Softwarization of Automotive E/E Architectures: A Software-Defined Networking Approach

Marco Haeberle*, Florian Heimgaertner*, Hans Loehr‡, Naresh Nayak†,
Dennis Grewe†, Sebastian Schildt†, and Michael Menth*
* University of Tuebingen, Chair of Communication Networks, Tuebingen, Germany
Email: {marco.haeberle,florian.heimgaertner,menth}@uni-tuebingen.de
† Robert Bosch GmbH, Corporate Sector Research and Advance Engineering, Renningen, Germany
Email: {naresh.nayak,dennis.grewe,sebastian.schildt}@de.bosch.com
‡ SUSE Software Solutions Germany GmbH, Nuremberg, Germany
The author was with † when the study was performed.
Email: hans.loehr@suse.com

***Abstract*—Traditional in-vehicle networks are based on low-bandwidth technologies like CAN. They are statically deployed and configured in the manufacturing process depending on the vehicle configuration. With the introduction of additional camera and entertainment applications and increased bandwidth demand, Ethernet technology becomes more relevant in the automotive sector. Use cases like trailer networks and integration of new applications requiring in-vehicle sensor data require re-configurable network architectures. In this work, we propose a flexible architecture for automotive Ethernet networks accommodating both high-bandwidth multimedia streams and time-critical low bandwidth data. Based on the software-defined networking paradigm, the in-vehicle network can be re-configured for the integration of additional hardware and applications. We illustrate its concepts for operations, management, safety, and security.**

I. Introduction

Connected and smart vehicles will have a disruptive impact on tomorrow's mobility solutions. The way vehicles will be used in the future will change fundamentally. One driving force of such change is highly automated driving (HAD). It will have an impact on quality of life by relieving drivers from the burden to deal with stressful or hazardous situations, enabling new models for using vehicles, e.g., as "mobile offices" or "living room on wheels" [1], [2].

In the past years, the rapid development of communication technologies formed the basis for the interconnection of multiple devices. This includes the evolution of the in-vehicle electrical/electronic (E/E) architecture – from a single CAN bus to a heterogeneous system with several bus technologies and centralized gateways – as well as the interconnection of vehicles and its surroundings towards a seamless mobile extension of the digital world of data and services.

To leverage the business potential, enabling technologies are required to introduce innovative use cases that profit from combining in-vehicle data with resources outside of the vehicle. Contemporary in-vehicle architectures have to deal with their legacy in-vehicle network architecture which consists of several bus systems (e.g., CAN [3], FlexRay [4], LIN [5], MOST [6], etc.) plus several dozens of network components and applications.

Currently, these networks are configured statically, resulting in tremendous planning overhead during the manufacturing process. The static communication patterns of these networks make it hard to introduce any new application/function during the lifetime of the vehicle as the network has to be re-configured to adapt to changes. This will cause issues with the upcoming HAD and the vision of a "living room on wheels" which are expected to increase the number of in-vehicle and ex-vehicle information exchange, deployed functions and applications, and to enable a faster innovation cycle on the software side.

Software-defined networking (SDN) [7] is a promising network paradigm to solve the problem set of statically configured networks by providing concepts to re-configure the behavior of network components during runtime. While SDN does not provide mechanisms to meet the real-time requirements of vehicular communication systems, Time-Sensitive Networking (TSN) [8] provides a collection of tools supporting real-time communication.

This work proposes an SDN architecture for heterogeneous automotive E/E architectures, supporting real-time capabilities using principles of TSN that can be re-configured on demand.

The remainder of the paper is structured as follows. Section II reviews the principles of SDN and TSN and provides an overview of SDN in the automotive context. Based on the introduction of automotive use cases, Section III introduces requirements for future E/E architectures. Section IV presents the novel SDN automotive architecture and discusses design rationales such as network management and security. Finally, Section V concludes the paper.

II. Related work

In this section we first review fundamentals of SDN and TSN. Then, we discuss related work in the area of automotive network architectures.

A. Software-Defined Networking

SDN separates the data plane and the control plane by shifting intelligence from distributed forwarding nodes, i.e.,

©2020 IEEE Vehicular Networking Conference (VNC), December 2020, Virtual Conference

routers or switches, to a logically centralized controller [7]. While non-SDN switches establish a forwarding table by learning addresses, e.g., locally or through distributed routing protocols, a controller installs a set of forwarding rules on SDN switches either on initialization or during run-time.

Figure 1 shows the layers of the SDN architecture as defined by the Open Networking Foundation (ONF). The infrastructure layer corresponds to the data plane, consists of the network nodes, and is responsible for data forwarding. The control layer corresponds to the control plane and is responsible for the configuration of the packet matching and forwarding rules of the data plane. The application layer consists of network applications implementing specific functionality giving input to the control layer. The southbound interface is the communication channel between the SDN control layer and the SDN infrastructure layer. The predominant standard southbound interface is OpenFlow [9] specified by the ONF. Southbound messages can be transmitted over the data plane infrastructure (in-band signalling) or over a dedicated control network (out-of-band signalling). The interface between the application layer and the control layer is called northbound interface. While there is no accepted standard for the northbound interface to date, a REST API is a common implementation.

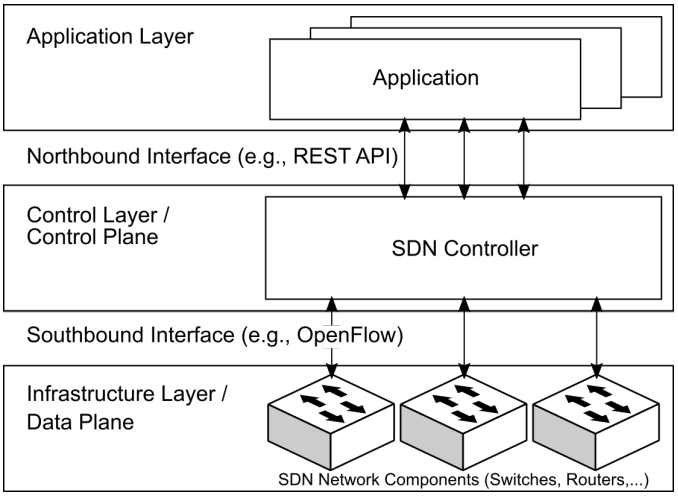


Fig. 1. 3-layer architecture of SDN.

SDN approaches like OpenFlow specify a data plane steered by a central SDN controller. While a highly flexible control plane behavior can be implemented in controller applications, the data plane is limited to a fixed set of functions which are determined by the OpenFlow protocol.

The concept of programmable data planes goes a step further and also enables flexible implementation of data plane behavior for programmable network devices in a high-level programming language such as P4 [10].

B. Time Sensitive Networking

The IEEE Time-Sensitive Networking Task Group (TSN-TG), originally started as the Audio-Video Bridging (AVB) Working Group, strives for handling synchronized low latency communication over Ethernet. As a part of this initiative, the TSN-TG has added several extensions to the IEEE 802.1Q and also published new standards like the IEEE 802.1CB. The extensions and new standards cover different aspects of real-time communication over Ethernet ranging from clock synchronization to traffic shaping and policing [8]. Overall, TSN can be seen as a collection of tools, each targeting a specific problem of real-time communication. While an elaborate description of all TSN mechanisms is out of scope in this summary, we briefly describe a few of those in the following.

The end points of a TSN stream are called talker and listener. TSN supports point-to-multipoint streams with multiple listeners. Credit-Based Shaper (CBS) from IEEE 802.1Qav is used for reserving (and limiting) bandwidth for multimedia streams in the network while the Time-Aware Shaper (TAS) from IEEE 802.1Qbv [11] handles scheduled periodic traffic typically stemming from industrial control systems. The IEEE 802.1Qbu [12] provides mechanisms for a higher-priority Ethernet frame to preempt the transmission of a lower priority frame. The mechanisms to replicate critical frames in the network and transport them over disjoint paths before eliminating duplicates (*aka* 1+1 protection) are included in IEEE 802.1CB [13]. TSN also includes traffic policing mechanisms to secure real-time traffic from best-effort traffic along with many other variants of traffic shapers.

The multitude of features available with TSN increases the complexity of network configuration. The IEEE 802.1Qcc proposes several configuration models for this purpose. The centralized configuration model resembles the SDN approach and requires a Central Network Controller (CNC) for configuration of the network nodes and a Centralized User Configuration (CUC) for setting up talkers and listeners. CNC and CUC communicate via the User Network Interface (UNI), similar to the southbound interface in SDN. This configuration model is the most likely bet to be able to use the full set of TSN features. SDN concepts for configuration and management of real-time networks based on TSN are discussed in the literature [14], [15].

C. Automotive Network Architectures

The most common in-vehicle communication technology is the Controller Area Network (CAN) [3]. CAN uses a bus topology and provides a low-bandwidth CSMA network that is used to connect all kinds of Electronic Control Units (ECUs). With the growing number of ECUs, manufacturers started to install multiple CAN buses. Using gateways, those buses are interconnected with each other and with other specialized communication systems, like LIN, FlexRay, and MOST.

The Local Interconnect Network (LIN) [5] uses a bus topology and a master/slave media access control scheme. LIN is mainly used for comfort features like window lifts or air conditioning. FlexRay [4] is a communication system for hard real-time requirements. It supports both bus and star topologies and uses a TDMA access scheme. The Media Oriented Systems Transport (MOST) [6] is a communication

system for multimedia applications. It uses a ring topology and a CSMA access control scheme.

While the traditional architecture model is based on distributed ECUs connected to a central gateway (see Figure 2(a)) there are efforts to consolidate functionality into more powerful devices known as vehicle computers. While conventional ECUs are mostly based on microcontrollers, vehicle computers are based on microprocessors and feature virtualization technologies. Shifting functionalities from many ECUs to few vehicle computers reduces the complexity of the architecture and allows for new features like over-the-air updates [17]. The domain architecture shown in Figure 2(b) separates ECUs into multiple domains (e.g., powertrain, safety, comfort, etc.) and each domain is managed by a domain controller (DC) ECU. The DCs are interconnected by a backbone network. In a zone architecture as shown in Figure 2(c) the ECUs are separated into topological zones instead of functional domains. Each ECU is connected to the nearest local zone I/O controller. The zone I/O controllers are interconnected via a backbone or mesh network. Zone architectures reduce cabling effort compared to domain architectures and enable stronger centralization of computing resources.

With increasing bandwidth demand resulting from the integration of camera and high-definition multimedia applications, Ethernet becomes a relevant network technology for in-vehicle networks [18]. Automotive Ethernet is based on 100Base-T1 over UTSP cables. It can be used as backbone network for domain and zone architectures.

Future enhancements of automotive Ethernet systems are discussed in the literature. Lo Bello et al. [19] provide an overview of recent development in automotive network systems. Alderisi et al. [20] perform simulation studies with AVB for advanced driver assistance systems. Brunner et al. [21] propose the use of TSN in a zone-based automotive Ethernet architecture. Migge et al. [22] present a case study about performance of AVB and TSN in automotive Ethernet networks. Additional use cases for the proposed TSN automotive profile IEEE P802.1DG are discussed by Pannell et al. [16].

Using SDN in automotive networks architectures is an active field of research. Fussey and Parisis [23] discuss the advantages that stem from in-vehicle network architectures using SDN. Among others, they name dynamic re-configuration that is one of the main features of the architecture presented in this work.

Häckel et al. [24] explore how SDN and TSN can be combined for use in automotive networks. They implement time-sensitive flows using OpenFlow.

Halba et al. [25] use an SDN approach to enable communication between ECUs using different in-vehicle networks. They equip ECUs with "IP to Legacy-In-Vehicle-Network adapters" and interconnect them with a Time Triggered Ethernet backbone and OpenFlow switches. In a related work [26], they also introduce a fast failover mechanism for mitigating link failures. However, they do not make use of most of the benefits that come with SDN.

III. Use Cases and Requirements

In this section, we present two use cases that require flexible network architectures and summarize their resulting requirements.

A. Trailer Networks

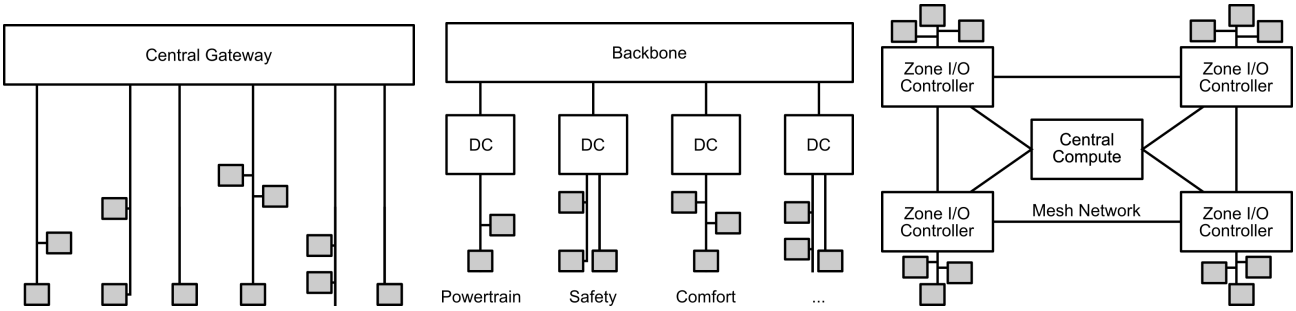
Trailers connected to cars or trucks constitute a use case that can particularly benefit from re-configurable in-vehicle networks. Nowadays, trailers are connected to the car electrically using one of several standardised connectors using 5 to 22 pins. The connectors support only a limited set of functions, e.g., tail lamps, stop lamps, turn signals, or in rare cases electric brakes. More sophisticated applications are not supported by traditional trailer connectors. An automotive SDN architecture facilitates the connection of arbitrary components in the trailer to the car's network. In addition, the SDN architecture enables automated adjustment of configuration parameters like the brake bias or settings of the blind spot sensor.

One demonstrative application in this use case is connecting park distance control (PDC) sensors or a rear view camera located in the trailer to the car's infotainment system. Another application could be allowing the trailer to use the Internet uplink of the car, e.g., for a Wi-Fi access point in a camping trailer. With TSN, it may even be possible to control brake systems in the trailer.

B. Downloadable Driver-Assistance Systems

An automotive SDN architecture can also change how connected components are developed and serviced during their life cycle. Traditionally, the feature set of a car and the functionality of its components does not change after the car is manufactured. Applying software patches normally requires the car to be brought to a repair shop. Adding new features with updates is usually not possible at all as the car's E/E architecture is static and cannot be changed. However, in recent years, the complexity of automotive software increased continuously and the state of the art of complex systems like driver assistance changes rapidly. This makes the possibility to patch existing systems and add new features to it via over-the-air (OTA) updates desirable. Decoupling the development of automotive software from the development of hardware has thus been called for by market analysts [27]. Software-wise, this becomes possible when replacing ECUs with vehicle computers that feature virtualization. Coupled with an automotive SDN architecture, this gives manufacturers more flexibility to develop updates and new features for cars as it enables them to change the car's data network.

One hypothetical example is an update of a collision avoidance system. Originally, the collision avoidance monitors traffic in front of the car using a camera. The manufacturer then wants to update the system and add a feature that monitors traffic behind the car while reversing by checking the PDC sensors. Using a traditional E/E architecture, this feature cannot be added as the collision avoidance system has no access to the PDC sensors. With an SDN architecture, the



(a) Topology with central gateway. (b) Domain model. (c) Zone model.

Fig. 2. Topology models for E/E architectures [16].

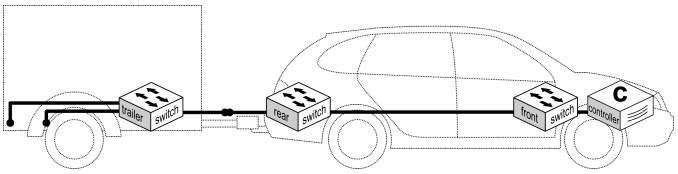


Fig. 3. Trailer network connected to vehicle network.

network can be re-configured to clone the packets coming from the PDC sensors and send the copies to the collision avoidance system.

C. Requirements

In both use cases, the most required feature is the ability to re-configure the network. Re-configuration of the network requires a central management entity that has a global view of the network in order to ensure that all necessary communication between the components and applications of the car can take place. This includes the need for a view of legacy field buses like CAN or LIN. To that end, a mechanism to discover the components installed in the car is necessary.

To benefit from the global view and optimally and efficiently manage networking resources, there is also a need for a northbound interface of the central controller that facilitates reactions to changed requirements of the components and applications. For safety-critical systems, the central controller needs to ensure that all requirements to the network, e.g., real-time constraints, are met. Both the northbound interface of the controller and the discovery mechanism must be well-defined and the definitions need to be accessible by all potential manufacturers of data plane devices for the purpose of interoperability between devices of different manufacturers.

As the network of the car is vital to the operation of the car, safety and security of the network architecture need to be guaranteed. This includes conformity to international standards like ISO 26262, but also the security of the architecture itself. In particular, attackers need to be prevented from gaining control over the network, e.g., by adding a malicious device to the network. In addition, the network architecture needs to be able to cope with failed components and be either fail-operational or fail-safe. A single-point-of-failure thus needs to be avoided.

IV. Architecture

In the following, we give an overview of the components of the proposed automotive SDN architecture and describe their interaction. We use the topology introduced in [16] as a basis. Figure 4 shows the set-up.

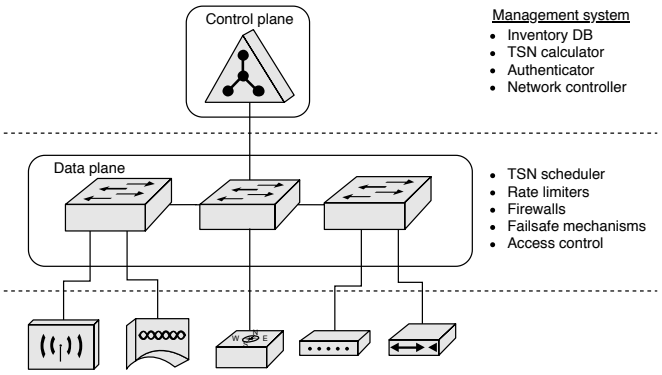


Fig. 4. Overview of the proposed automotive SDN architecture.

The in-car data plane consists of schedulers, rate limiters, firewalls, fail-safe mechanisms, access control mechanisms and redundant links. It connects the car's components to the applications and a management system. The management system is responsible for authenticating components and applications, takes care of TSN configuration, controls the in-car network, and keeps an inventory of components and applications and their permissions. The internal inventory is augmented by an external database containing information about components that may be added to the car.

A. Data Plane

The in-car data plane is shown in Figure 5 and consists of two switches connected with two redundant backbone links. One of the switches is located in the front of the car and the other in the back to keep the cables to the ECUs as short as possible. Each backbone link runs on one side of the car to minimize the risk of both links failing in case of a crash.

Traffic on the data plane is divided into hard real-time traffic, soft real-time traffic, best-effort traffic and network configuration traffic classes. Hard real-time traffic originates from safety-critical components, e.g., the brake system, and must always be transmitted within fixed deadlines. Soft real-time traffic is associated with systems that are less critical and can operate in a degraded state if the deadline is not met, e.g., light and rain sensors. Best-effort traffic is associated with systems that are not safety-critical, e.g., infotainment systems. Soft real-time and best effort traffic classes may prioritize certain traffic within the class. Hard real-time traffic does not need prioritization within its class as all hard real-time traffic needs to be transmitted in time. A rate limiter prevents faulty components from flooding the network. Non-related traffic flows are isolated from each other. Further protection from intruders is achieved by authenticating traffic using systems like MACsec (IEEE 802.1AE) or AUTOSAR SecOC [28]. The in-car network provides an Internet uplink that is secured using a firewall.

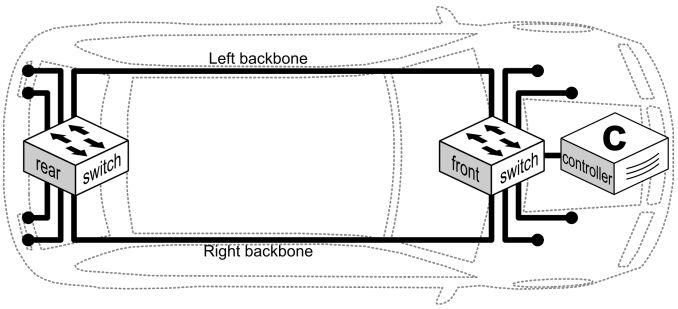


Fig. 5. Network with in-band controller.

During normal operation, the two links between the back and the front of the car are bonded using link aggregation. Different scheduling variants can be used depending on the requirements regarding throughput and safety. Possible scheduling variants may be round-robin scheduling for creating one logical link, load-balancing traffic over both links on a per-flow or per-component basis, or using 1+1 protection based on IEEE 802.1CB for traffic with redundancy requirements. In case of a link failure, all traffic is redistributed to the functioning link. If maximum throughput on the single link is not sufficient, best-effort traffic can be dropped to guarantee a safe operation of the car. Furthermore, it is possible to reduce sensor rates to the minimal safe rate to reduce total traffic.

B. Management

The data plane is configured by the network controller via in-band signalling. In-band signalling is preferred over out-of-band-signalling in this application as the reduced amount of cabling reduces cost and especially weight. Furthermore, in-band signalling enables an easier expansion of the network, e.g., as described in Section III-A, as no separate cables for management are needed. The network controller is connected directly to one of the switches (see Figure 5). It features a northbound interface that can be used by components and applications to trigger certain re-configurations of the network. Access to the northbound interface is regulated by access control lists (ACLs) and different permission levels.

The configuration of the network utilizes information from an inventory of components (e.g., sensors, actuators, and infotainment systems) and applications.

C. Operations

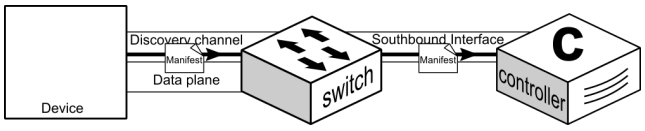
In the following we explain how TSN schedules are configured, we describe discovery mechanisms for new devices or applications, and we show how the architecture can handle failures of network components.

1) TSN Configuration: When a new safety-critical device requiring TSN communication is connected to the car's network, the TSN configuration needs to be updated. This includes updates of the routing for 1+1 protection, bandwidth re-allocation for credit-based shaping, and re-calculation of the TSN schedule. To ensure safe operation, the TSN re-configuration is not done automatically after device discovery has finished. Re-calculation of the TSN schedule has to be triggered by the device in question via the northbound interface of the controller. As TSN schedule calculation is an NP-hard problem, calculating it fully in-car is not feasible. External schedule calculation, e.g. in a cloud environment, may not be possible at all times as it requires an Internet connection. We thus propose a hybrid approach with internal and external calculation.

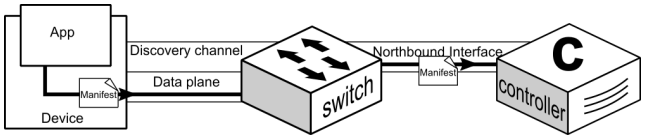
At first, the in-car controller updates the existing schedule by adding additional flows. The configuration of existing flows is not changed by the in-car controller. The resulting schedule provides guarantees for all safety-critical systems, but it is not optimal. This may result in a higher network utilization and might make it necessary to disable less critical systems. Afterwards, calculation of a schedule in a cloud service is triggered as soon as an Internet connection is available. This cloud service first checks if a schedule for the same constellation of devices and applications has already been calculated in the past and re-uses this schedule if possible. If a cached schedule is not available, a schedule is calculated, sent to the car and cached for further use. As an initial schedule has already been calculated by the car's controller, it is not necessary that the cloud service responds with a schedule immediately upon the controller's request. It thus can take the time to calculate an optimal schedule that can satisfy the requirements of all systems.

2) Device and Application Discovery: As additional participants like devices or applications may be added to the network at any time, means must be provided to re-configure the network and provide some sort of access control. We propose two similar discovery mechanisms that enable devices and applications to authenticate themselves and communicate their requirements regarding the network to the controller.

The discovery mechanism for devices is illustrated in Figure 6(a). When the device is connected to a switch for the first time, all traffic but the traffic on a dedicated discovery channel is dropped by the switch it is connected to. The



(a) Device discovery.



(b) Application discovery.

Fig. 6. Discovery mechanisms for devices and applications.

device communicates its identification and requirements by sending a signed manifest to the network controller via a broadcast message on the discovery channel. The switches forward all packets sent on this channel to the network controller. The controller then checks the manifest's integrity and trustworthiness by verifying if the manifest's signature is valid and originates from a trusted manufacturer.

If the signature is valid, the information in the manifest is stored in the local device inventory and the network is re-configured to meet the requirements of the device. This may include re-calculation of the TSN schedule. If the manifest indicates that the requirements are not static, e.g., if network flows need to be configured on demand, the device is given access to the northbound interface of the network controller.

One case where access to the northbound interface is necessary is the installation of a new application on a device. The device that hosts the application needs to notify the network controller about the application's properties like necessary network flows and access permissions. This happens similarly to the discovery of devices by sending a signed manifest to the controller. As shown in Figure 6(b), the manifest is not sent over a discovery channel by the application itself, but by the device via the northbound interface.

These discovery mechanisms configure communication paths and access to the network only. However, it may be necessary that ECUs discover services that are provided by other ECUs. For this purpose, existing mechanisms, e.g., as defined in Adaptive AutoSAR, may be used [29].

3) Failover Scenarios: There are three different cases of component failures that need to be addressed: failure of one of the backbone links, failure of one of the switches or both backbone links, and failure of the controller.

If one of the backbone link fails, all traffic between the front and the back switch needs to be routed through the other link. For safety-critical traffic scheduled by TSN, a pre-calculated outage schedule is applied. This guarantees transmission of the most critical traffic to ensure safe operation of the car. If the capacity of the single remaining backbone link is not sufficient to carry critical traffic, traffic generation by non-critical systems may be restricted or even stopped.

Coping with a failure of one of the switches or of both backbone links is more difficult. If a switch fails, all devices connected to it lose the connection to the car's network. If both backbone links fail, devices can only communicate with other devices connected to the same switch. It has to be ensured that safety-critical systems needed to stop the car in this situation, e.g., the brake system, can continue their operation in such a situation. These systems thus need to be able to operate even if connectivity is lost, or they need to be connected via a back-up network like a bus system.

The third case of failure is the failure of the network controller. If the controller fails, re-configuration of the network is no longer possible. As a precaution, the network controller configures backup flows and a backup TSN schedule on the switches. These flows and schedule represent the minimal configuration that guarantees safe operation of the most critical systems in the car. Similar to the failure of one of the backbone links, communication of non-critical systems may be restricted or stopped. If the switches lose connection to the controller, they apply the backup flows and schedule.

D. Security Considerations

An SDN architecture for automotive applications enables several novel use cases. However, it can also cause problems if security is not addressed properly. In the following, we take a look at what has to be done to secure the SDN architecture.

1) Security of Devices and Applications: The biggest potential threat comes from devices that are added to the car and applications that are installed. As described in Section IV-C, new devices only have access to the network for discovery purposes. Further access to the network is granted only if the device can provide a manifest that is signed by a trusted manufacturer. Applications have no access to the network at all until the device they are installed on sends a signed manifest to the controller via the northbound API. As the signatures of the manifest need to be checked, a central certification authority (CA) store must be provided that contains the CA certificates of all providers. Today, several public key infrastructures (PKIs) for automotive applications exist that could be extended accordingly. The car must either query this CA store when a device is installed or it must keep a local copy of it. In case a CA certificate becomes compromised, e.g., when a signing key is leaked or when a manufacturer loses its trust, the CA certificate must be revoked. Because of this, the car must refresh its local CA cache regularly. In addition to revoking a CA certificate, a way should be provided to exclude both new and existing devices and applications from the network in certain cases, e.g., if a product call back is issued for a device. One possible way would be the use of a mechanism similar to certificate revocation lists.

Even if an application is deemed trustworthy when it is installed, it may pose security threats afterwards, e.g., if an attacker alters an application. To prevent this, applications need to be authenticated and their integrity needs to be checked on every startup of the application, e.g., by code signing. In addition, applications need to be isolated from each other and their resource usage needs to be monitored. However, these

tasks are out-of-scope for the SDN architecture and are within responsibility of the devices that host the applications.

2) Network Security: Besides ensuring the security of devices and applications, the network has to be secured as well. As the network is no longer static in contrast to traditional E/E architectures, the segmentation of the network needs to be dynamic as well. It is not possible anymore to segment the network physically by using separate buses. For this reason, the proposed automotive SDN architecture supports only specific flows between devices and applications. These flows are derived by the network controller from the requirements stated in the manifest of devices and applications. These requirements include resources that the device or application needs to access. By installing flows only for these resources, the network is effectively segmented logically in slices of minimal size. This also removes the need for VLANs and similar technology. Flows that connect a device or application to the outside world are steered through a firewall. This is particularly necessary for the Internet upstream, V2X communication, or wireless networks like Bluetooth or Wi-Fi for the car's passengers.

Aside from protecting the devices and applications from each other as far as possible, the transmitted data needs to be protected as well. We propose to use MACsec to ensure integrity of the transmitted data. In addition to ensuring integrity, MACsec can also be used to encrypt the transmitted data if required. While this is generally desirable, it should be optional as it is not common practice in automotive applications. However, encrypting the traffic can help secure the network from intruders with physical access to the network. Hardware modules implementing MACsec are widely available, which facilitates integration. Keying material can be generated by the SDN controller comparable to P4-MACsec [30]. AUTOSAR Secure Onboard Communication [28] may be an alternative to MACsec.

The network controller requires special attention as well. As it is possible to request re-configuration of the network via the northbound interface, access has to be restricted. Devices and applications may gain permission according to the content of their manifest. Whenever a request is received through the northbound interface, its origination needs to be authenticated. Furthermore, care must be taken that none of the devices or applications exhausts the resources of the network.

V. Conclusion

The E/E architecture in vehicular application evolved from a single CAN bus with few control units to highly complex architectures with several buses and numerous control units. This trend will continue even further, especially due to advancements in driver assistance systems and automated driving systems. With these advancements, the complexity of the vehicular E/E architectures will increase even more, bringing traditional approaches to their limits.

This paper proposed an architecture for automotive Ethernet networks based on the SDN paradigm and with support for TSN. Using SDN enables re-configuration of the network if needed. Combining an SDN-controlled network with a discovery mechanism enables new use cases, e.g., integrating ECUs of a trailer into the network of the car. As the network is used by several systems that have different requirements concerning bandwidth and transmission reliability, traffic is divided into traffic classes ranging from best-effort traffic to hard real-time traffic that uses TSN. Calculation of TSN schedules is performed using a hybrid calculation approach that computes a basic TSN schedule in the car first and then computes an optimal schedule in a cloud environment. Separation of systems that are not supposed to communicate with each other is ensured by supporting only specific network flows. Access to the network is controlled by using signatures and a PKI with CA certificates of trusted device manufacturers. Failures of links, switches or the central controller are dealt with by pre-computing backup network flows and TSN schedules.

The architecture is a visionary sketch. Domain-specific protocols, data formats, and interfaces are needed. Preferentially, existing technology should be extended to facilitate integration and adoption of the architecture. The next steps are prototypical implementations to validate the overall concept.

Acknowledgment

The authors thank Mark Schmidt for valuable input and fruitful discussions.

References

[1] BMW AG, "The Road to Autonomous Driving," 2019, accessed on March 5th, 2020. [Online]. Available: https://www.bmw.com/en/innovation/the-development-of-self-driving-cars.html

[2] Audi AG, "Future cars: Relaxing in the Audi AI:ME," 2019, accessed on March 5th, 2020. [Online]. Available: https://www.audi.com/en/experience-audi/models-and-technology/concept-cars/audi-aime.html

[3] International Standards Organisation, "ISO 11898: Road Vehicles: Interchange of Digital Information: Controller Area Network (CAN) for High-speed Communication," 1993.

[4] FlexRay Consortium, "FlexRay Communications System Protocol Specification Version 2.1," 2005.

[5] LIN Consortium, "LIN Protocol Specification Package , Revision 1.3," 2002.

[6] A. Grzemba, *MOST: The Automotive Multimedia Network*. Franzis Verlag, 2012.

[7] W. Braun and M. Menth, "Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices," *Future Internet*, vol. 6, no. 2, pp. 302–336, 2014.

[8] J. L. Messenger, "Time-Sensitive Networking: An Introduction," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 29–33, 2018.

[9] Open Networking Foundation, "OpenFlow Switch Specification," 2012.

[10] 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 Computer Communications Review*, vol. 44, no. 3, pp. 87–95, Jul. 2014.

[11] "IEEE Standard for Local and Metropolitan Area Networks - Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic," IEEE Std. 802.1Qbv-2015, 2015.

[12] "IEEE Standard for Local and Metropolitan Area Networks - Bridges and Bridged Networks - Amendment 26: Frame Preemption," IEEE Std. 802.1Qbu-2016, 2016.

[13] "IEEE Standard for Local and Metropolitan Area Networks - Frame Replication and Elimination for Reliability," IEEE Std. 802.1CB-2017, 2017.

[14] N. G. Nayak, F. Duerr, and K. Rothermel, "Time-Sensitive Software-Defined Network (TSSDN) for Real-Time Applications," in *International Conference on Real-Time Networks and Systems (RTNS)*, 2016.

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

[16] D. Pannell, L. Chen, J. Dorr, W. Lo, M. Potts, H. Zinner, and A. Zu, "Use Cases - IEEE P802.1DG V0.4," Jul. 2019.

[17] A. Lock, N. Tracey, and D. Zerfowski, "Entering New Worlds: New E/E Architectures With Vehicle Computers Offer New Opportunities," ETAS GmbH, Tech. Rep., 2020.

[18] L. L. Bello, "The Case for Ethernet in Automotive Communications," *SIGBED Rev.*, vol. 8, no. 4, pp. 7–15, Dec. 2011.

[19] L. L. Bello, R. Mariani, S. Mubeen, and S. Saponara, "Recent Advances and Trends in On-Board Embedded and Networked Automotive Systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1038–1051, 2019.

[20] G. Alderisi, G. Iannizzotto, and L. L. Bello, "Towards IEEE 802.1 Ethernet AVB for Advanced Driver Assistance Systems: A preliminary assessment," in *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, Sep. 2012.

[21] S. Brunner, J. Roeder, M. Kucera, and T. Waas, "Automotive E/E-Architecture Enhancements by usage of Ethernet TSN," in *Workshop on Intelligent Solutions in Embedded Systems (WISES)*, Jun. 2017, pp. 9–13.

[22] J. Migge, J. Villanueva, N. Navet, and M. Boyer, "Insights on the Performance and Configuration of AVB and TSN in Automotive Ethernet Networks," *Proc. Embedded Real-Time Software and Systems (ERTS 2018)*, 2018.

[23] P. Fussey and G. Parisis, "Poster: An In-Vehicle Software Defined Network Architecture for Connected and Automated Vehicles," *Proceedings of the 2nd ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services*, 2017.

[24] T. Hackel, P. Meyer, F. Korf, and T. C. Schmidt, "Software-Defined Networks Supporting Time-Sensitive In-Vehicular Communication," in *IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, 2019, pp. 1–5.

[25] K. Halba and C. Mahmoudi, "In-Vehicle Software Defined Networking: An Enabler for Data Interoperability," in *Proceedings of the 2nd International Conference on Information System and Data Mining (ICISDM '18)*, 2018.

[26] K. Halba, C. Mahmoudi, and E. Griffor, "Robust Safety for Autonomous Vehicles through Reconfigurable Networking," *Electronic Proceedings in Theoretical Computer Science*, vol. 269, p. 48–58, 4 2018.

[27] S. Apostu, O. Burkacky, J. Deichmann, and G. Doll, "Automotive Software and Electrical/Electronic Architecture: Implications for OEMs," McKinsey & Co, Tech. Rep., Apr. 2020.

[28] AUTOSAR, "Specification of Secure Onboard Communication - AUTOSAR CPRelease 4.3.1," 2017.

[29] ——, "Specification of Service Discovery - AUTOSAR CPRelease 4.3.1," 2017.

[30] F. Hauser, M. Schmidt, M. Haeberle, and M. Menth, "P4-MACsec: Dynamic Topology Monitoring and Data Layer Protection With MACsec in P4-Based SDN," *IEEE Access*, vol. 8, pp. 58 845–58 858, 2020.