. Intell. Signal Process. (WISP)*, May 2015, pp. 1–6.

[66] Z. Abdellaoui, I. B. Mbarek, R. Bouhouch, and S. Hasnaoui, ''DDS middleware on FlexRay network: Simulink blockset implementation of wheel's sub-blocks and its adaptation to DDS concept,'' in *Proc. IEEE 9th Int. Symp. Intell. Signal Process. (WISP)*, May 2015, pp. 1–6.

[67] T. Agarwal, P. Niknejad, M. R. Barzegaran, and L. Vanfretti, ''Multi-level time-sensitive networking (TSN) using the data distribution services (DDS) for synchronized three-phase measurement data transfer,'' *IEEE Access*, vol. 7, pp. 131407–131417, 2019.

[68] *Using DDS Over TSN to Support NATO Generic Vehicle Architecture (NGVA) for Land Systems*, RELYUM and Real Time Innovations (RTI), Singapore, 2019.

[69] *Driving Interoperability and Performance in Automotive Systems With DDS and TSN*, NXP, Eindhoven, The, Netherlands, 2021.

[70] *SOME/IP Protocol Specification*, AUTOSAR, Bayern, Germany, 2016.

[71] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, ''OpenFlow: Enabling innovation in campus networks,'' *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Apr. 2008.

[72] M. Doering and J. Bierschenk, *Software-Defined Networking in Automotive*. Bosch, Gerlingen, Germany, 2018.

[73] 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, Auto., Connected Veh. Syst. Services*, Oct. 2017, pp. 73–74.

[74] K. Halba and C. Mahmoudi, ''In-vehicle software defined networking: An enabler for data interoperability,'' in *Proc. 2nd Int. Conf. Inf. Syst. Data Mining*, Apr. 2018, pp. 93–97.

[75] M. Doering and M. Wagner, ''Retrofitting SDN to classical in-vehicle networks: SDN4CAN,'' Robert Bosch GmbH, Corporate Res. Commun. Netw. Technol., Renningen, Germany, Tech. Rep., Oct. 2017.

[76] M. Häberle, F. Heimgaertner, H. Loehr, N. Nayak, D. Grewe, S. Schildt, and M. Menth, ''An SDN architecture for automotive Ethernets,'' Robert Bosch GmbH, Corporate Sector Res. Advance Eng., Renningen, Germany, Tech. Rep., Apr. 2020.

[77] 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. 2019, pp. 1–5.

[78] H. Chahed and A. J. Kassler, ''Software-defined time sensitive networks configuration and management,'' in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2021, pp. 124–128.

[79] A. Shukla and K.-T. Foerster, ''Shortcutting fast failover routes in the data plane,'' in *Proc. Symp. Archit. Netw. Commun. Syst.*, Dec. 2021, pp. 15–22.

[80] 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.

[81] A. A. Syed, S. Ayaz, T. Leinmuller, 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.

[82] C. Perner, C. Schmitt, and G. Carle, ''Dynamic network reconfiguration in safety-critical aeronautical systems,'' in *Proc. AIAA/IEEE 39th Digit. Avionics Syst. Conf. (DASC)*, Oct. 2020, pp. 1–8.

[83] 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.

[84] M. Wagner, D. Zobel, and A. Meroth, ''SODA: Service-oriented architecture for runtime adaptive driver assistance systems,'' in *Proc. IEEE 17th Int. Symp. Object/Component/Service-Oriented Real-Time Distrib. Comput.*, Jun. 2014, pp. 150–157.

[85] N. Gebara, A. Lerner, M. Yang, M. Yu, P. Costa, and M. Ghobadi, ''Challenging the stateless quo of programmable switches,'' in *Proc. 19th ACM Workshop Hot Topics Netw.*, Nov. 2020, pp. 153–159.

[86] A. Srivastava and D. Adhikari, "CAN-LIN bridge for driver assistance and passenger comfort an optimized resource approach," in *Proc. 2nd IEEE Int. Conf. Recent Trends Electron., Inf. Commun. Technol. (RTEICT)*, May 2017, pp. 506–510.

[87] J. Gosda, "Autosar communication stack," AUTOSAR, Munich, Germany, Tech. Rep., 2009.

[88] A. Hbaieb, O. B. Rhaiem, and L. Chaari, "In-car gateway architecture for intra and inter-vehicular networks," in *Proc. 14th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2018, pp. 1489–1494.

[89] *ElectroKnox Makes Software-Defined Vehicles a Reality With the Xilinx Zynq Platform*, ElectroKnox, Santa Clara, CA, USA, 2021.

[90] *Ethernovia Virtualizing Vehicle Communication*, ElectroKnox, Santa Clara, CA, USA, 2021.

[91] *Open vSwitch*, Linux Found., San Francisco, CA, USA, 2022.

[92] *Why Open vSwitch*, Linux Found., San Francisco, CA, USA, 2022.

[93] *Myth-Busting DPDK in 2020*, Avid Think, Burbank, CA, USA, 2020.

[94] F. Rezabek, M. Bosk, T. Paul, K. Holzinger, S. Gallenmuller, A. Gonzalez, A. Kane, F. Fons, Z. Haigang, G. Carle, and J. Ott, "EnGINE: Developing a flexible research infrastructure for reliable and scalable intra-vehicular TSN networks," in *Proc. 17th Int. Conf. Netw. Service Manage. (CNSM)*, Oct. 2021, pp. 530–536.

[95] M. Bosk, F. Rezabek, K. Holzinger, A. Gonzalez, A. Kane, F. Fons, Z. Haigang, G. Carle, and J. Ott, "Demo: Environment for generic in-vehicular network experiments–EnGINE," in *Proc. IEEE Veh. Netw. Conf. (VNC)*, Nov. 2021, pp. 117–118.

[96] C. Jo, J. Park, and J. Jeon, "Multi-core gateway architecture and scheduling algorithm for high-performance gateway implementation," in *Proc. IEEE Int. Conf. Consum. Electron. Asia (ICCE-Asia)*, Nov. 2020, pp. 1–6.

[97] A. Olofsson, T. Nordstrom, and Z. Ul-Abdin, "Kickstarting high-performance energy-efficient manycore architectures with epiphany," in *Proc. 48th Asilomar Conf. Signals, Syst. Comput.*, Nov. 2014, pp. 1719–1726.

[98] G. Xie, G. Zeng, R. Kurachi, H. Takada, Z. Li, R. Li, and K. Li, "WCRT analysis and evaluation for sporadic message-processing tasks in multicore automotive gateways," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 2, pp. 281–294, Feb. 2019.

[99] A. S. A. Dhanotia, "Accelerometer: Understanding acceleration opportunities for data center overheads at hyperscale," Facebook, Sacramento, CA, USA, Tech. Rep., 2020.

[100] D. Trilla, J.-D. Wellman, A. Buyuktosunoglu, and P. Bose, *NOVIA: A Framework for Discovering Non-Conventional Inline Accelerators*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 507–521.

[101] L. Apvrille, R. El Khayari, H. Rachid, O. Henniger, Y. Roudier, H. Schweppe, H. Seudié, B. Weyl, and M. Wolf, "Secure automotive on-board electronics network architecture," EVITA Project, Germany, Tech. Rep., 2010.

[102] J. Rettkowski, J. Haase, S. Primus, M. Hübner, and D. Göhringer," *Performance Analysis of Application-Specific Instruction-Set Routers in Networks-on-Chip*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 16–21.

[103] D. Firestone *et al.*, "Azure accelerated networking: SmartNICs in the public cloud," USENIX, Berkeley, CA, USA, Tech. Rep., Apr. 2018.

[104] A. Putnam *et al.*, "A reconfigurable fabric for accelerating large-scale datacenter services," in *Proc. ACM/IEEE 41st Int. Symp. Comput. Archit. (ISCA)*, Jun. 2014, pp. 13–24.

[105] L. Wirbel, "Xilinx SDNet: A new way to specify network hardware," Linley Group, Mountain View, CA, USA, Tech. Rep., 2014.

[106] V. Krishnan, O. Serres, and M. Blocksome, "Configurable network protocol accelerator (COPA)," *IEEE Micro*, vol. 41, no. 1, pp. 8–14, Jan./Feb. 2021, doi: 10.1109/MM.2020.3042167.

[107] S. Shreejith, P. Mundhenk, A. Ettner, A. S. Fahmy, S. Steinhorst, M. Lukasiewycz, and S. Chakraborty, "VEGa: A high performance vehicular Ethernet gateway on hybrid FPGA," *IEEE Trans. Comput.*, vol. 66, no. 10, pp. 1790–1803, Oct. 2017.

[108] F. Gross, T. Steinbach, F. Korf, T. C. Schmidt, and B. Schwarz, "A hardware/software co-design approach for Ethernet controllers to support time-triggered traffic in the upcoming IEEE TSN standards," in *Proc. IEEE 4th Int. Conf. Consum. Electron. Berlin (ICCE-Berlin)*, Sep. 2014, pp. 9–13.

[109] 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.

[110] A. Puhm, P. Roessler, M. Wimmer, R. Swierczek, and P. Balog, "Development of a flexible gateway platform for automotive networks," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Autom.*, Sep. 2008, pp. 456–459.

[111] S. Shreejith and S. A. Fahmy, "Smart network interfaces for advanced automotive applications," *IEEE Micro*, vol. 38, no. 2, pp. 72–80, Mar. 2018.

[112] S. Shreejith, S. A. Fahmy, and M. Lukasiewycz, "Reconfigurable computing in next-generation automotive networks," *IEEE Embedded Syst. Lett.*, vol. 5, no. 1, pp. 12–15, Mar. 2013.

[113] S. Shreejith and S. A. Fahmy, "Security aware network controllers for next generation automotive embedded systems," in *Proc. 52nd Annu. Design Autom. Conf.*, Jun. 2015, pp. 1–6.

[114] *S32G Processors for Vehicle Networking*, NXP, Eindhoven, The, Netherlands, 2021.

[115] *32-Bit TriCoreT AURIXT-TC4x*, Infineon Technologies, Neubiberg, Germany, 2022.

[116] *BCM88480 800-Gb/s Integrated Packet Processor and Traffic Manager Single-Chip Device*, Broadcom, San Jose, CA, USA, 2022.

[117] *Marvell 88Q5050 Secure Automotive Switch*, Marvell, Wilmington, DE, USA, 2020.

[118] *BlueBox 3.0 Third-Generation, Automotive High Performance Compute (AHPC) Development Platform*, NXP, Eindhoven, The, Netherlands, 2020.

[119] *Arion IP Evaluation Platform*, TTTech, Vienna, Austria, 2021.

[120] *S32G2 PFE Product Brief*, NXP, Eindhoven, The, Netherlands, 2021.

[121] *S32G2 LLCE Standard Firmware Product Brief*, NXP, Eindhoven, The, Netherlands, 2022.

[122] *TTTech Edge IP Solution*, TTTech Ind., Vienna, Austria.

[123] *TSN-SE: Ethernet TSN Switched Endpoint Core*, Fraunhofer IPMS, Munich, Germany, 2021.

[124] *TSN-EP: Ethernet TSN Endpoint Core*, Fraunhofer IPMS, Fraunhofer IPMS, 2021.

[125] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "p4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014.

[126] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 87–95, Jul. 2014.

[127] F. Hauser, M. Häberle, D. Merling, S. Lindner, V. Gurevich, F. Zeiger, R. Frank, and M. Menth, "A survey on data plane programming with p4: Fundamentals, advances, and applied research," Dept. Comput. Sci., Chair Commun. Netw., Univ. Tübingen, Tübingen, Germany, Intel, Barefoot Division (BXD), USA, Siemens AG, Corporate Technol., Munich, Germany, Tech. Rep., 2021.

[128] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, "An exhaustive survey on p4 programmable data plane switches: Taxonomy, applications, challenges, and future trends," *IEEE Access*, vol. 9, pp. 87094–87155, 2021.

[129] O. Michel, R. Bifulco, G. Rétvári, and S. Schmid, "The programmable data plane: Abstractions, architectures, algorithms, and applications," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 1–36, May 2021.

[130] H. Harkous, N. Kroger, M. Jarschel, R. Pries, and W. Keller, "Modeling and performance analysis of p4 programmable devices," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2021, pp. 67–73.

[131] J. S. D. Silva, T. Stimpfling, T. Luinaud, B. Fradj, and B. Boughzala, "One for all, all for one: A heterogeneous data plane for flexible p4 processing," in *Proc. IEEE 26th Int. Conf. Netw. Protocols (ICNP)*, Sep. 2018, pp. 440–441.

[132] D. D. Robin and I. Dr Javed Khan, "Toward an abstract model of programmable data plane devices," Kent State Univ., Kent, OH, USA, Tech. Rep., 2020. [Online]. Available: https://arxiv.org/abs/2008.08697v1

[133] I. Butun, Y. K. Tuncel, and K. Oztoprak, "Application layer packet processing using Pisa switches," *Sensors*, vol. 21, no. 23, p. 8010, Nov. 2021.

[134] *P416 Portable Switch Architecture (PSA) Version 1.1*, documentP4.org, 2018.

[135] Z. Cao, H. Su, Q. Yang, J. Shen, M. Wen, and C. Zhang, "P4 to FPGA—A fast approach for generating efficient network processors," *IEEE Access*, vol. 8, pp. 23440–23456, 2020.

[136] *Tofino 2 P4 Programmability With More Bandwidth*, INTEL, Santa Clara, CA, USA, 2018.

[137] *An Overview of Pensando Systems Capri ASIC*, Pensando Syst., San Jose, CA, USA, 2018.

[138] *NFP4000 Theory of Operation*, Netronome, Santa Clara, CA, USA, 2016.

[139] P. Bosshart, G. Gibb, H.-S. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, and M. Horowitz, "Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 99–110, Oct. 2013.

[140] S. Chole, A. Fingerhut, S. Ma, A. Sivaraman, S. Vargaftik, A. Berger, G. Mendelson, M. Alizadeh, S.-T. Chuang, I. Keslassy, A. Orda, and T. Edsall, "DRMT: Disaggregated programmable switching," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2017, pp. 1–14.

[141] S. Pontarelli, R. Bifulco, M. Bonola, C. Cascone, M. Spaziani, V. Bruschi, D. Sanvito, G. Siracusano, A. Capone, M. Honda, F. Huici, and G. Siracusano, "FlowBlaze: Stateful packet processing in hardware," in *Proc. 16th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*. Boston, MA, USA: USENIX Association, MA, Feb. 2019, pp. 531–548.

[142] D. Merling, S. Lindner, and M. Menth, "Hardware-based evaluation of scalable and resilient multicast with BIER in p4," *IEEE Access*, vol. 9, pp. 34500–34514, 2021.

[143] S. Y. Rim, Z. Cui, and L. Qian, "High performance packet processor architecture for network virtualization: Programmable packet processor architecture as a data flow machine," in *Proc. Int. Conf. Algorithms, Comput. Artif. Intell.*, Dec. 2018, pp. 1–5.

[144] I. Kunze, M. Gunz, D. Saam, K. Wehrle, and J. Rüth, "Tofino + p4: A strong compound for AQM on high-speed networks," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2021, pp. 72–80.

[145] S. Ibanez, G. Antichi, G. Brebner, and N. McKeown, "Event-driven packet processing," in *Proc. HotNets*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 133–140.

[146] B. P. Viegas, G. A. de Castro, F. A. Lorenzon, D. F. Rossi, and C. M. Luizelli, "The actual cost of programmable SmartNICs: Diving into the existing limits," in *Advanced Information Networking and Applications*, L. Barolli, I. Woungang, and T. Enokido, Eds. Cham, Switzerland: Springer, 2021, pp. 181–194.

[147] P. Li and Y. Luo, "P4GPU: Acceleration of programmable data plane using a CPU-GPU heterogeneous architecture," in *Proc. IEEE 17th Int. Conf. High Perform. Switching Routing (HPSR)*, Jun. 2016, pp. 168–175.

[148] K. Karras, T. Wild, and A. Herkersdorf, "A folded pipeline network processor architecture for 100 Gbit/s networks," in *Proc. 6th ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, Oct. 2010, pp. 1–11.

[149] J. Santiago Da Silva, T. Stimpfling, T. Luinaud, B. Fradj, and B. Boughzala, "One for all, all for one: A heterogeneous data plane for flexible p4 processing," in *Proc. IEEE 26th Int. Conf. Netw. Protocols (ICNP)*, Sep. 2018, pp. 440–441.

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 Politécnica 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 with 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. Currently, she is with Huawei Technologies with the Applied Network Technology Laboratory, Munich Research Center, Munich, Germany, focusing on HW accelerators design for automotive networking solutions. Her current research interests include HW design for automotive in-vehicle networks and systems 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 the Universitat Rovira i Virgili (URV), Tarragona, Spain, in 1995, 2001, and 2012, respectively.

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 in Europe and Asia. Currently, he is with Huawei Technologies, where he has the role of Chief Automotive In-Vehicle Network Researcher with the Applied Network Technology Laboratory of the Huawei Munich Research Center.

JUAN MANUEL MORENO AROSTEGUI received the Ph.D. degree in telecommunications engineering from the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 1994.

He is currently an Associate Professor with the Department of Electronic Engineering, UPC. His research interests include analog and digital VLSI design, neural networks models, architectures for programmable devices and systems on chip, bio-inspired computing techniques, and electronic systems for e-health infrastructures.